Oracle® Database Security Guide





Oracle Database Security Guide, 19c

E96299-50

Copyright © 1996, 2024, Oracle and/or its affiliates.

Primary Author: Patricia Huey

Contributing Authors: Sumit Jeloka

Contributors: Suraj Adhikari, Thomas Baby, Tammy Bednar, Todd Bottger, Sanjay Bharadwaj, Leo Cloutier, Sudha Duraiswamy, Naveen Gopal, Rishabh Gupta, Yong Hu, Srinidhi Kayoor, Peter Knaggs, Andre Kruklikov, Sanjay Kulhari, Anup A. Kumar, Bryn Llewellyn, Dah-Yoh Lim, Rahil Mir, Hari Mohankumar, Gopal Mulagund, Abhishek Munnolimath, Paul Needham, Robert Pang, Dilip Raj, Kumar Rajamani, Kathy Rich, Saikat Saha, Vipin Samar, Saravana Soundararajan, James Spiller, Srividya Tata, Kamal Tbeileh, Can Tuzla, Anand Verma, Patrick Wheeler, Peter H. Wong

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, MySQL, and NetSuite 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

	Audience Documentation Accessibility Related Documents Conventions					
	Changes in This Release for Oracle Database Security Guide					
	Changes in Oracle Database Security 19c Updates to Oracle Database Security 19c	liv lix				
1	Introduction to Oracle Database Security					
	1.1 About Oracle Database Security1.2 Additional Oracle Database Security Products	1-1 1-2				
Par	Managing User Authentication and Authorization					
Par 2	Managing User Authentication and Authorization Managing Security for Oracle Database Users					
		2-1				
	Managing Security for Oracle Database Users	2-1 2-2				
	Managing Security for Oracle Database Users 2.1 About User Security					
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts	2-2				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users	2-2 2-2				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users 2.2.1.1 About Common Users 2.2.1.2 How Plugging in PDBs Affects CDB Common Users 2.2.1.3 About Local Users	2-2 2-2 2-3 2-4 2-5				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users 2.2.1.1 About Common Users 2.2.1.2 How Plugging in PDBs Affects CDB Common Users 2.2.1.3 About Local Users 2.2.2. Who Can Create User Accounts?	2-2 2-2 2-3 2-4 2-5 2-6				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users 2.2.1.1 About Common Users 2.2.1.2 How Plugging in PDBs Affects CDB Common Users 2.2.1.3 About Local Users 2.2.2 Who Can Create User Accounts? 2.2.3 Creating a New User Account That Has Minimum Database Privileges	2-2 2-2 2-3 2-4 2-5 2-6 2-6				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users 2.2.1.1 About Common Users 2.2.1.2 How Plugging in PDBs Affects CDB Common Users 2.2.1.3 About Local Users 2.2.2 Who Can Create User Accounts? 2.2.3 Creating a New User Account That Has Minimum Database Privileges 2.2.4 Restrictions on Creating the User Name for a New Account	2-2 2-2 2-3 2-4 2-5 2-6 2-6 2-7				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users 2.2.1.1 About Common Users 2.2.1.2 How Plugging in PDBs Affects CDB Common Users 2.2.1.3 About Local Users 2.2.2 Who Can Create User Accounts? 2.2.3 Creating a New User Account That Has Minimum Database Privileges 2.2.4 Restrictions on Creating the User Name for a New Account 2.2.4.1 Uniqueness of User Names	2-2 2-2 2-3 2-4 2-5 2-6 2-6 2-7 2-8				
	Managing Security for Oracle Database Users 2.1 About User Security 2.2 Creating User Accounts 2.2.1 About Common Users and Local Users 2.2.1.1 About Common Users 2.2.1.2 How Plugging in PDBs Affects CDB Common Users 2.2.1.3 About Local Users 2.2.2 Who Can Create User Accounts? 2.2.3 Creating a New User Account That Has Minimum Database Privileges 2.2.4 Restrictions on Creating the User Name for a New Account	2-2 2-2 2-3 2-4 2-5 2-6 2-6 2-7				



2.2.5 Assignment of User Passwords

2-9

	2.2.6	Defa	ult Tablespace for the User	2-9
	2	2.2.6.1	About Assigning a Default Tablespace for a User	2-9
	2	2.2.6.2	DEFAULT TABLESPACE Clause for Assigning a Default Tablespace	2-10
	2.2.7	Tabl	espace Quotas for a User	2-10
	2	2.2.7.1	About Assigning a Tablespace Quota for a User	2-11
	2	2.2.7.2	CREATE USER Statement for Assigning a Tablespace Quota	2-11
	2	2.2.7.3	Restriction of the Quota Limits for User Objects in a Tablespace	2-12
	2	2.2.7.4	Grants to Users for the UNLIMITED TABLESPACE System Privilege	2-12
	2.2.8	Tem	porary Tablespaces for the User	2-12
	2	2.2.8.1	About Assigning a Temporary Tablespace for a User	2-12
	2	2.2.8.2	TEMPORARY TABLESPACE Clause for Assigning a Temporary Tablespace	2-13
	2.2.9	Prof	les for the User	2-13
	2.2.1	.0 Cre	eation of a Common User or a Local User	2-14
	2	2.2.10.1	About Creating Common User Accounts	2-15
	2	2.2.10.2	CREATE USER Statement for Creating a Common User Account	2-15
	2	2.2.10.3	About Creating Local User Accounts	2-16
	2	2.2.10.4	CREATE USER Statement for Creating a Local User Account	2-17
	2.2.1	.1 Cre	ating a Default Role for the User	2-17
2.3	Alt	ering U	ser Accounts	2-18
	2.3.1	. Abo	ut Altering User Accounts	2-18
	2.3.2	Meth	nods of Altering Common or Local User Accounts	2-18
	2.3.3	Cha	nging Non-SYS User Passwords	2-19
	2	2.3.3.1	About Changing Non-SYS User Passwords	2-20
	2	2.3.3.2	Using the PASSWORD Command or ALTER USER Statement to Change a Password	2-20
	2.3.4	Cha	nging the SYS User Password	2-21
	2	2.3.4.1	About Changing the SYS User Password	2-21
	2	2.3.4.2	ORAPWD Utility for Changing the SYS User Password	2-22
2.4	Co	nfigurin	g User Resource Limits	2-22
	2.4.1	. Abo	ut User Resource Limits	2-23
	2.4.2	Туре	es of System Resources and Limits	2-23
	2	2.4.2.1	Limits to the User Session Level	2-24
	2	2.4.2.2	Limits to Database Call Levels	2-24
	2	2.4.2.3	Limits to CPU Time	2-24
	2	2.4.2.4	Limits to Logical Reads	2-24
	2	2.4.2.5	Limits to Other Resources	2-25
	2.4.3	Valu	es for Resource Limits of Profiles	2-25
	2.4.4	Man	aging Resources with Profiles	2-26
	2	2.4.4.1	About Profiles	2-26
	2	2.4.4.2	ora_stig_profile User Profile	2-27
	2	2.4.4.3	Creating a Profile	2-27



	2-28
	2-28
	2-29
	2-29
	2-29
	2-30
nnected to the Database	2-30
	2-31
oase	2-31
	2-31
	2-32
	2-35
	2-36
	2-36
s and Profiles	2-37
	2-38
	2-38
	2-39
	2-40
	3-2
	3-2
ctions?	3-3
	3-4
use	3-4
	3-4
	3-5
ords	3-6
	3-6
file Limits	3-8
curity Settings	3-8
counts	3-9
cified Number of Failed	3-10
PROFILE Statement	3-10
	3-11
asswords	3-11
n	3-12
LE Statement to Set a	3-13
	3-13



3.	2.4.14	Password Change Life Cycle	3-13
3.	2.4.15	PASSWORD_LIFE_TIME Profile Parameter Low Value	3-15
3.2.5	Mana	aging Gradual Database Password Rollover for Applications	3-16
3.	2.5.1	About Managing Gradual Database Password Rollover for Applications	3-17
3.	2.5.2	Password Change Life Cycle During a Gradual Database Password Rollover	3-18
3.	2.5.3	Enabling the Gradual Database Password Rollover	3-19
3.	2.5.4	Changing a Password to Begin the Gradual Database Password Rollover Period	3-20
3.	2.5.5	Changing a Password During the Gradual Database Password Rollover Period	3-21
3.	2.5.6	Ending the Password Rollover Period	3-22
3.	2.5.7	Database Behavior During the Gradual Password Rollover Period	3-22
3.	2.5.8	Database Server Behavior After the Password Rollover Period Ends	3-23
3.	2.5.9	Guideline for Handling Compromised Passwords	3-23
3.	2.5.10	How Gradual Database Password Rollover Works During Oracle Data Pump Exports	3-24
3.	2.5.11	Using Gradual Database Password Rollover in an Oracle Data Guard Environment	3-24
3.	2.5.12	Finding Users Who Still Use Their Old Passwords	3-24
3.2.6	Mana	aging the Complexity of Passwords	3-25
3.	2.6.1	About Password Complexity Verification	3-26
3.	2.6.2	How Oracle Database Checks the Complexity of Passwords	3-26
3.	2.6.3	Who Can Use the Password Complexity Functions?	3-26
3.	2.6.4	verify_function_11G Function Password Requirements	3-26
3.	2.6.5	ora12c_verify_function Password Requirements	3-27
3.	2.6.6	ora12c_strong_verify_function Function Password Requirements	3-27
3.	2.6.7	ora12c_stig_verify_function Password Requirements	3-28
3.	2.6.8	About Customizing Password Complexity Verification	3-28
3.	2.6.9	Enabling Password Complexity Verification	3-29
3.2.7	Mana	aging Password Case Sensitivity	3-30
3.	2.7.1	SEC_CASE_SENSITIVE_LOGON Parameter and Password Case Sensitivity	3-30
3.	2.7.2	Using the ALTER SYSTEM Statement to Enable Password Case Sensitivity	3-31
3.	2.7.3	Management of Case Sensitivity for Secure Role Passwords	3-31
3.	2.7.4	Management of Password Versions of Users	3-32
3.	2.7.5	Finding and Resetting User Passwords That Use the 10G Password Version	3-34
3.	2.7.6	How Case Sensitivity Affects Password Files	3-36
3.	2.7.7	How Case Sensitivity Affects Passwords Used in Database Link Connections	3-37
3.2.8	Ensu Versi	ring Against Password Security Threats by Using the 12C Password on	3-37



	3.2.8	.1 About the 12C Version of the Password Hash	3-37
	3.2.8	.2 Oracle Database 12C Password Version Configuration Guidelines	3-38
	3.2.8	.3 Configuring Oracle Database to Use the 12C Password Version Exclusively	3-40
	3.2.8	.4 How Server and Client Logon Versions Affect Database Links	3-42
	3.2.8	.5 Configuring Oracle Database Clients to Use the 12C Password Version	
		Exclusively	3-43
	3.2.9 N	lanaging the Secure External Password Store for Password Credentials	3-44
	3.2.9	.1 About the Secure External Password Store	3-44
	3.2.9	.2 How Does the Secure External Password Store Work?	3-45
	3.2.9	.3 About Configuring Clients to Use the Secure External Password Store	3-46
	3.2.9	.4 Configuring a Client to Use the Secure External Password Store	3-46
	3.2.9	.5 Example: Sample sqlnet.ora File with Wallet Parameters Set	3-47
	3.2.9	.6 Managing External Password Store Credentials	3-48
	3.2.10	Managing Passwords for Administrative Users	3-50
	3.2.1	0.1 About Managing Passwords for Administrative Users	3-50
	3.2.1	0.2 Setting the LOCK and EXPIRED Status of Administrative Users	3-50
	3.2.1	0.3 Password Profile Settings for Administrative Users	3-51
	3.2.1	0.4 Last Successful Login Time for Administrative Users	3-51
	3.2.1	0.5 Management of the Password File of Administrative Users	3-51
	3.2.1	0.6 Migration of the Password File of Administrative Users	3-52
	3.2.1	0.7 How the Multitenant Option Affects Password Files for Administrative Users	3-52
	3.2.1	0.8 Password Complexity Verification Functions for Administrative Users	3-53
3.3	Authen	tication of Database Administrators	3-53
	3.3.1 A	bout Authentication of Database Administrators	3-54
	3.3.2	trong Authentication, Centralized Management for Administrators	3-54
	3.3.2	.1 About Strong Authentication for Database Administrators	3-54
	3.3.2	.2 Configuring Directory Authentication for Administrative Users	3-54
	3.3.2	.3 Configuring Kerberos Authentication for Administrative Users	3-55
	3.3.2	.4 Configuring User Authentication with Transport Layer Security	3-56
	3.3.3 A	uthentication of Database Administrators by Using the Operating System	3-58
	3.3.4 A	uthentication of Database Administrators by Using Their Passwords	3-59
	3.3.5 F	tisks of Using Password Files for Database Administrator Authentication	3-60
3.4	Databa	se Authentication of Users	3-60
	3.4.1 A	bout Database Authentication of Users	3-61
	3.4.2 A	dvantages of Database Authentication	3-62
	3.4.3	creating Users Who Are Authenticated by the Database	3-62
3.5	Schem	a-Only Accounts	3-63
	3.5.1 A	bout Schema-Only Accounts	3-63
	3.5.2	creating a Schema-Only Account	3-64
	3.5.3 A	Itering a Schema-Only Account	3-64
3.6	Operat	ing System Authentication of Users	3-64



3.7	Netw	ork Au	thentication of Users	3-65
	3.7.1	Authe	ntication with Transport Layer Security	3-66
	3.7.2	Authe	ntication with Third-Party Services	3-66
	3.7	'.2.1	About Authentication Using Third-Party Services	3-66
	3.7	'.2.2	Authentication with Kerberos	3-66
	3.7	'.2.3	Authentication with RADIUS	3-67
	3.7	'.2.4	Authentication with Directory-Based Services	3-67
	3.7	'.2.5	Authentication with Public Key Infrastructure	3-67
3.8	Conf	iguring	Operating System Users for a PDB	3-68
	3.8.1	About	Configuring Operating System Users for a PDB	3-68
	3.8.2	Config	guring an Operating System User for a PDB	3-69
3.9	Glob	al User	Authentication and Authorization	3-70
	3.9.1	About	Configuring Global User Authentication and Authorization	3-70
	3.9.2	Config	guration of Users Who Are Authorized by a Directory Service	3-71
	3.9	0.2.1	Creating a Global User Who Has a Private Schema	3-71
	3.9	.2.2	Creating Multiple Enterprise Users Who Share Schemas	3-71
	3.9.3	Advar	ntages of Global Authentication and Global Authorization	3-72
3.1	O Cor	nfigurin	g an External Service to Authenticate Users and Passwords	3-72
	3.10.1	Abou	ut External Authentication	3-73
	3.10.2	Adva	intages of External Authentication	3-73
	3.10.3	Enab	oling External Authentication	3-74
	3.10.4	Crea	ting a User Who Is Authenticated Externally	3-74
	3.10.5	Auth	entication of User Logins By Using the Operating System	3-75
	3.10.6	Auth	entication of User Logins Using Network Authentication	3-75
3.1	L Mul	titier Au	thentication and Authorization	3-75
3.1	2 Adr	ninistra	tion and Security in Clients, Application Servers, and Database Servers	3-76
3.1	3 Pre	serving	User Identity in Multitiered Environments	3-77
	3.13.1	Midd	le Tier Server Use for Proxy Authentication	3-78
	3.1	.3.1.1	About Proxy Authentication	3-79
	3.1	.3.1.2	Advantages of Proxy Authentication	3-79
	3.1	.3.1.3	Who Can Create Proxy User Accounts?	3-80
	3.1	.3.1.4	Guidelines for Creating Proxy User Accounts	3-80
	3.1	.3.1.5	Creating Proxy User Accounts and Authorizing Users to Connect Through Them	3-81
	3.1	.3.1.6	Proxy User Accounts and the Authorization of Users to Connect Through Them	3-82
	3.1	3.1.7	Using Proxy Authentication with the Secure External Password Store	3-82
	3.1	3.1.8	How the Identity of the Real User Is Passed with Proxy Authentication	3-83
	3.1	3.1.9	Limits to the Privileges of the Middle Tier	3-84
	3.1	3.1.10	Authorizing a Middle Tier to Proxy and Authenticate a User	3-85
	3.1	.3.1.11	Authorizing a Middle Tier to Proxy a User Authenticated by Other Means	3-8
	3.1	.3.1.12	Reauthenticating a User Through the Middle Tier to the Database	3-86



	3.1.	3.1.13	Using Password-Based Proxy Authentication	3-86
	3.13	3.1.14	Using Proxy Authentication with Enterprise Users	3-87
	3.13.2	Using	Client Identifiers to Identify Application Users Unknown to the Database	3-88
	3.13	3.2.1	About Client Identifiers	3-88
	3.13	3.2.2	How Client Identifiers Work in Middle Tier Systems	3-88
	3.13	3.2.3	Use of the CLIENT_IDENTIFIER Attribute to Preserve User Identity	3-89
	3.13		Use of the CLIENT_IDENTIFIER Independent of Global Application Context	3-89
	3.13		Setting the CLIENT_IDENTIFIER Independent of Global Application Context	3-90
	3.13		Use of the DBMS_SESSION PL/SQL Package to Set and Clear the Client Identifier	3-91
	3.13	3.2.7	Enabling the CLIENTID_OVERWRITE Event System-Wide	3-91
	3.13	3.2.8	Enabling the CLIENTID_OVERWRITE Event for the Current Session	3-92
	3.13	3.2.9	Disabling the CLIENTID_OVERWRITE Event	3-92
	3.14 Use	r Auther	ntication Data Dictionary Views	3-92
4			rivilege and Role Authorization ges and Roles	4-2
			Be Granted Privileges?	4-3
			cle Multitenant Option Affects Privileges	4-3
			ministrative Privileges	4-4
	4.4.1	About A	Administrative Privileges	4-5
	4.4.2	Grants	of Administrative Privileges to Users	4-5
	4.4.3	SYSDB	BA and SYSOPER Privileges for Standard Database Operations	4-5
	4.4.4	Forcing	oracle Users to Enter a Password When Logging in as SYSDBA	4-6
	4.4.5	SYSBA	CKUP Administrative Privilege for Backup and Recovery Operations	4-6
	4.4.6	SYSDG	Administrative Privilege for Oracle Data Guard Operations	4-7
	4.4.7	SYSKM	Administrative Privilege for Transparent Data Encryption	4-8
	4.4.8	SYSRA	C Administrative Privilege for Oracle Real Application Clusters	4-9
	4.5 Mana	ging Sy	stem Privileges	4-10
	4.5.1	About S	System Privileges	4-11
	4.5.2	Why Is	It Important to Restrict System Privileges?	4-11
	4.5.	2.1 A	bout the Importance of Restricting System Privileges	4-11
	4.5.		Iser Access to Objects in the SYS Schema	4-11
	4.5.3		and Revokes of System Privileges	4-12
			an Grant or Revoke System Privileges?	4-12
	4.5.5		ANY Privileges and the PUBLIC Role	4-13
			mmonly and Locally Granted Privileges	4-13
	4.6.1		Commonly and Locally Granted Privileges	4-14
	4.6.2		ommonly Granted System Privileges Work	4-14
	4.6.3	How Co	ommonly Granted Object Privileges Work	4-15



	4.6.4	Gran	iting or Revoking Privileges to Access a PDB	4-16
	4.6.5	Exar	nple: Granting a Privilege in a Multitenant Environment	4-16
	4.6.6	Enab	oling Common Users to View CONTAINER_DATA Object Information	4-16
	4.6	6.6.1	Viewing Data About the Root, CDB, and PDBs While Connected to the	
			Root	4-16
		5.6.2	Enabling Common Users to Query Data in Specific PDBs	4-17
4.7	Mana		Common Roles and Local Roles	4-18
	4.7.1		ut Common Roles and Local Roles	4-19
	4.7.2	_	Common Roles Work	4-19
	4.7.3	How	the PUBLIC Role Works in a Multitenant Environment	4-20
	4.7.4	Privil	leges Required to Create, Modify, or Drop a Common Role	4-20
	4.7.5	Rule	s for Creating Common Roles	4-20
	4.7.6	Crea	ting a Common Role	4-21
	4.7.7	Rule	s for Creating Local Roles	4-21
	4.7.8	Crea	ting a Local Role	4-21
	4.7.9	Role	Grants and Revokes for Common Users and Local Users	4-22
4.8	Mana	aging	User Roles	4-23
	4.8.1	Abou	ut User Roles	4-23
	4.8	3.1.1	What Are User Roles?	4-24
	4.8	3.1.2	The Functionality of Roles	4-24
	4.8	3.1.3	Properties of Roles and Why They Are Advantageous	4-25
	4.8	3.1.4	Typical Uses of Roles	4-26
	4.8	3.1.5	Common Uses of Application Roles	4-26
	4.8	3.1.6	Common Uses of User Roles	4-27
	4.8	3.1.7	How Roles Affect the Scope of a User's Privileges	4-27
	4.8	3.1.8	How Roles Work in PL/SQL Blocks	4-27
	4.8	3.1.9	How Roles Aid or Restrict DDL Usage	4-28
	4.8	3.1.10	How Operating Systems Can Aid Roles	4-29
	4.8	3.1.11	How Roles Work in a Distributed Environment	4-29
	4.8.2	Pred	efined Roles in an Oracle Database Installation	4-29
	4.8.3	Crea	iting a Role	4-34
	4.8	3.3.1	About the Creation of Roles	4-34
	4.8	3.3.2	Creating a Role That Is Authenticated With a Password	4-35
	4.8	3.3.3	Creating a Role That Has No Password Authentication	4-36
	4.8	3.3.4	Creating a Role That Is External or Global	4-36
	4.8	3.3.5	Altering a Role	4-37
	4.8.4	Spec	cifying the Type of Role Authorization	4-37
	4.8	3.4.1	Authorizing a Role by Using the Database	4-38
	4.8	3.4.2	Authorizing a Role by Using an Application	4-38
	4.8	3.4.3	Authorizing a Role by Using an External Source	4-39
	4.8	3.4.4	Authorizing a Role by Using the Operating System	4-39
	4.8	3.4.5	Authorizing a Role by Using a Network Client	4-39



	4.8	.4.6	Authorizing a Global Role by an Enterprise Directory Service	4-40
	4.8.5	Grant	ing and Revoking Roles	4-40
	4.8	.5.1	About Granting and Revoking Roles	4-40
	4.8	.5.2	Who Can Grant or Revoke Roles?	4-41
	4.8	.5.3	Granting and Revoking Roles to and from Program Units	4-41
	4.8.6	Dropp	ping Roles	4-42
	4.8.7	Restr	icting SQL*Plus Users from Using Database Roles	4-42
	4.8	.7.1	Potential Security Problems of Using Ad Hoc Tools	4-42
	4.8	.7.2	How the PRODUCT_USER_PROFILE System Table Can Limit Roles	4-43
	4.8	.7.3	How Stored Procedures Can Encapsulate Business Logic	4-43
	4.8.8	Role	Privileges and Secure Application Roles	4-44
4.9	Rest	ricting	Operations on PDBs Using PDB Lockdown Profiles	4-44
	4.9.1	Abou	t PDB Lockdown Profiles	4-45
	4.9.2	PDB	Lockdown Profile Inheritance	4-46
	4.9.3	Defau	ılt PDB Lockdown Profiles	4-47
	4.9.4	Creat	ing a PDB Lockdown Profile	4-47
	4.9.5	Enab	ling or Disabling a PDB Lockdown Profile	4-49
	4.9.6	Dropp	oing a PDB Lockdown Profile	4-50
4.10	Mar	naging	Object Privileges	4-51
	4.10.1	Abo	ut Object Privileges	4-51
	4.10.2	Who	Can Grant Object Privileges?	4-52
	4.10.3	Gra	nts and Revokes of Object Privileges	4-52
	4.1	0.3.1	About Granting and Revoking Object Privileges	4-53
	4.1	0.3.2	How the ALL Clause Grants or Revokes All Available Object Privileges	4-53
	4.10.4	REA	D and SELECT Object Privileges	4-53
	4.1	0.4.1	About Managing READ and SELECT Object Privileges	4-53
	4.1	0.4.2	Enabling Users to Use the READ Object Privilege to Query Any Table in the Database	4-54
	4.1	0.4.3	Restrictions on the READ and READ ANY TABLE Privileges	4-54
	4.10.5	Obje	ect Privilege Use with Synonyms	4-55
	4.10.6	Sha	ring Application Common Objects	4-56
	4.1	0.6.1	Metadata-Linked Application Common Objects	4-56
	4.1	0.6.2	Data-Linked Application Common Objects	4-57
	4.1	0.6.3	Extended Data-Linked Application Common Objects	4-58
4.11	Tab	le Privi	leges	4-58
	4.11.1	How	Table Privileges Affect Data Manipulation Language Operations	4-59
	4.11.2	How	Table Privileges Affect Data Definition Language Operations	4-59
4.12	. Vie	w Privil	eges	4-60
	4.12.1	Privi	leges Required to Create Views	4-60
	4.12.2	Privi	leges to Query Views in Other Schemas	4-60
	4.12.3	The	Use of Views to Increase Table Security	4-61
4.13	B Pro	cedure	Privileges	4-61



4	1.13.1	The	Use of the EXECUTE Privilege for Procedure Privileges	4-62
4	1.13.2	Proc	edure Execution and Security Domains	4-62
4	1.13.3	Syst	em Privileges Required to Create or Replace a Procedure	4-62
2	1.13.4	Syst	em Privileges Required to Compile a Procedure	4-63
2	1.13.5	How	Procedure Privileges Affect Packages and Package Objects	4-63
	4.13	3.5.1	About the Effect of Procedure Privileges on Packages and Package	
			Objects	4-63
	4.13	3.5.2	Example: Procedure Privileges Used in One Package	4-64
	4.13	3.5.3	Example: Procedure Privileges and Package Objects	4-64
4.14	Type	Privil	eges	4-65
2	1.14.1	Syst	em Privileges for Named Types	4-66
4	1.14.2	Obje	ct Privileges for Named Types	4-66
4	1.14.3	Meth	nod Execution Model for Named Types	4-66
4	1.14.4	Privi	leges Required to Create Types and Tables Using Types	4-66
4	1.14.5	Exar	nple: Privileges for Creating Types and Tables Using Types	4-67
4	1.14.6	Privi	leges on Type Access and Object Access	4-68
4	1.14.7	Type	Dependencies	4-69
4.15	Gran	its of l	Jser Privileges and Roles	4-70
4	1.15.1	Gran	nting System Privileges and Roles to Users and Roles	4-70
	4.15	5.1.1	Privileges for Grants of System Privileges and Roles to Users and Roles	4-70
	4.15	5.1.2	Example: Granting a System Privilege and a Role to a User	4-71
	4.15	5.1.3	Example: Granting the EXECUTE Privilege on a Directory Object	4-71
	4.15	5.1.4	Use of the ADMIN Option to Enable Grantee Users to Grant the Privilege	4-71
	4.15	5.1.5	Creating a New User with the GRANT Statement	4-71
2	1.15.2	Gran	nting Object Privileges to Users and Roles	4-72
	4.15	5.2.1	About Granting Object Privileges to Users and Roles	4-72
	4.15	5.2.2	How the WITH GRANT OPTION Clause Works	4-73
	4.15	5.2.3	Grants of Object Privileges on Behalf of the Object Owner	4-73
	4.15	5.2.4	Grants of Privileges on Columns	4-75
	4.15	5.2.5	Row-Level Access Control	4-75
4.16	Revo	okes o	f Privileges and Roles from a User	4-75
4	1.16.1	Revo	okes of System Privileges and Roles	4-76
4	1.16.2	Revo	okes of Object Privileges	4-76
	4.16	6.2.1	About Revokes of Object Privileges	4-76
	4.16	5.2.2	Revokes of Multiple Object Privileges	4-77
	4.16	5.2.3	Revokes of Object Privileges on Behalf of the Object Owner	4-77
	4.16	5.2.4	Revokes of Column-Selective Object Privileges	4-78
	4.16	5.2.5	Revokes of the REFERENCES Object Privilege	4-78
2	1.16.3	Caso	cading Effects of Revoking Privileges	4-79
	4.16	5.3.1	Cascading Effects When Revoking System Privileges	4-79
	4.16	5.3.2	Cascading Effects When Revoking Object Privileges	4-79
4.17	Gran	its and	Revokes of Privileges to and from the PUBLIC Role	4-80



	nts of Roles Using the Operating System or Network	4-80
4.18.1	About Granting Roles Using the Operating System or Network	4-81
4.18.2	Operating System Role Identification	4-82
4.18.3	Operating System Role Management	4-83
4.18.4	Role Grants and Revokes When OS_ROLES Is Set to TRUE	4-83
4.18.5	Role Enablements and Disablements When OS_ROLES Is Set to TRUE	4-83
4.18.6	Network Connections with Operating System Role Management	4-83
4.19 How	Grants and Revokes Work with SET ROLE and Default Role Settings	4-84
4.19.1	When Grants and Revokes Take Effect	4-84
4.19.2	How the SET ROLE Statement Affects Grants and Revokes	4-84
4.19.3	Specifying the Default Role for a User	4-85
4.19.4	The Maximum Number of Roles That a User Can Have Enabled	4-85
4.20 Use	Privilege and Role Data Dictionary Views	4-85
4.20.1	Data Dictionary Views to Find Information about Privilege and Role Grants	4-86
4.20.2	Query to List All System Privilege Grants	4-88
4.20.3	Query to List All Role Grants	4-88
4.20.4	Query to List Object Privileges Granted to a User	4-89
4.20.5	Query to List the Current Privilege Domain of Your Session	4-89
4.20.6	Query to List Roles of the Database	4-90
	Quary to List Information About the Drivillage Demains of Delec	4-91
4.20.7 Performi	Query to List Information About the Privilege Domains of Roles Ing Privilege Analysis to Identify Privilege Use	
Performi		5-1
Performi	ng Privilege Analysis to Identify Privilege Use	5-1
Performi	ng Privilege Analysis to Identify Privilege Use Is Privilege Analysis?	5-1 5-2
Performi 5.1 What 5.1.1 5.1.2	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis	5-1 5-2 5-2
Performi 5.1 What 5.1.1 5.1.2 5.1.2	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis	5-1 5-2 5-2 5-2
Performi 5.1 What 5.1.1 5.1.2 5.1.2	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice	5-1 5-2 5-2 5-2
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications	5-1 5-2 5-2 5-2 5-3 5-3
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis?	5-1 5-2 5-2 5-2 5-3 5-3 5-3
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis	5-1 5-2 5-2 5-2 5-3 5-3 5-3 5-4
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis?	5-1 5-2 5-2 5-3 5-3 5-3 5-4 5-4
Derformi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects	5-1 5-2 5-2 5-3 5-3 5-3 5-4 5-4 5-5
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies	5-1 5-2 5-2 5-2 5-3 5-3 5-4 5-4 5-5
Derformi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat 5.2.1	Ing Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies About Creating and Managing Privilege Analysis Policies	5-1 5-2 5-2 5-3 5-3 5-3 5-4 5-4 5-5 5-5
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat 5.2.1 5.2.2	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies About Creating and Managing Privilege Analysis Policies General Steps for Managing Privilege Analysis	5-1 5-2 5-2 5-2 5-3 5-3 5-4 5-4 5-5 5-5 5-5
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat 5.2.1 5.2.2 5.2.3	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies About Creating and Managing Privilege Analysis Policies General Steps for Managing Privilege Analysis Creating a Privilege Analysis Policies Examples of Creating Privilege Analysis Policies	5-1 5-2 5-2 5-3 5-3 5-3 5-4 5-4 5-5 5-5 5-5 5-6 5-8
Performi 5.1 What 5.1.1 5.1.2 5.1. 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat 5.2.1 5.2.2 5.2.3 5.2.4 5.2.4	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies About Creating and Managing Privilege Analysis Policies General Steps for Managing Privilege Analysis Creating a Privilege Analysis Policies Examples of Creating Privilege Analysis Policies	5-1 5-2 5-2 5-2 5-3
Performi 5.1 What 5.1.1 5.1.2 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat 5.2.1 5.2.2 5.2.3 5.2.4 5.2.5 5.2.4 5.2.5 5.2.4	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies About Creating and Managing Privilege Analysis Policies General Steps for Managing Privilege Analysis Creating a Privilege Analysis Policy Examples of Creating Privilege Analysis of Database-Wide Privileges	5-1 5-2 5-2 5-2 5-3 5-3 5-3 5-4 5-4 5-5 5-5 5-5 5-5 5-6 5-8
Performion 5.1 What 5.1.1 5.1.2 5.1.3 5.1.4 5.1.5 5.1.6 5.2 Creat 5.2.1 5.2.2 5.2.3 5.2.4 5.2. 5.2.4 5.2. 5.2.	Is Privilege Analysis to Identify Privilege Use Is Privilege Analysis? About Privilege Analysis Benefits and Use Cases of Privilege Analysis 2.1 Least Privileges Best Practice 2.2 Development of Secure Applications Who Can Perform Privilege Analysis? Types of Privilege Analysis How Does a Multitenant Environment Affect Privilege Analysis? How Privilege Analysis Works with Pre-Compiled Database Objects ing and Managing Privilege Analysis Policies About Creating and Managing Privilege Analysis Policies General Steps for Managing Privilege Analysis Creating a Privilege Analysis Policy Examples of Creating Privilege Analysis Policies 4.1 Example: Privilege Analysis of Privilege Usage of Two Roles	5-1 5-2 5-2 5-3 5-3 5-3 5-4 5-4 5-5 5-5 5-5 5-6 5-8 5-8



ļ	5.2.5	Enab	oling a Privilege Analysis Policy	5-9
ļ	5.2.6	Disal	bling a Privilege Analysis Policy	5-10
į	5.2.7	Gene	erating a Privilege Analysis Report	5-10
	5.2	.7.1	About Generating a Privilege Analysis Report	5-10
	5.2.7.2		General Process for Managing Multiple Named Capture Runs	5-11
	5.2.7.3		Generating a Privilege Analysis Report Using DBMS_PRIVILEGE_CAPTURE	5-12
	5.2	.7.4	Generating a Privilege Analysis Report Using Cloud Control	5-12
	5.2	.7.5	Accessing Privilege Analysis Reports Using Cloud Control	5-13
į	5.2.8	Drop	ping a Privilege Analysis Policy	5-13
5.3	Crea	ting R	oles and Managing Privileges Using Cloud Control	5-14
į	5.3.1	Crea	ting a Role from a Privilege Analysis Report in Cloud Control	5-14
į	5.3.2	Revo	oking and Regranting Roles and Privileges Using Cloud Control	5-15
į	5.3.3	Gene	erating a Revoke or Regrant Script Using Cloud Control	5-15
	5.3	.3.1	About Generating Revoke and Regrant Scripts	5-15
	5.3	.3.2	Generating a Revoke Script	5-16
	5.3	.3.3	Generating a Regrant Script	5-17
5.4	Tutor	ial: Us	sing Capture Runs to Analyze ANY Privilege Use	5-17
ļ	5.4.1	Step	1: Create User Accounts	5-18
į	5.4.2	Step	2: Create and Enable a Privilege Analysis Policy	5-19
į	5.4.3	Step	3: Use the READ ANY TABLE System Privilege	5-19
į	5.4.4	Step	4: Disable the Privilege Analysis Policy	5-20
į	5.4.5	Step	5: Generate and View a Privilege Analysis Report	5-20
į	5.4.6	Step	6: Create a Second Capture Run	5-21
į	5.4.7	Step	7: Remove the Components for This Tutorial	5-22
5.5	Tutor	ial: Ar	nalyzing Privilege Use by a User Who Has the DBA Role	5-22
ļ	5.5.1	Step	1: Create User Accounts	5-23
į	5.5.2	Step	2: Create and Enable a Privilege Analysis Policy	5-23
į	5.5.3	Step	3: Perform the Database Tuning Operations	5-24
į	5.5.4	Step	4: Disable the Privilege Analysis Policy	5-25
į	5.5.5	Step	5: Generate and View Privilege Analysis Reports	5-25
į	5.5.6	Step	6: Remove the Components for This Tutorial	5-26
5.6	Privil	ege A	nalysis Policy and Report Data Dictionary Views	5-27
Co	nfigu	ring	Centrally Managed Users with Microsoft Active Directo	ry
6.1	Intro	ductio	n to Centrally Managed Users with Microsoft Active Directory	6-1
(6.1.1	Abou	at the Oracle Database-Microsoft Active Directory Integration	6-2
(6.1.2		Centrally Managed Users with Microsoft Active Directory Works	6-3
(6.1.3		rally Managed User-Microsoft Active Directory Architecture	6-3
(6.1.4	Supp	ported Authentication Methods	6-4
(6.1.5	User	s Supported by Centrally Managed Users with Microsoft Active Directory	6-4



6

	6.1.6	How	the Oracle Multitenant Option Affects Centrally Managed Users	6-5
	6.1.7	Cent	rally Managed Users with Database Links	6-5
6.2	Conf	igurin	g the Oracle Database-Microsoft Active Directory Integration	6-6
	6.2.1	Abou	at Configuring the Oracle Database-Microsoft Active Directory Connection	6-6
	6.2.2	Conr	necting to Microsoft Active Directory	6-6
	6.2.2.1		Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions	6-7
	6.2	2.2.2	Step 2: For Password Authentication, Install the Password Filter and Extend the Microsoft Active Directory Schema	6-8
	6.2	2.2.3	Step 3: If Necessary, Install the Oracle Database Software	6-10
	6.2	2.2.4	Step 4: Create the dsi.ora or Idap.ora File	6-10
	6.2	2.2.5	Step 5: Request an Active Directory Certificate for a Secure Connection	6-15
	6.2	2.2.6	Step 6: Create the Wallet for a Secure Connection	6-16
	6.2	2.2.7	Step 7: Configure the Microsoft Active Directory Connection	6-17
	6.2	2.2.8	Step 8: Verify the Oracle Wallet	6-21
	6.2	2.2.9	Step 9: Test the Integration	6-22
6.3	Conf	igurin	g Authentication for Centrally Managed Users	6-23
	6.3.1	Conf	iguring Password Authentication for Centrally Managed Users	6-23
	6.3	3.1.1	About Configuring Password Authentication for Centrally Managed Users	6-23
	6.3.1.2		Configuring Password Authentication for a Centrally Managed User	6-24
	6.3	3.1.3	Logging in to an Oracle Database Using Password Authentication	6-25
	6.3.2	Conf	iguring Kerberos Authentication for Centrally Managed Users	6-26
	6.3.3	Conf	iguring Authentication Using PKI Certificates for Centrally Managed Users	6-27
6.4	Conf	igurin	g Authorization for Centrally Managed Users	6-27
	6.4.1	Abou	ut Configuring Authorization for Centrally Managed Users	6-28
	6.4.2	Марі	ping a Directory Group to a Shared Database Global User	6-29
	6.4.3	Марі	ping a Directory Group to a Global Role	6-30
	6.4.4	Excl	usively Mapping a Directory User to a Database Global User	6-30
	6.4.5	Alter	ing or Migrating a User Mapping Definition	6-30
	6.4.6	Conf	iguring Administrative Users	6-31
	6.4	1.6.1	Configuring Database Administrative Users with Shared Access Accounts	6-31
	6.4	1.6.2	Configuring Database Administrative Users Using Exclusive Mapping	6-31
	6.4.7	Verif	ying the Centrally Managed User Logon Information	6-32
6.5	Troul	blesho	poting Centrally Managed Users	6-35
	6.5.1	ORA	-28276 Connection Errors	6-35
	6.5.2	ORA	-01017 Connection Errors	6-36
	6.5.3	ORA	-28274 Connection Errors	6-36
	6.5.4	ORA	-28300 Connection Errors	6-37
	6.5.5	Usin	g Trace Files to Diagnose CMU Connection Errors	6-37
6.6	Integ	ration	of Oracle Database with Microsoft Active Directory Account Policies	6-38
6.7	Conf	igurin	g Centrally Managed Users with Oracle Autonomous Database	6-38
6.8	Troul	blesho	poting Centrally Managed Users	6-39



6.8.1	ORA	A-28276 Connection Errors	6-39
6.8.2	ORA	A-01017 Connection Errors	6-40
6.8.3	ORA	A-28274 Connection Errors	6-40
6.8.4	ORA	A-28300 Connection Errors	6-41
6.8.5	Usir	ng Trace Files to Diagnose CMU Connection Errors	6-41
Auther	nticati	ing and Authorizing IAM Users for Oracle DBaaS Datab	ases
7.1 Intr	roductio	on to Authenticating and Authorizing IAM Users for Oracle DBaaS	7-1
7.1.1	Abo	ut Authenticating and Authorizing IAM Users for Oracle DBaaS	7-2
7.1.2	Arch	nitecture of the IAM Integration with Oracle DBaaS	7-4
7.1.3	IAM	Users and Groups to Map with Oracle DBaaS	7-8
7.2 Co	nfigurin	g Oracle DBaaS for IAM	7-8
7.2.1	Ena	bling External Authentication for Oracle DBaaS	7-8
7.2.2		figuring Authorization for IAM Users and Oracle Cloud Infrastructure lications	7-9
7	7.2.2.1	About Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications	7-10
7	7.2.2.2	Mapping an IAM Group to a Shared Oracle Database Global User	7-11
7	7.2.2.3	Mapping an IAM Group to an Oracle Database Global Role	7-12
7	7.2.2.4	Exclusively Mapping an IAM User to an Oracle Database Global User	7-12
7	7.2.2.5	Altering or Migrating an IAM User Mapping Definition	7-13
7	7.2.2.6	Mapping Instance and Resource Principals	7-13
7	7.2.2.7	Verifying the IAM User Logon Information	7-14
7.2.3	Con	figuring IAM Proxy Authentication	7-17
7	7.2.3.1	About Configuring IAM Proxy Authentication	7-17
7	7.2.3.2	Configuring Proxy Authentication for the IAM User	7-18
7	7.2.3.3	Validating the IAM User Proxy Authentication	7-18
7.3 Co	nfigurin	g IAM for Oracle DBaaS	7-19
7.3.1	Crea	ating an IAM Policy to Authorize Users Authenticating with Tokens	7-19
7.3.2	Crea	ating an IAM Database Password	7-20
7.4 Ac	cessing	the Database Using an Instance Principal or a Resource Principal	7-20
7.5 Co	nfigurin	g the Database Client Connection	7-21
7.5.1	Abo	ut Connecting to an Autonomous Database Instance Using IAM	7-22
7.5.2	Sup	ported Client Drivers for IAM Connections	7-22
7.5.3	Usir Sec	ng Centralized Oracle Cloud Infrastructure Services for Net Naming and rets	7-22
7.5.4	Clie	nt Connections That Use an IAM Database Password Verifier	7-23
7.5.5		nt Connections That Use a Token Requested by an IAM User Name and abase Password	7-23
7	7.5.5.1	About Client Connections That Use a Token Requested by an IAM User Name and Database Password	7-23
7	7.5.5.2	Parameters to Set for Client Connections That Use a Token Requested by an IAM User Name and Database Password	7-24



Client Application or Tool at Uses an IAM at Uses an IAM Token on Instances	7-27 7-28 7-28 7-28 7-29 7-31 7-31 7-32 7-33
at Uses an IAM at Uses an IAM Token on Instances	7-28 7-28 7-28 7-29 7-31 7-31 7-32
at Uses an IAM Token on Instances	7-28 7-28 7-29 7-31 7-31 7-32
at Uses an IAM Token on Instances	7-28 7-29 7-31 7-31 7-32
at Uses an IAM Token on Instances	7-29 7-31 7-31 7-32
on Instances	7-31 7-31 7-32
Instances	7-31 7-32
	7-32
ICV	
ICV	7-33
)CV	
.09	7-33
	7-34
groups in Another	7-35
S	7-36
mand-Line Interface	7-36
	7-36
	7-37
'S	7-37
	7-39
racle Database for	7-40
AM Users	7-41
ng to Access a Database	7-42
01017 Errors	7-42
r	Froups in Another Is sumand-Line Interface Irs Pracle Database for Samuel Am Users Ing to Access a Database O1017 Errors

7.5.5.3 Configuring the Database Client to Retrieve a Token Using an IAM User



	8.2.3	3	Enab	ling Microsoft Azure AD v2 Access Tokens	8-12
	8.2.4	1	Mana	nging App Roles in Microsoft Azure AD	8-12
	8	8.2.4	4.1	Creating a Microsoft Azure AD App Role	8-13
	8	8.2.4	4.2	Assigning Users and Groups to the Microsoft Azure AD App Role	8-14
	8	8.2.4	4.3	Assigning an Application to an App Role	8-14
	8.2.5	5	Enab	ling Azure AD External Authentication for Oracle Database	8-15
	8.2.6	6	Disab	oling Azure AD External Authentication for Oracle Database	8-16
8.3	Ma	appi	ng O	racle Database Schemas and Roles	8-17
	8.3.1	L I	Exclu	sively Mapping an Oracle Database Schema to a Microsoft Azure AD User	8-17
	8.3.2	2	Марр	ing a Shared Oracle Schema to an App Role	8-17
	8.3.3	3	Марр	ing an Oracle Database Global Role to an App Role	8-18
8.4	Co	onfig	juring	Azure AD Client Connections to the Oracle Database	8-18
	8.4.1	L ,	Abou	t Configuring Client Connections to Azure ADs	8-19
	8.4.2	2	Supp	orted Client Drivers for Azure AD Connections	8-19
	8.4.3			Centralized Oracle Cloud Infrastructure Services for Net Naming and	
			Secre		8-19
	8.4.4		Opera Datal	ational Flow for SQL*Plus Client Connection in PowerShell to Oracle	8-20
	8.4.5			stering a Client with Azure AD Application Registration	8-20
		8.4.	-	Confidential and Public Client Registration	8-21
	8	3.4.	5.2	Registering a Database Client App with Azure AD	8-21
	8.4.6	6	Exam	nples of Retrieving Azure AD OAuth2 Tokens	8-23
	8	8.4.6		Example: Using PowerShell to Get a Token Using Resource Owner	
				Password Credentials	8-23
	8	8.4.0	5.2	Example: Using Python with Microsoft Authentication Library Using an Authorization Flow	8-25
	8	8.4.6	6.3	Example: Using Curl with a Resource Owner Password Credential Flow	8-25
	8	8.4.6	6.4	Example: Azure CLI Using Authorization Flow	8-26
	8.4.7	7	Confi	guring SQL*Plus for Azure AD Access Tokens	8-26
	8.4.8	3	Creat	ing a Network Proxy for the Database to Connect with the Internet	8-28
	8	8.4.8	3.1	About Creating a Network Proxy for the Database to Connect with the	0.00
				Internet	8-28
		8.4.8		Testing the Accessibility of the Azure Endpoint	8-28
		8.4.8		Creating the Network Proxy for the Default Oracle Database Environment	8-30
	ì	8.4.8	3.4	Creating the Network Proxy for an Oracle Real Application Clusters Environment	8-30
	8	8.4.8	3.5	Creating the Network Proxy in the Windows Registry Editor	8-31
	8.4.9)	Enab	ling Clients to Directly Retrieve Azure Tokens	8-32
8.5	Co	onfig	juring	Microsoft Azure AD Proxy Authentication	8-33
	8.5.1	L ,	Abou	t Configuring Microsoft Azure AD Proxy Authentication	8-34
	8.5.2			guring Proxy Authentication for the Azure AD User	8-34
	8.5.3			ating the Azure AD User Proxy Authentication	8-34
8.6	Tro	oubl	esho	oting Microsoft Azure AD Connections	8-35



	8.6.1		e Files for Troubleshooting Oracle Database Client Connections with Azure	0.05
	0.4	AD	About Trace Files Head for Traublesheating Compations	8-35
		6.1.1	About Trace Files Used for Troubleshooting Connections	8-35
		6.1.2	Setting Client Tracing for Token Authentication	8-36
	8.6.2		-12599 and ORA-03114 Errors Caused When Trying to Access a Database g a Token	8-37
	8.6.3		cking the Azure AD Access Token Version	8-37
9	Managi	ng S	ecurity for Definer's Rights and Invoker's Rights	
	9.1 Abo	ut Defi	ner's Rights and Invoker's Rights	9-1
	9.2 How	/ Proce	dure Privileges Affect Definer's Rights	9-2
	9.3 How	/ Proce	dure Privileges Affect Invoker's Rights	9-3
	9.4 Whe	en You	Should Create Invoker's Rights Procedures	9-3
	9.5 Con	trolling	Invoker's Rights Privileges for Procedure Calls and View Access	9-4
	9.5.1	How	the Privileges of a Schema Affect the Use of Invoker's Rights Procedures	9-5
	9.5.2	How	the INHERIT [ANY] PRIVILEGES Privileges Control Privilege Access	9-6
	9.5.3	Gran	ts of the INHERIT PRIVILEGES Privilege to Other Users	9-6
	9.5.4	Exan	nple: Granting INHERIT PRIVILEGES on an Invoking User	9-6
	9.5.5	Exan	nple: Revoking INHERIT PRIVILEGES	9-7
	9.5.6	Gran	ts of the INHERIT ANY PRIVILEGES Privilege to Other Users	9-7
	9.5.7	Exan	nple: Granting INHERIT ANY PRIVILEGES to a Trusted Procedure Owner	9-7
	9.5.8	Mana	aging INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES	9-7
	9.6 Defi	ner's R	tights and Invoker's Rights in Views	9-8
	9.6.1	Abou	at Controlling Definer's Rights and Invoker's Rights in Views	9-8
	9.6.2	Usin	g the BEQUEATH Clause in the CREATE VIEW Statement	9-9
	9.6.3	Findi	ng the User Name or User ID of the Invoking User	9-10
	9.6.4	Findi	ng BEQUEATH DEFINER and BEQUEATH_CURRENT_USER Views	9-10
	9.7 Usin	g Cod	e Based Access Control for Definer's Rights and Invoker's Rights	9-11
	9.7.1	Abou	It Using Code Based Access Control for Applications	9-11
	9.7.2	Who	Can Grant Code Based Access Control Roles to a Program Unit?	9-12
	9.7.3	How	Code Based Access Control Works with Invoker's Rights Program Units	9-12
	9.7.4	How	Code Based Access Control Works with Definer's Rights Program Units	9-14
	9.7.5	Gran	ts of Database Roles to Users for Their CBAC Grants	9-15
	9.7.6	Gran	ts and Revokes of Database Roles to a Program Unit	9-16
	9.7.7	Tuto	rial: Controlling Access to Sensitive Data Using Code Based Access Control	9-17
	9.	7.7.1	About This Tutorial	9-17
	9.	7.7.2	Step 1: Create the User and Grant HR the CREATE ROLE Privilege	9-18
	9.	7.7.3	Step 2: Create the print_employees Invoker's Rights Procedure	9-18
	9.	7.7.4	Step 3: Create the hr_clerk Role and Grant Privileges for It	9-19
	9.	7.7.5	Step 4: Test the Code Based Access Control HR.print_employees Procedure	9-19
	9.	7.7.6	Step 5: Create the view_emp_role Role and Grant Privileges for It	9-20



		9.7.7	.7	Step 6: Test the HR.print_employees Procedure Again	9-20
		9.7.7	.8	Step 7: Remove the Components of This Tutorial	9-21
	9.8	Contro	lling	Definer's Rights Privileges for Database Links	9-21
	9.8	8.1 A	bout	Controlling Definer's Rights Privileges for Database Links	9-22
	9.8	8.2	rant	s of the INHERIT REMOTE PRIVILEGES Privilege to Other Users	9-23
	9.8	3.3 E	xam	ple: Granting INHERIT REMOTE PRIVILEGES on a Connected User	9-23
	9.8	8.4 G	Frant	s of the INHERIT ANY REMOTE PRIVILEGES Privilege to Other Users	9-24
	9.8	8.5 F	Revol	kes of the INHERIT [ANY] REMOTE PRIVILEGES Privilege	9-24
	9.8	8.6 E	xam	ple: Revoking the INHERIT REMOTE PRIVILEGES Privilege	9-25
	9.8	8.7 E	xam	ple: Revoking the INHERIT REMOTE PRIVILEGES Privilege from PUBLIC	9-25
	9.8	8.8 T	utori	al: Using a Database Link in a Definer's Rights Procedure	9-25
		9.8.8	.1	About This Tutorial	9-26
		9.8.8	.2	Step 1: Create User Accounts	9-26
		9.8.8	.3	Step 2: As User dbuser2, Create a Table to Store User IDs	9-26
		9.8.8	.4	Step 3: As User dbuser1, Create a Database Link and Definer's Rights	0.00
		0.0.0	_	Procedure	9-26
		9.8.8		Step 4: Test the Definer's Rights Procedure	9-27
		9.8.8	.6	Step 5: Remove the Components of This Tutorial	9-28
10	10.1			ne-Grained Access in PL/SQL Packages and Types haging Fine-Grained Access in PL/SQL Packages and Types	10-2
	10.2			e-Grained Access Control to External Network Services	10-2
	10.3	About	Acc	ess Control to Oracle Wallets	10-3
	10.4	Upgra Servic		Applications That Depend on Packages That Use External Network	10-3
	10.5	Config	gurin	g Access Control for External Network Services	10-4
	10	.5.1	Synt	ax for Configuring Access Control for External Network Services	10-4
	10	.5.2	Enal	oling the Listener to Recognize Access Control for External Network	
			Serv	ices	10-6
	10	.5.3	Exar	mple: Configuring Access Control for External Network Services	10-6
				oking Access Control Privileges for External Network Services	10-6
	10			mple: Revoking External Network Services Privileges	10-7
	10.6	`		g Access Control to an Oracle Wallet	10-7
				ut Configuring Access Control to an Oracle Wallet	10-8
				1: Create an Oracle Wallet	10-8
				2: Configure Access Control Privileges for the Oracle Wallet	10-8
	10			3: Make the HTTP Request with the Passwords and Client Certificates	10-9
		10.6.		Making the HTTPS Request with the Passwords and Client Certificates	10-10
		10.6.	4.2	Using a Request Context to Hold the Wallet When Sharing the Session with Other Applications	10-11
		10.6.	12	Use of Only a Client Certificate to Authenticate	10-11
		10.6.		Use of a Password to Authenticate	10-11
		10.0.	4.4	OSE OF A F ASSWORD TO AUDITIONALE	10-11



	10.6	6.5	Revoking Access Control Privileges for Oracle Wallets	10-12
	10.6	6.6	Troubleshooting ORA-29024 Errors	10-13
	10.7 E	Exan	nples of Configuring Access Control for External Network Services	10-13
	10.7	'.1	Example: Configuring Access Control for a Single Role and Network	
			Connection	10-14
	10.7		Example: Configuring Access Control for a User and Role	10-14
	10.7		Example: Using the DBA_HOST_ACES View to Show Granted Privileges	10-15
	10.7		Example: Configuring ACL Access Using Passwords in a Non-Shared Wallet	10-15
	10.7		Example: Configuring ACL Access for a Wallet in a Shared Database Session	10-16
			ifying a Group of Network Host Computers	10-17
			edence Order for a Host Computer in Multiple Access Control List Assignments	10-17
			cedence Order for a Host in Access Control List Assignments with Port Ranges	10-18
	10.11	Che	cking Privilege Assignments That Affect User Access to Network Hosts	10-19
	10.1	1.1	About Privilege Assignments that Affect User Access to Network Hosts	10-19
	10.1	1.2	How to Check User Network Connection and Domain Privileges	10-20
	10.1	1.3	Example: Administrator Checking User Network Access Control Permissions	10-20
	10.1	1.4	How Users Can Check Their Network Connection and Domain Privileges	10-21
	10.1	1.5	Example: User Checking Network Access Control Permissions	10-21
	10.12	Con	figuring Network Access for Java Debug Wire Protocol Operations	10-21
	10.13	Data	a Dictionary Views for Access Control Lists Configured for User Access	10-22
	11.1 A	Abou	t Managing Security for a Multitenant Environment in Enterprise Manager	11-1
	11.2 L	.oggi	ing into a Multitenant Environment in Enterprise Manager	11-1
	11.2	.1	Logging into a CDB or a PDB	11-2
	11.2	.2	Switching to a Different PDB or to the Root	11-3
	11.3 N	/lana	iging Common and Local Users in Enterprise Manager	11-3
	11.3	3.1	Creating a Common User Account in Enterprise Manager	11-4
	11.3	.2	Editing a Common User Account in Enterprise Manager	11-4
	11.3	3.3	Dropping a Common User Account in Enterprise Manager	11-5
	11.3	.4	Creating a Local User Account in Enterprise Manager	11-6
	11.3	.5	Editing a Local User Account in Enterprise Manager	11-6
	11.3	.6	Dropping a Local User Account in Enterprise Manager	11-7
	11.4 N	/lana	iging Common and Local Roles and Privileges in Enterprise Manager	11-7
	11.4	.1	Creating a Common Role in Enterprise Manager	11-8
	11.4	.2	Editing a Common Role in Enterprise Manager	11-8
	11.4	.3	Dropping a Common Role in Enterprise Manager	11-9
	11.4	.4	Revoking Common Privilege Grants in Enterprise Manager	11-9
	11.4	.5	Creating a Local Role in Enterprise Manager	11-10
	11.4	.6	Editing a Local Role in Enterprise Manager	11-10
	11.4	.7	Dropping a Local Role in Enterprise Manager	11-11



Part II Application Development Security

12 Managing Security for Application Developers

12.	1	Abou	t Appli	ication Security Policies	12-1
12.	2	Cons	iderati	ions for Using Application-Based Security	12-2
	12.	.2.1	Are A	application Users Also Database Users?	12-2
	12.	.2.2	Is Se	curity Better Enforced in the Application or in the Database?	12-3
12.	3	Secu	ring Pa	asswords in Application Design	12-3
	12.	.3.1	Gene	ral Guidelines for Securing Passwords in Applications	12-4
		12.3	1.1	Platform-Specific Security Threats	12-4
		12.3	1.2	Guidelines for Designing Applications to Handle Password Input	12-4
		12.3	1.3	Guidelines for Configuring Password Formats and Behavior	12-5
		12.3	1.4	Guidelines for Handling Passwords in SQL Scripts	12-6
	12	.3.2	Use o	of an External Password Store to Secure Passwords	12-7
	12	.3.3	Secu	ring Passwords Using the ORAPWD Utility	12-8
	12	.3.4	Exam	nple: Java Code for Reading Passwords	12-8
12.	4	Secu	ring E	xternal Procedures	12-12
	12	.4.1	Abou	t Securing External Procedures	12-12
	12	.4.2	Gene	ral Process for Configuring extproc for a Credential Authentication	12-13
	12	.4.3	extpr	oc Process Authentication and Impersonation Expected Behaviors	12-13
	12	.4.4	Confi	guring Authentication for External Procedures	12-14
	12	.4.5	Exter	nal Procedures for Legacy Applications	12-16
12.	5	Secu	ring Lo	OBs with LOB Locator Signatures	12-16
	12	.5.1	Abou	t Securing LOBs with LOB Locator Signatures	12-16
	12	.5.2	Mana	iging the Encryption of a LOB Locator Signature Key	12-17
12.	6	Mana	ging A	Application Privileges	12-18
12.	7	Adva	ntages	s of Using Roles to Manage Application Privileges	12-18
12.	8	Creat	ing Se	ecure Application Roles to Control Access to Applications	12-19
	12	.8.1	Step	1: Create the Secure Application Role	12-19
	12.	.8.2		2: Create a PL/SQL Package to Define the Access Policy for the cation	12-20
		12.8	.2.1	About Creating a PL/SQL Package to Define the Access Policy for an Application	12-20
		12.8	.2.2	Creating a PL/SQL Package or Procedure to Define the Access Policy for an Application	12-21
		12.8	.2.3	Testing the Secure Application Role	12-22
12.	9	Asso	ciation	of Privileges with User Database Roles	12-22
	12.	.9.1	Why	Users Should Only Have the Privileges of the Current Database Role	12-22
	12.	.9.2	Use o	of the SET ROLE Statement to Automatically Enable or Disable Roles	12-23



		12.10.2	Pro	tection of Database Objects in a Shared Schema	12-24
	12	.11 Ob	ject Pr	ivileges in an Application	12-24
		12.11.1	Wh	at Application Developers Must Know About Object Privileges	12-24
		12.11.2	SQ	L Statements Permitted by Object Privileges	12-25
	12	.12 Pa	ramete	ers for Enhanced Security of Database Communication	12-26
		12.12.1	Bac	d Packets Received on the Database from Protocol Errors	12-26
		12.12.2	Cor	ntrolling Server Execution After Receiving a Bad Packet	12-27
		12.12.3	Cor	nfiguration of the Maximum Number of Authentication Attempts	12-28
		12.12.4	Cor	nfiguring the Display of the Database Version Banner	12-28
		12.12.5	Cor	nfiguring Banners for Unauthorized Access and Auditing User Actions	12-29
Part		Con	itrolli	ng Access to Data	
L3	U:	sing A	pplica	ation Contexts to Retrieve User Information	
	13	.1 Abo	ut App	lication Contexts	13-1
		13.1.1	Wha	t Is an Application Context?	13-2
		13.1.2	Com	ponents of the Application Context	13-2
		13.1.3	Whe	re Are the Application Context Values Stored?	13-2
		13.1.4	Bene	efits of Using Application Contexts	13-3
		13.1.5	How	Editions Affects Application Context Values	13-3
		13.1.6	Appl	ication Contexts in a Multitenant Environment	13-3
	13	.2 Type	es of A	pplication Contexts	13-4
	13	.3 Usir	ng Data	abase Session-Based Application Contexts	13-5
		13.3.1	Abou	ut Database Session-Based Application Contexts	13-6
		13.3.2	Com	ponents of a Database Session-Based Application Context	13-7
		13.3.3	Crea	ating Database Session-Based Application Contexts	13-7
		13.	3.3.1	About Creating Database Session-Based Application Contexts	13-8
		13.	3.3.2	Creating a Database Session-Based Application Context	13-8
		13.	3.3.3	Database Session-Based Application Contexts for Multiple Applications	13-9
		13.3.4	Crea	ating a Package to Set a Database Session-Based Application Context	13-9
		13.	3.4.1	About the Package That Manages the Database Session-Based Application Context	13-10
		13.	3.4.2	Using the SYS_CONTEXT Function to Retrieve Session Information	13-11
		13.	3.4.3	Checking the SYS_CONTEXT Settings	13-12
		13.	3.4.4	Dynamic SQL with SYS_CONTEXT	13-12
		13.	3.4.5	SYS_CONTEXT in a Parallel Query	13-12
		13.	3.4.6	SYS_CONTEXT with Database Links	13-13
			3.4.7	DBMS_SESSION.SET_CONTEXT for Setting Session Information	13-13
			3.4.8	Example: Simple Procedure to Create an Application Context Value	13-14

12.10 Protecting Database Objects by Using Schemas

12.10.1 Protecting Database Objects in a Unique Schema



12-23

12-23

	13.3.5	Logo	on Triggers to Run a Database Session Application Context Package	13-15
	13.3.6	Exan	nple: Creating a Simple Logon Trigger	13-16
	13.3.7	Exan	nple: Creating a Logon Trigger for a Production Environment	13-16
	13.3.8	Exan	nple: Creating a Logon Trigger for a Development Environment	13-16
	13.3.9	Tuto	rial: Creating and Using a Database Session-Based Application Context	13-17
	13.3	3.9.1	Step 1: Create User Accounts and Ensure the User SCOTT Is Active	13-17
	13.3	3.9.2	Step 2: Create the Database Session-Based Application Context	13-18
	13.3	3.9.3	Step 3: Create a Package to Retrieve Session Data and Set the Application Context	13-18
	13.3	3.9.4	Step 4: Create a Logon Trigger for the Package	13-20
	13.3	3.9.5	Step 5: Test the Application Context	13-20
	13.3	3.9.6	Step 6: Remove the Components of This Tutorial	13-21
	13.3.10	Initi	alizing Database Session-Based Application Contexts Externally	13-21
	13.3	3.10.1	About Initializing Database Session-Based Application Contexts Externally	13-21
	13.3	3.10.2	Default Values from Users	13-22
	13.3	3.10.3	Values from Other External Resources	13-22
	13.3	3.10.4	Example: Creating an Externalized Database Session-based	
			Application Context	13-22
		3.10.5	Initialization of Application Context Values from a Middle-Tier Server	13-23
	13.3.11		alizing Database Session-Based Application Contexts Globally	13-23
	13.3	3.11.1	About Initializing Database Session-Based Application Contexts Globally	13-23
	13.3	3.11.2	Database Session-Based Application Contexts with LDAP	13-24
	13.3	3.11.3	How Globally Initialized Database Session-Based Application Contexts Work	13-25
	13.3	3.11.4	Initializing a Database Session-Based Application Context Globally	13-26
	13.3.12	Ext	ernalized Database Session-Based Application Contexts	13-27
13.	4 Glob	al App	olication Contexts	13-28
	13.4.1	Abou	ut Global Application Contexts	13-28
	13.4.2	Uses	for Global Application Contexts	13-29
	13.4.3	Com	ponents of a Global Application Context	13-29
	13.4.4		al Application Contexts in an Oracle Real Application Clusters ronment	13-30
	13.4.5	Crea	ting Global Application Contexts	13-30
	13.4	1.5.1	Ownership of the Global Application Context	13-30
	13.4	1.5.2	Creating a Global Application Context	13-31
	13.4.6	PL/S	QL Package to Manage a Global Application Context	13-31
	13.4	1.6.1	About the Package That Manages the Global Application Context	13-32
	13.4	1.6.2	How Editions Affects the Results of a Global Application Context PL/SQL	12.22
	10	160	Package	13-32
		1.6.3	DBMS_SESSION.SET_CONTEXT username and client_id Parameters	13-33
	13.4	1.6.4	Sharing Global Application Context Values for All Database Users	13-34



		Users	13-34
	13.4.6.6	Global Contexts for Database Users Who Move Between Applications	13-35
	13.4.6.7	Global Application Context for Nondatabase Users	13-37
	13.4.6.8	Example: Package to Manage Global Application Context Values for Nondatabase Users	13-37
	13.4.6.9	Clearing Session Data When the Session Closes	13-39
	13.4.7 Emb	pedding Calls in Middle-Tier Applications to Manage the Client Session ID	13-40
	13.4.7.1	About Managing Client Session IDs Using a Middle-Tier Application	13-40
	13.4.7.2	Step 1: Retrieve the Client Session ID Using a Middle-Tier Application	13-40
	13.4.7.3	Step 2: Set the Client Session ID Using a Middle-Tier Application	13-41
	13.4.7.4	Step 3: Clear the Session Data Using a Middle-Tier Application	13-43
	13.4.8 Tuto	orial: Creating a Global Application Context That Uses a Client Session ID	13-43
	13.4.8.1	About This Tutorial	13-44
	13.4.8.2	Step 1: Create User Accounts	13-44
	13.4.8.3	Step 2: Create the Global Application Context	13-44
	13.4.8.4	Step 3: Create a Package for the Global Application Context	13-45
	13.4.8.5	Step 4: Test the Newly Created Global Application Context	13-46
	13.4.8.6	Step 5: Modify the Session ID and Test the Global Application Context Again	13-47
	13.4.8.7	Step 6: Remove the Components of This Tutorial	13-48
	13.4.9 Glob	pal Application Context Processes	13-48
	13.4.9.1	Simple Global Application Context Process	13-48
	13.4.9.2	Global Application Context Process for Lightweight Users	13-49
	13.5 Using Clie	ent Session-Based Application Contexts	13-51
	13.5.1 Abo	ut Client Session-Based Application Contexts	13-52
	13.5.2 Sett	ing a Value in the CLIENTCONTEXT Namespace	13-53
	13.5.3 Retr	rieving the CLIENTCONTEXT Namespace	13-53
		mple: Retrieving a Client Session ID Value for Client Session-Based texts	13-53
		aring a Setting in the CLIENTCONTEXT Namespace	13-54
		aring All Settings in the CLIENTCONTEXT Namespace	13-54
		n Context Data Dictionary Views	13-55
14	Using Oracle	e Virtual Private Database to Control Data Access	
	14.1 About Ora	ucle Virtual Private Database	14-1
	14.1.1 Wha	at Is Oracle Virtual Private Database?	14-2
	14.1.2 Ben	efits of Using Oracle Virtual Private Database Policies	14-3
	14.1.2.1	Security Policies Based on Database Objects Rather Than Applications	14-3
	14.1.2.2	Control Over How Oracle Database Evaluates Policy Functions	14-3
	14.1.3 Who	Can Create Oracle Virtual Private Database Policies?	14-4
	14.1.4 Priv	ileges to Run Oracle Virtual Private Database Policy Functions	14-4

13.4.6.5 Example: Package to Manage Global Application Values for All Database



14.1.5	5 Orac	le Virtual Private Database Use with an Application Context	14-4
14.1.6	6 Orac	le Virtual Private Database in a Multitenant Environment	14-5
14.2 Co	mponer	nts of an Oracle Virtual Private Database Policy	14-6
14.2.1	L Fund	ction to Generate the Dynamic WHERE Clause	14-6
14.2.2	2 Polic	ies to Attach the Function to the Objects You Want to Protect	14-8
14.3 Co	onfigurat	ion of Oracle Virtual Private Database Policies	14-8
14.3.1	L Abou	ut Oracle Virtual Private Database Policies	14-9
14.3.2	2 Attac	ching a Policy to a Database Table, View, or Synonym	14-10
14.3.3	B Exar	nple: Attaching a Simple Oracle Virtual Private Database Policy to a Table	14-10
14.3.4	1 Enfo	rcing Policies on Specific SQL Statement Types	14-11
14.3.5	5 Exar	nple: Specifying SQL Statement Types with DBMS_RLS.ADD_POLICY	14-11
14.3.6	6 Cont	rol of the Display of Column Data with Policies	14-12
14	4.3.6.1	Policies for Column-Level Oracle Virtual Private Database	14-12
14	4.3.6.2	Example: Creating a Column-Level Oracle Virtual Private Database Policy	14-13
14	4.3.6.3	Display of Only the Column Rows Relevant to the Query	14-13
14	4.3.6.4	Column Masking to Display Sensitive Columns as NULL Values	14-14
14	4.3.6.5	Example: Adding Column Masking to an Oracle Virtual Private Database	
		Policy	14-15
14.3.7	7 Orac	ele Virtual Private Database Policy Groups	14-15
14	4.3.7.1	About Oracle Virtual Private Database Policy Groups	14-16
14	4.3.7.2	Creation of a New Oracle Virtual Private Database Policy Group	14-16
	4.3.7.3	Default Policy Group with the SYS_DEFAULT Policy Group	14-17
	4.3.7.4	Multiple Policies for Each Table, View, or Synonym	14-17
	4.3.7.5	Validation of the Application Used to Connect to the Database	14-18
14.3.8	3 Optir	mizing Performance by Using Oracle Virtual Private Database Policy Types	14-18
14	4.3.8.1	About Oracle Virtual Private Database Policy Types	14-19
	4.3.8.2	Dynamic Policy Type to Automatically Rerun Policy Functions	14-20
14	4.3.8.3	Example: Creating a DYNAMIC Policy with DBMS_RLS.ADD_POLICY	14-20
14	4.3.8.4	Static Policy to Prevent Policy Functions from Rerunning for Each Query	14-21
14	4.3.8.5	Example: Creating a Static Policy with DBMS_RLS.ADD_POLICY	14-21
14	4.3.8.6	Example: Shared Static Policy to Share a Policy with Multiple Objects	14-22
14	4.3.8.7	When to Use Static and Shared Static Policies	14-22
14	4.3.8.8	Context-Sensitive Policy for Application Context Attributes That Change	14-23
14	4.3.8.9	Example: Creating a Context-Sensitive Policy with DBMS_RLS.ADD_POLICY	14-24
14	4.3.8.10	Example: Refreshing Cached Statements for a VPD Context-Sensitive Policy	14-24
14	4.3.8.11	Example: Altering an Existing Context-Sensitive Policy	14-24
14	4.3.8.12	Example: Using a Shared Context Sensitive Policy to Share a Policy with Multiple Objects	14-25
14	4.3.8.13	When to Use Context-Sensitive and Shared Context-Sensitive Policies	14-25
14	4.3.8.14	Summary of the Five Oracle Virtual Private Database Policy Types	14-26



14.4	Tuto	rials: C	Creating Oracle Virtual Private Database Policies	14-26
	14.4.1	Tuto	rial: Creating a Simple Oracle Virtual Private Database Policy	14-27
	14.4	1.1.1	About This Tutorial	14-27
	14.4	1.1.2	Step 1: Ensure That the OE User Account Is Active	14-27
	14.4	4.1.3	Step 2: Create a Policy Function	14-28
	14.4	1.1.4	Step 3: Create the Oracle Virtual Private Database Policy	14-29
	14.4	4.1.5	Step 4: Test the Policy	14-29
	14.4	1.1.6	Step 5: Remove the Components of This Tutorial	14-30
	14.4.2	Tuto	rial: Implementing a Session-Based Application Context Policy	14-30
	14.4	1.2.1	About This Tutorial	14-31
	14.4	1.2.2	Step 1: Create User Accounts and Sample Tables	14-31
	14.4	1.2.3	Step 2: Create a Database Session-Based Application Context	14-33
	14.4	1.2.4	Step 3: Create a PL/SQL Package to Set the Application Context	14-33
	14.4	1.2.5	Step 4: Create a Logon Trigger to Run the Application Context PL/SQL Package	14-34
	14.4	1.2.6	Step 5: Test the Logon Trigger	14-34
	14.4	1.2.7	Step 6: Create a PL/SQL Policy Function to Limit User Access to Their Orders	14-35
	14.4	1.2.8	Step 7: Create the New Security Policy	14-35
	14.4	1.2.9	Step 8: Test the New Policy	14-36
	14.4	1.2.10	Step 9: Remove the Components of This Tutorial	14-37
	14.4.3	Tuto	rial: Implementing an Oracle Virtual Private Database Policy Group	14-37
	14.4	1.3.1	About This Tutorial	14-38
	14.4	1.3.2	Step 1: Create User Accounts and Other Components for This Tutorial	14-38
	14.4	1.3.3	Step 2: Create the Two Policy Groups	14-39
	14.4	1.3.4	Step 3: Create PL/SQL Functions to Control the Policy Groups	14-40
	14.4	1.3.5	Step 4: Create the Driving Application Context	14-41
	14.4	1.3.6	Step 5: Add the PL/SQL Functions to the Policy Groups	14-41
	14.4	1.3.7	Step 6: Test the Policy Groups	14-42
	14.4	1.3.8	Step 7: Remove the Components of This Tutorial	14-43
14.5	How	Oracl	e Virtual Private Database Works with Other Oracle Features	14-44
	14.5.1	Orac	le Virtual Private Database Policies with Editions	14-44
	14.5.2	SELE	ECT FOR UPDATE Statement in User Queries on VPD-Protected Tables	14-45
	14.5.3	Orac	le Virtual Private Database Policies and Outer or ANSI Joins	14-45
	14.5.4	Orac	le Virtual Private Database Security Policies and Applications	14-45
	14.5.5	Auto	matic Reparsing for Fine-Grained Access Control Policies Functions	14-46
	14.5.6	Orac	le Virtual Private Database Policies and Flashback Queries	14-46
	14.5.7	Orac	le Virtual Private Database and Oracle Label Security	14-47
	14.5	5.7.1	Using Oracle Virtual Private Database to Enforce Oracle Label Security Policies	14-47
	14.5	5.7.2	Oracle Virtual Private Database and Oracle Label Security Exceptions	14-47
	14.5.8	Expo	ort of Data Using the EXPDP Utility access_method Parameter	14-49
	14.5.9	User	Models and Oracle Virtual Private Database	14-49



15 Using Transparent Sensitive Data Protection

15.1	About	t Trans	sparent Sensitive Data Protection	15-2
15.2	15.2 General Ste		eps for Using Transparent Sensitive Data Protection	15-2
15.3	Use C	Cases	for Transparent Sensitive Data Protection Policies	15-3
15.4	Privile	eges F	Required for Using Transparent Sensitive Data Protection	15-4
15.5	5.5 How a Multi		itenant Environment Affects Transparent Sensitive Data Protection	15-4
15.6	Creat	ing Tr	ansparent Sensitive Data Protection Policies	15-5
15	5.6.1	Step	1: Create a Sensitive Type	15-5
15	5.6.2	Step	2: Identify the Sensitive Columns to Protect	15-6
15	15.6.3 Step		3: Import the Sensitive Columns List from ADM into Your Database	15-7
15	5.6.4	Step	4: Create the Transparent Sensitive Data Protection Policy	15-7
	15.6.	4.1	About Creating the Transparent Sensitive Data Protection Policy	15-8
	15.6.	4.2	Creating the Transparent Sensitive Data Protection Policy	15-8
	15.6.	4.3	Setting the Oracle Data Redaction or Virtual Private Database Feature Options	15-9
	15.6.	4.4	Setting Conditions for the Transparent Sensitive Data Protection Policy	15-10
	15.6.	4.5	Specifying the DBMS_TSDP_PROTECT.ADD_POLICY Procedure	15-11
15	5.6.5	Step	5: Associate the Policy with a Sensitive Type	15-11
15	5.6.6	Step	6: Enable the Transparent Sensitive Data Protection Policy	15-12
	15.6.	6.1	Enabling Protection for the Current Database in a Protected Source	15-12
	15.6.	6.2	Enabling Protection for a Specific Table Column	15-12
	15.6.	6.3	Enabling Protection for a Specific Column Type	15-13
15	5.6.7	Step	7: Optionally, Export the Policy to Other Databases	15-13
15.7	Alterir	ng Tra	Insparent Sensitive Data Protection Policies	15-14
15.8	Disab	ling T	ransparent Sensitive Data Protection Policies	15-14
15.9	Dropp	ing T	ransparent Sensitive Data Protection Policies	15-15
15.10	Usin	g the	Predefined REDACT_AUDIT Policy to Mask Bind Values	15-17
15	5.10.1	Abo	ut the REDACT_AUDIT Policy	15-17
15	5.10.2	Varia	ables Associated with Sensitive Columns	15-18
	15.10	0.2.1	About Variables Associated with Sensitive Columns	15-18
	15.10	0.2.2	Bind Variables and Sensitive Columns in the Expressions of Conditions	15-18
	15.10	0.2.3	A Bind Variable and a Sensitive Column Appearing in the Same SELECT Item	15-19
	15.10	0.2.4	Bind Variables in Expressions Assigned to Sensitive Columns in INSERT or UPDATE Operations	15-20
15	15.10.3 H		Bind Variables on Sensitive Columns Behave with Views	15-20
15	5.10.4	Disa	abling the REDACT_AUDIT Policy	15-20
15	5.10.5	Ena	bling the REDACT_AUDIT Policy	15-21
15.11	Tran	spare	nt Sensitive Data Protection Policies with Data Redaction	15-21
15.12	Usin	g Trai	nsparent Sensitive Data Protection Policies with Oracle VPD Policies	15-22



	15.12.1	About Using TSDP Policies with Oracle Virtual Private Database Policies	15-22
	15.12.2	DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies	15-23
	15.12.3	Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection	15-24
	15.12		15-25
	15.12		15-25
	15.12	·	15-26
	15.12	·	15-26
	15.12		15-27
	15.12		15-28
	15.13 Usin	g Transparent Sensitive Data Protection Policies with Unified Auditing	15-28
	15.13.1	About Using TSDP Policies with Unified Audit Policies	15-29
	15.13.2	Unified Audit Policy Settings That Are Used with TSDP Policies	15-30
	15.14 Usin	g Transparent Sensitive Data Protection Policies with Fine-Grained Auditing	15-31
	15.14.1	About Using TSDP Policies with Fine-Grained Auditing	15-31
	15.14.2	Fine-Grained Auditing Parameters That Are Used with TSDP Policies	15-31
	15.15 Usin	g Transparent Sensitive Data Protection Policies with TDE Column Encryption	15-32
	15.15.1	About Using TSDP Policies with TDE Column Encryption	15-33
	15.15.2	TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies	15-34
	15.16 Tran	sparent Sensitive Data Protection Data Dictionary Views	15-34
16		n of Sensitive Credential Data in the Data Dictionary Encrypting Sensitive Credential Data in the Data Dictionary	16-1
		he Multitenant Option Affects the Encryption of Sensitive Data	16-2
		pting Sensitive Credential Data in System Tables	16-2
	,	ring Sensitive Credential Data in the SYS.LINK\$ System Table	16-3
	-	ng Sensitive Credential Data in System Tables	16-4
		ring the Functioning of Database Links After a Lost Keystore	16-5
		Dictionary Views for Encrypted Data Dictionary Credentials	16-6
17	Manually	Encrypting Data	
	17.1 Secur	ity Problems That Encryption Does Not Solve	17-1
	17.1.1	Principle 1: Encryption Does Not Solve Access Control Problems	17-1
	17.1.2	Principle 2: Encryption Does Not Protect Against a Malicious Administrator	17-2
	17.1.3	Principle 3: Encrypting Everything Does Not Make Data Secure	17-3
	17.2 Data I	Encryption Challenges	17-4
	17.2.1	Encrypted Indexed Data	17-4
	17.2.2	Generated Encryption Keys	17-5
	17.2.3	Transmitted Encryption Keys	17-5
		, , , , , , , , , , , , , , , , , , ,	1, 0
	17.2.4	Storing Encryption Keys	17-5



		17.2.4	.2 Storage of Encryption Keys in the Database	17-6
		17.2.4	.3 Storage of Encryption Keys in the Operating System	17-7
		17.2.4	.4 Users Managing Their Own Encryption Keys	17-8
		17.2.4	.5 Manual Encryption with Transparent Database Encryption and Tablespace Encryption	17-8
	17	7.2.5 lı	mportance of Changing Encryption Keys	17-8
	17	7.2.6 E	Encryption of Binary Large Objects	17-8
	17.3	Data E	ncryption Storage with the DBMS_CRYPTO Package	17-9
	17.4	Asymn	netric Key Operations with the DBMS_CRYPTO Package	17-12
	17.5	Using (Ciphertexts Encrypted in OFB Mode in Oracle Database Release 11g	17-12
	17.6	Examp	les of Using the Data Encryption API	17-15
	17	7.6.1 E	Example: Data Encryption Procedure	17-15
	17	7.6.2 E	Example: AES 256-Bit Data Encryption and Decryption Procedures	17-16
	17	7.6.3 E	Example: Encryption and Decryption Procedures for BLOB Data	17-17
	17.7	Data D	ictionary Views for Encrypted Data	17-20
L8	Con	•	g Oracle Database Native Network Encryption and D	Oata
		9 -7		
	18.1	About (Oracle Database Native Network Encryption and Data Integrity	18-1
	18.1		Oracle Database Native Network Encryption and Data Integrity How Oracle Database Native Network Encryption and Integrity Works	18-1 18-2
	18	3.1.1 F	How Oracle Database Native Network Encryption and Integrity Works	18-2
	18 18	3.1.1 F 3.1.2 <i>A</i>	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard	
	18 18 18	3.1.1 H 3.1.2 A 3.1.3 T	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption	18-2 18-2 18-2
	18 18 18	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi	18-2 18-2 18-2
	18 18 18	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption	18-2 18-2 18-2 ty 18-3
	18 18 18 18.2 18.3	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity	18-2 18-2 18-2 ty 18-3 18-3
	18 18 18 18.2 18.3	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security	18-2 18-2 18-2 ty 18-3 18-3 18-4
	18 18 18 18.2 18.3	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4
	18 18 18 18.2 18.3 18	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Criple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5
	18 18 18 18.2 18.3 18 18.4	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption integrity Algorithms Support	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5 18-6
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption integrity Algorithms Support Hellman Based Key Negotiation	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5 18-6 18-7
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu 3.6.1 A	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Criple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption integrity Algorithms Support Hellman Based Key Negotiation uration of Data Encryption and Integrity	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5 18-6 18-7
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu 3.6.1 A	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Criple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption integrity Algorithms Support Hellman Based Key Negotiation uration of Data Encryption and Integrity About Activating Encryption and Integrity About Negotiating Encryption and Integrity	18-2 18-2 18-2 ty 18-3 18-4 18-4 18-5 18-6 18-7 18-7 18-8
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu 3.6.1 A 3.6.2 A	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption integrity Algorithms Support Hellman Based Key Negotiation furation of Data Encryption and Integrity About Activating Encryption and Integrity About Negotiating Encryption and Integrity About Negotiating Encryption and Integrity About Negotiating Encryption and Integrity About the Values for Negotiating Encryption and Integrity	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5 18-6 18-7 18-7 18-8 18-9
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu 3.6.1 A 18.6.2 A	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption integrity Algorithms Support Hellman Based Key Negotiation furation of Data Encryption and Integrity About Activating Encryption and Integrity About Negotiating Encryption and Integrity About Negotiating Encryption and Integrity Respected Configuration Parameter	18-2 18-2 18-2 ty 18-3 18-4 18-4 18-5 18-6 18-7 18-7 18-8 18-9
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu 3.6.1 A 3.6.2 A 18.6.2	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Security Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption Integrity Algorithms Support Hellman Based Key Negotiation For Data Encryption and Integrity About Activating Encryption and Integrity About Negotiating Encryption and Integrity About Negotiating Encryption and Integrity E.1 About the Values for Negotiating Encryption and Integrity REJECTED Configuration Parameter E.3 ACCEPTED Configuration Parameter	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5 18-6 18-7 18-7 18-8 18-9 18-9 18-10
	18 18 18.2 18.3 18 18.4 18.5 18.6	3.1.1 H 3.1.2 A 3.1.3 T 3.1.4 C Oracle Improv 3.3.1 A 3.3.2 A Data In Diffie-H Configu 3.6.1 A 18.6.2 18.6.2 18.6.2	How Oracle Database Native Network Encryption and Integrity Works Advanced Encryption Standard Triple-DES Encryption Choosing Between Native Network Encryption and Transport Layer Securi Database Native Network Encryption Data Integrity ing Native Network Encryption Security About Improving Native Network Encryption Security Applying Security Improvement Updates to Native Network Encryption attegrity Algorithms Support Hellman Based Key Negotiation furation of Data Encryption and Integrity About Activating Encryption and Integrity About Negotiating Encryption and Integrity About Negotiating Encryption and Integrity REJECTED Configuration Parameter ACCEPTED Configuration Parameter REQUESTED Configuration Parameter	18-2 18-2 18-2 ty 18-3 18-3 18-4 18-4 18-5 18-6 18-7 18-7 18-8 18-9 18-9 18-10

17.2.4.1 About Storing Encryption Keys



17-6

17-6

	18.6.3.1 Configuring Encryption on the Client and the Server	18-11
	18.6.3.2 Configuring Integrity on the Client and the Server	18-13
	18.6.3.3 Enabling Both Oracle Native Encryption and SSL Authentication for	
	Different Users Concurrently	18-14
19	Configuring the Thin JDBC Client Network	
	19.1 About the Java Implementation	19-1
	19.2 Java Database Connectivity Support	19-1
	19.3 Thin JDBC Features	19-2
	19.4 Implementation Overview	19-3
	19.5 Obfuscation of the Java Cryptography Code	19-3
	19.6 Configuration Parameters for the Thin JDBC Network Implementation	19-4
	19.6.1 About the Thin JDBC Network Implementation Configuration Parameters	19-4
	19.6.2 Client Encryption Level Parameter	19-5
	19.6.3 Client Encryption Selected List Parameter	19-5
	19.6.4 Client Integrity Level Parameter	19-5
	19.6.5 Client Integrity Selected List Parameter	19-6
	19.6.6 Client Authentication Service Parameter	19-6
	19.6.7 AnoServices Constants	19-7
Par	t V Managing Strong Authentication	
20	Introduction to Strong Authentication	
	20.1 What Is Strong Authentication?	20-1
	20.2 Centralized Authentication and Single Sign-On	20-2
	20.3 How Centralized Network Authentication Works	20-2
	20.4 Supported Strong Authentication Methods	20-3
	20.4.1 About Kerberos	20-4
	20.4.2 About Remote Authentication Dial-In User Service (RADIUS)	20-4
	20.4.3 About Transport Layer Security	20-5
	20.5 Oracle Database Native Network Encryption/Strong Authentication Architecture	20-6
	20.6 System Requirements for Strong Authentication	20-7
	20.7 Oracle Database Native Network Encryption and Strong Authentication Restrictions	20-7
21	Strong Authentication Administration Tools	
	21.1 About the Configuration and Administration Tools	21-1
	21.2 Native Network Encryption and Strong Authentication Configuration Tools	21-1
	21.2.1 About Oracle Net Manager	21-1
	21.2.2 Kerberos Adapter Command-Line Utilities	21-2



21.3 Pub	ic Key Infrastructure Credentials Management Tools	21-2
21.3.1	About Oracle Wallet Manager	21-3
21.3.2	About the orapki Utility	21-3
21.4 Duti	es of Strong Authentication Administrators	21-4
Configu	ing Kerberos Authentication	
22.1 Ena	oling Kerberos Authentication	22-1
22.1.1	Step 1: Install Kerberos	22-2
22.1.2	Step 2: Configure a Service Principal for an Oracle Database Server	22-3
22.1.3	Step 3: Extract a Service Key Table from Kerberos	22-3
22.1.4	Step 4: Install an Oracle Database Server and an Oracle Client	22-4
22.1.5	Step 5: Configure Oracle Net Services and Oracle Database	22-4
22.1.6	Step 6: Configure Kerberos Authentication	22-4
22.	6.1 Step 6A: Configure Kerberos on the Client and on the Database Serve	er 22-5
22.	6.2 Step 6B: Set the Initialization Parameters	22-7
22.	6.3 Step 6C: Set sqlnet.ora Parameters (Optional)	22-8
22.1.7	Step 7: Create a Kerberos User	22-10
22.1.8	Step 8: Create an Externally Authenticated Oracle User	22-10
22.1.9	Step 9: Get an Initial Ticket for the Kerberos/Oracle User	22-11
22.2 Utilit	es for the Kerberos Authentication Adapter	22-11
22.2.1	okinit Utility Options for Obtaining the Initial Ticket	22-12
22.2.2	oklist Utility Options for Displaying Credentials	22-13
22.2.3	okdstry Utility Options for Removing Credentials from the Cache File	22-14
22.2.4	okcreate Utility Options for Automatic Keytab Creation	22-14
22.3 Con	necting to an Oracle Database Server Authenticated by Kerberos	22-15
22.4 Con	iguring Interoperability with a Windows 2008 Domain Controller KDC	22-16
22.4.1	About Configuring Interoperability with a Microsoft Windows Server Domain Controller KDC	22-16
22.4.2	Step 1: Configure Oracle Kerberos Client for Windows 2008 Domain Contro	oller 22-16
22.	1.2.1 Step 1A: Create the Client Kerberos Configuration Files	22-17
22.	1.2.2 Step 1B: Specify the Oracle Configuration Parameters in the sqlnet.or File	ra 22-17
22.	1.2.3 Step 1C: Optionally, Specify Additional Kerberos Principals Using tnsnames.ora	22-18
22.	1.2.4 Step 1D: Specify the Listening Port Number	22-18
22.4.3	Step 2: Configure a Microsoft Windows Server Domain Controller KDC for to Oracle Client	he 22-19
22.	3.3.1 Step 2A: Create the User Account	22-19
22.	1.3.2 Step 2B: Create the Oracle Database Principal User Account and Key	/tab 22-19
22.4.4	Step 3: Configure Oracle Database for a Microsoft Windows Server Domain Controller KDC	22-20
22.	4.4.1 Step 3A: Set Configuration Parameters in the sqlnet.ora File	22-20



22-21 22-21 22-22 22-23 22-23
22-22 22-23
22-23
22-23
22-24
22-24
22-25
23-2
23-2
23-2
23-3
ke 23-4
23-4
23-5
23-5
23-6
23-6
23-6
23-7
23-8
23-8
23-9
23-9
23-10
23-10
23-11
23-11
23-12
23-13
23-14
23-14
23-15
23-16
23-19
23-20
k



		23.9.	1.6	Step 1F: Set Transport Layer Security as an Authentication Service on	
				the Server (Optional)	23-21
		23.9.	1.7	Step 1G: Disable SSLv3 on the Server and Client (Optional)	23-22
		23.9.	1.8	Step 1H: Create a Listening Endpoint that Uses TCP/IP with Transport Layer Security on the Server	23-22
		23.9.	1.9	Step 1H: Restart the Database	23-22
	23.	9.2	Step	2: Configure Transport Layer Security on the Client	23-22
		23.9.	2.1	Step 2A: Confirm Client Wallet Creation	23-23
		23.9.	2.2	Step 2B: Configure Server DN Matching and Use TCP/IP with TLS on the Client	23-23
		23.9.	2.3	Step 2C: Specify Required Client TLS Configuration (Wallet Location)	23-26
		23.9.	2.4	Step 2D: Set the Client Transport Layer Security Cipher Suites (Optional)	23-28
		23.9.	2.5	Step 2E: Set the Required TLS Version on the Client (Optional)	23-30
		23.9.	2.6	Step 2F: Set TLS as an Authentication Service on the Client (Optional)	23-31
		23.9.	2.7	Step 2G: Specify the Certificate to Use for Authentication on the Client (Optional)	23-31
		23.9.	2.8	Step 2H: Check That the Connections Are Using Transport Layer	
				Security	23-32
		23.9.		Step 2I: Restart the Database	23-32
	_			3: Log in to the Database Instance	23-33
23.			•	Layer Security Oracle Wallet Search Order	23-33
23.	11		sport ronme	Layer Security Connections in an Oracle Real Application Clusters ent	23-34
	23.	11.1	Step	1: Configure TCPS Protocol Endpoints	23-34
	23.	11.2		2: Ensure That the LOCAL_LISTENER Parameter Is Correctly Set on h Node	23-36
	23.	11.3	Step	3: Create Transport Layer Security Wallets and Certificates	23-36
		23.11	3.1	Oracle Real Application Clusters Components That Need Certificates	23-37
		23.11	3.2	Creating Transport Layer Security Wallets and Certificates	23-37
	23.	11.4	Step	4: Create a Wallet in Each Node of the Oracle RAC Cluster	23-39
	23.	11.5	Step	5: Define Wallet Locations in the listener.ora and sqlnet.ora Files	23-40
	23.	11.6	Step	6: Restart the Database Instances and Listeners	23-40
	23.	11.7	Step	7: Test the Cluster Node Configuration	23-41
	23.	11.8	Step	8: Test the Remote Client Configuration	23-41
23.	12			ng Transport Layer Security for Client Authentication and Encryption Using Certificate Store	23-42
	23.	12.1		ut Configuring Transport Layer Security for Client Authentication and ryption Using Microsoft Certificate Store	23-43
	23.	12.2	Step	1: Create and Configure the Server Wallet	23-43
	23.	12.3	Step	2: Create and Configure the Client Wallet	23-44
	23.	12.4		3: Create an External User in the Oracle Database	23-45
	23.	12.5	Step	4: Configure the Server listener.ora File	23-45
	23.	12.6	Step	5: Configure the Server sqlnet.ora File	23-46
	23.	12.7	Step	6: Import the Client Wallet into the Microsoft Certificate Store	23-46



	23.12.8	Step	7: Configure the Client sqlnet.ora File	23-47
	23.12.9	Step	8: Configure the Oracle Database	23-47
	23.12.10	Ste	p 9: Test the Client and Server Connection	23-47
23.2	13 Trou	blesho	ooting the Transport Layer Security Configuration	23-48
23.2	14 Certi	ficate	Validation with Certificate Revocation Lists	23-51
	23.14.1	Abou	ut Certificate Validation with Certificate Revocation Lists	23-51
	23.14.2	Wha	t CRLs Should You Use?	23-52
	23.14.3	How	CRL Checking Works	23-52
	23.14.4	Conf	figuring Certificate Validation with Certificate Revocation Lists	23-53
	23.14	.4.1	About Configuring Certificate Validation with Certificate Revocation Lists	23-53
	23.14	.4.2	Enabling Certificate Revocation Status Checking for the Client or Server	23-53
	23.14	.4.3	Disabling Certificate Revocation Status Checking	23-55
	23.14.5	Certi	ificate Revocation List Management	23-56
	23.14	.5.1	About Certificate Revocation List Management	23-56
	23.14	.5.2	Displaying orapki Help for Commands That Manage CRLs	23-57
	23.14	.5.3	Renaming CRLs with a Hash Value for Certificate Validation	23-57
	23.14	1.5.4	Uploading CRLs to Oracle Internet Directory	23-58
	23.14	.5.5	Listing CRLs Stored in Oracle Internet Directory	23-58
	23.14	.5.6	Viewing CRLs in Oracle Internet Directory	23-59
	23.14	.5.7	Deleting CRLs from Oracle Internet Directory	23-60
	23.14.6	Trou	bleshooting CRL Certificate Validation	23-60
	23.14.7	Orac	cle Net Tracing File Error Messages Associated with Certificate Validation	23-61
23.2	15 Conf	igurin	g Your System to Use Hardware Security Modules	23-62
	23.15.1	Gen	eral Guidelines for Using Hardware Security Modules for TLS	23-63
	23.15.2 Cor		figuring Your System to Use nCipher Hardware Security Modules	23-63
	23.15	5.2.1	About Configuring Your System to Use nCipher Hardware Security	00.00
	00.45		Modules	23-63
	23.15	0.2.2	Oracle Components Required To Use an nCipher Hardware Security Module	23-64
	23.15	5.2.3	Directory Path Requirements for Installing an nCipher Hardware Security Module	23-64
	23.15.3	Conf	figuring Your System to Use SafeNET Hardware Security Modules	23-65
	23.15	5.3.1	About Configuring Your System to Use SafeNET Hardware Security Modules	23-65
	23.15	5.3.2	Oracle Components Required for SafeNET Luna SA Hardware Security Modules	23-65
	23.15	5.3.3	Directory Path Requirements for Installing a SafeNET Hardware	
			Security Module	23-66
	23.15.4 Tro		bleshooting Using Hardware Security Modules	23-66
	23.15	5.4.1	Errors in the Oracle Net Trace Files	23-66
	23.15	.4.2	Error Messages Associated with Using Hardware Security Modules	23-67



24 Configuring RADIUS Authentication

24.1	Abou	ıt Con	figuring RADIUS Authentication	24-1
24.2	RAD	IUS C	Components	24-2
24.3	RAD	IUS A	authentication Modes	24-3
24	4.3.1	Synd	chronous Authentication Mode	24-3
	24.3	3.1.1	Sequence for Synchronous Authentication Mode	24-3
	24.3	3.1.2	Example: Synchronous Authentication with SecurID Token Cards	24-4
24	1.3.2	Chal	llenge-Response (Asynchronous) Authentication Mode	24-5
	24.3	3.2.1	Sequence for Challenge-Response (Asynchronous) Authentication Mode	24-5
	24.3	3.2.2	Example: Asynchronous Authentication with Smart Cards	24-7
	24.3	3.2.3	Example: Asynchronous Authentication with ActivCard Tokens	24-7
24.4	Enal	oling F	RADIUS Authentication, Authorization, and Accounting	24-8
24	4.4.1	Step	1: Configure RADIUS Authentication	24-8
	24.4	1.1.1	Step 1A: Configure RADIUS on the Oracle Client	24-9
	24.4	1.1.2	Step 1B: Configure RADIUS on the Oracle Database Server	24-10
	24.4	1.1.3	Step 1C: Configure Additional RADIUS Features	24-13
24	1.4.2	Step	2: Create a User and Grant Access	24-16
24	1.4.3	Step	3: Configure External RADIUS Authorization (Optional)	24-16
	24.4	1.3.1	Step 3A: Configure the Oracle Server (RADIUS Client)	24-16
	24.4	1.3.2	Step 3B: Configure the Oracle Client Where Users Log In	24-17
	24.4	1.3.3	Step 3C: Configure the RADIUS Server	24-17
24	1.4.4	Step	4: Configure RADIUS Accounting	24-18
	24.4	1.4.1	Step 4A: Set RADIUS Accounting on the Oracle Database Server	24-18
	24.4	1.4.2	Step 4B: Configure the RADIUS Accounting Server	24-18
24	4.4.5	Step	5: Add the RADIUS Client Name to the RADIUS Server Database	24-19
24	4.4.6	Step	6: Configure the Authentication Server for Use with RADIUS	24-19
24	1.4.7	Step	7: Configure the RADIUS Server for Use with the Authentication Server	24-19
24	1.4.8	Step	8: Configure Mapping Roles	24-20
24.5	Usin	g RAE	DIUS to Log in to a Database	24-20
24.6	RSA	ACE/	Server Configuration Checklist	24-21
Cus	tomi	zing	the Use of Strong Authentication	
25.1	Coni	nectin	g to a Database Using Strong Authentication	25-1
25.2	Disa	bling S	Strong Authentication and Native Network Encryption	25-2
25.3	Conf	igurin	g Multiple Authentication Methods	25-4
25.4	Conf	igurin	g Oracle Database for External Authentication	25-5
2	5.4.1	Setti	ng the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora	25-5
25	25.4.2 Setting OS_AUTHENT_PREFIX to a Null Value		25-6	



25

Part VI Monitoring Database Activity with Auditing

Intro	duction	to Auditing	
26.1	What Is A	uditing?	26-2
26.2	Why Is Au	Iditing Used?	26-3
26.3	Best Pract	tices for Auditing	26-3
26.4	What Is U	nified Auditing?	26-4
26.5	Benefits of	f the Unified Audit Trail	26-5
26.6	Checking i	if Your Database Has Migrated to Pure Unified Auditing	26-5
26.7	Mixed Mod	de Auditing	26-6
26.	.7.1 Abo	ut Mixed Mode Auditing	26-6
26.	.7.2 Enal	blement of Unified Auditing	26-7
26.	.7.3 How	Database Creation Determines the Type of Auditing You Have Enabled	26-8
26.	.7.4 Cap	abilities of Mixed Mode Auditing	26-8
26.8	Who Can	Perform Auditing?	26-9
26.9	Unified Au	ıditing in a Multitenant Environment	26-9
26.10	Auditing	in a Distributed Database	26-10
	-	an Auditing Type	27-1 27-1
27.1	Selecting a	an Auditing Type	27-1
27.	.1.1 Audi	iting SQL Statements, Privileges, and Other General Activities	27-1
		iting Commonly Used Security-Relevant Activities	27-2
		iting Specific, Fine-Grained Activities	27-2
27.2	_	ctivities with Unified Audit Policies and the AUDIT Statement	27-3
27.	.2.1 Abo	ut Auditing Activities with Unified Audit Policies and AUDIT	27-4
27.	.2.2 Best	t Practices for Creating Custom Unified Audit Policies	27-5
27.	.2.3 Synt	tax for Creating a Unified Audit Policy	27-5
27.	.2.4 Audi	iting Roles	27-7
	27.2.4.1	About Role Auditing	27-7
	27.2.4.2	Configuring Role Unified Audit Policies	27-8
	27.2.4.3	Example: Auditing the DBA Role in a Multitenant Environment	27-8
27.	.2.5 Audi	iting System Privileges	27-8
	27.2.5.1	About System Privilege Auditing	27-9
	27.2.5.2	System Privileges That Can Be Audited	27-9
	27.2.5.3	System Privileges That Cannot Be Audited	27-10
	27.2.5.4	Configuring a Unified Audit Policy to Capture System Privilege Use	27-10
	27.2.5.5	Example: Auditing a User Who Has ANY Privileges	27-10
	27.2.5.6	Example: Using a Condition to Audit a System Privilege	27-11
	27.2.5.7	How System Privilege Unified Audit Policies Appear in the Audit Trail	27-11
27.	.2.6 Audi	iting Administrative Users	27-11



27.2.6.1	Administrative User Accounts That Can Be Audited	27-12
27.2.6.2	Configuring a Unified Audit Policy to Capture Administrator Activities	27-12
27.2.6.3	Example: Auditing the SYS User	27-12
27.2.7 Aud	liting Object Actions	27-13
27.2.7.1	About Auditing Object Actions	27-13
27.2.7.2	Object Actions That Can Be Audited	27-14
27.2.7.3	Configuring an Object Action Unified Audit Policy	27-14
27.2.7.4	Example: Auditing Actions on SYS Objects	27-15
27.2.7.5	Example: Auditing Multiple Actions on One Object	27-15
27.2.7.6	Example: Auditing GRANT and REVOKE Operations on an Object	27-15
27.2.7.7	Example: Auditing Both Actions and Privileges on an Object	27-16
27.2.7.8	Example: Auditing All Actions on a Table	27-16
27.2.7.9	Example: Auditing All Actions in the Database	27-16
27.2.7.10	How Object Action Unified Audit Policies Appear in the Audit Trail	27-17
27.2.7.11	Auditing Functions, Procedures, Packages, and Triggers	27-17
27.2.7.12	2 Auditing of Oracle Virtual Private Database Predicates	27-18
27.2.7.13	Audit Policies for Oracle Virtual Private Database Policy Functions	27-19
27.2.7.14	Unified Auditing with Editioned Objects	27-20
27.2.8 Aud	liting the READ ANY TABLE and SELECT ANY TABLE Privileges	27-20
27.2.8.1	About Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges	27-20
27.2.8.2	Creating a Unified Audit Policy to Capture READ Object Privilege Operations	27-21
27.2.8.3	How the Unified Audit Trail Captures READ ANY TABLE and SELECT ANY TABLE	27-21
27.2.9 Aud	liting SQL Statements and Privileges in a Multitier Environment	27-23
27.2.10 Cr	eating a Condition for a Unified Audit Policy	27-25
27.2.10.2	L About Conditions in Unified Audit Policies	27-26
27.2.10.2	Configuring a Unified Audit Policy with a Condition	27-26
27.2.10.3	B Example: Auditing Access to SQL*Plus	27-28
27.2.10.4	Example: Auditing Actions Not in Specific Hosts	27-28
27.2.10.	Example: Auditing Both a System-Wide and a Schema-Specific Action	27-28
27.2.10.6	Example: Auditing a Condition Per Statement Occurrence	27-28
27.2.10.	7 Example: Unified Audit Session ID of a Current Administrative User Session	27-29
27.2.10.8	Example: Unified Audit Session ID of a Current Non-Administrative User Session	27-29
27.2.10.9	How Audit Records from Conditions Appear in the Audit Trail	27-30
27.2.11 Au	diting Application Context Values	27-30
27.2.11.1	About Auditing Application Context Values	27-30
27.2.11.2	2 Configuring Application Context Audit Settings	27-31
27.2.11.3	B Disabling Application Context Audit Settings	27-31
27.2.11.4	Example: Auditing Application Context Values in a Default Database	27-32



27.2.1	1.5	Example: Auditing Application Context values from Oracle Label	07.00
		•	27-32
	_		27-32
			27-32
27.2.1	2.1	About Auditing Oracle Database Real Application Security Events	27-33
27.2.1	2.2	Oracle Database Real Application Security Auditable Events	27-34
27.2.1	L2.3	Oracle Database Real Application Security User, Privilege, and Role Audit Events	27-34
27.2.1	2.4	Oracle Database Real Application Security Security Class and ACL Audit Events	27-36
27.2.1	2.5	Oracle Database Real Application Security Session Audit Events	27-37
27.2.1	2.6	Oracle Database Real Application Security ALL Events	27-38
27.2.1	L2.7	Configuring a Unified Audit Policy for Oracle Database Real Application Security	27-38
27.2.1	2.8	Example: Auditing Real Application Security User Account Modifications	27-39
27.2.1	2.9	Example: Using a Condition in a Real Application Security Unified Audit Policy	27-39
27.2.1	L2.10	How Oracle Database Real Application Security Events Appear in the Audit Trail	27-39
2.13	Audit	ing Oracle Recovery Manager Events	27-39
27.2.1	13.1	About Auditing Oracle Recovery Manager Events	27-40
27.2.1	13.2	Oracle Recovery Manager Unified Audit Trail Events	27-40
27.2.1	13.3	How Oracle Recovery Manager Audited Events Appear in the Audit Trail	27-41
2.14	Audit	ing Oracle Database Vault Events	27-41
27.2.1	4.1	About Auditing Oracle Database Vault Events	27-42
27.2.1	4.2	Who Is Audited in Oracle Database Vault?	27-43
27.2.1	4.3	About Oracle Database Vault Unified Audit Trail Events	27-43
27.2.1	4.4	Oracle Database Vault Realm Audit Events	27-44
27.2.1	4.5	Oracle Database Vault Rule Set and Rule Audit Events	27-44
27.2.1	4.6	Oracle Database Vault Command Rule Audit Events	27-45
27.2.1	4.7	Oracle Database Vault Factor Audit Events	27-46
27.2.1	L4.8	Oracle Database Vault Secure Application Role Audit Events	27-47
27.2.1	4.9	Oracle Database Vault Oracle Label Security Audit Events	27-47
27.2.1	4.10	Oracle Database Vault Oracle Data Pump Audit Events	27-48
27.2.1	4.11	Oracle Database Vault Enable and Disable Audit Events	27-48
27.2.1	4.12	Configuring a Unified Audit Policy for Oracle Database Vault	27-49
27.2.1	4.13	Example: Auditing an Oracle Database Vault Realm	27-49
27.2.1	4.14	Example: Auditing an Oracle Database Vault Rule Set	27-50
27.2.1	4.15	Example: Auditing Two Oracle Database Vault Events	27-50
27.2.1	4.16	Example: Auditing Oracle Database Vault Factors	27-50
27.2.1	4.17	How Oracle Database Vault Audited Events Appear in the Audit Trail	27-50
2.15	Audit	ing Oracle Label Security Events	27-51
27.2.1	L5.1	About Auditing Oracle Label Security Events	27-51
	27.2.1 27.2.1	27.2.12.1 27.2.12.2 27.2.12.3 27.2.12.4 27.2.12.5 27.2.12.6 27.2.12.7 27.2.12.8 27.2.12.9 27.2.12.10 2.13 Audit 27.2.13.1 27.2.13.2 27.2.13.3 2.14 Audit 27.2.14.1 27.2.14.2 27.2.14.4 27.2.14.5 27.2.14.6 27.2.14.7 27.2.14.8 27.2.14.10 27.2.14.11 27.2.14.12 27.2.14.11 27.2.14.12 27.2.14.15 27.2.14.16 27.2.14.15 27.2.14.16 27.2.14.16 27.2.14.16 27.2.14.17	Security 27.2.11.6 How Audited Application Contexts Appear in the Audit Trail 2.12 Auditing Oracle Database Real Application Security Events 27.2.12.1 About Auditing Oracle Database Real Application Security Events 27.2.12.2 Oracle Database Real Application Security Events 27.2.12.3 Oracle Database Real Application Security User, Privilege, and Role Audit Events 27.2.12.4 Oracle Database Real Application Security Security Class and ACL Audit Events 27.2.12.5 Oracle Database Real Application Security Session Audit Events 27.2.12.6 Oracle Database Real Application Security Session Audit Events 27.2.12.7 Configuring a Unified Audit Policy for Oracle Database Real Application Security 27.2.12.8 Example: Auditing Real Application Security User Account Modifications Security 27.2.12.9 Example: Using a Condition in a Real Application Security Unified Audit Policy 27.2.12.10 How Oracle Database Real Application Security Events Appear in the Audit Trail 2.13 Auditing Oracle Recovery Manager Events 27.2.13.1 About Auditing Oracle Recovery Manager Events 27.2.13.2 Oracle Recovery Manager Unified Audit Trail Events 27.2.13.3 How Oracle Database Vault Events 27.2.14.1 About Auditing Oracle Database Vault Events 27.2.14.2 Who Is Audited in Oracle Database Vault Events 27.2.14.3 About Oracle Database Vault Unified Audit Trail Events 27.2.14.4 Oracle Database Vault Realm Audit Events 27.2.14.5 Oracle Database Vault Realm Audit Events 27.2.14.6 Oracle Database Vault Command Rule Audit Events 27.2.14.7 Oracle Database Vault Feator Audit Events 27.2.14.9 Oracle Database Vault Feator Audit Events 27.2.14.1 Oracle Database Vault Feator Babase Vault Events 27.2.14.1 Example:



	27.2.15.2	Oracle Label Security Unified Audit Trail Events	27-52
	27.2.15.3	Oracle Label Security Auditable User Session Labels	27-54
	27.2.15.4	Configuring a Unified Audit Policy for Oracle Label Security	27-54
	27.2.15.5	Example: Auditing Oracle Label Security Session Label Attributes	27-55
	27.2.15.6	Example: Excluding a User from an Oracle Label Security Policy	27-55
	27.2.15.7	Example: Auditing Oracle Label Security Policy Actions	27-55
	27.2.15.8	Example: Querying for Audited OLS Session Labels	27-55
	27.2.15.9	How Oracle Label Security Audit Events Appear in the Audit Trail	27-56
27	.2.16 Audi	ting Oracle Data Mining Events	27-56
	27.2.16.1	About Auditing Oracle Data Mining Events	27-57
	27.2.16.2	Oracle Data Mining Unified Audit Trail Events	27-57
	27.2.16.3	Configuring a Unified Audit Policy for Oracle Data Mining	27-57
	27.2.16.4	Example: Auditing Multiple Oracle Data Mining Operations by a User	27-58
	27.2.16.5	Example: Auditing All Failed Oracle Data Mining Operations by a User	27-58
	27.2.16.6	How Oracle Data Mining Events Appear in the Audit Trail	27-58
27	.2.17 Audi	ting Oracle Data Pump Events	27-59
	27.2.17.1	About Auditing Oracle Data Pump Events	27-59
	27.2.17.2	Oracle Data Pump Unified Audit Trail Events	27-60
	27.2.17.3	Configuring a Unified Audit Policy for Oracle Data Pump	27-60
	27.2.17.4	Example: Auditing Oracle Data Pump Import Operations	27-60
	27.2.17.5	Example: Auditing All Oracle Data Pump Operations	27-60
	27.2.17.6	How Oracle Data Pump Audited Events Appear in the Audit Trail	27-61
27	.2.18 Audi	ting Oracle SQL*Loader Direct Load Path Events	27-61
	27.2.18.1	About Auditing in Oracle SQL*Loader Direct Path Load Events	27-61
	27.2.18.2	Oracle SQL*Loader Direct Load Path Unified Audit Trail Events	27-62
	27.2.18.3	Configuring a Unified Audit Trail Policy for Oracle SQL*Loader Direct Path Events	27-62
	27.2.18.4	Example: Auditing Oracle SQL*Loader Direct Path Load Operations	27-63
	27.2.18.5	How SQL*Loader Direct Path Load Audited Events Appear in the Audit Trail	27-63
27	.2.19 Audi	ting Only Top-Level Statements	27-63
	27.2.19.1	About Auditing Only Top-Level SQL Statements	27-64
	27.2.19.2	Configuring a Unified Audit Policy to Capture Only Top-Level Statements	27-64
	27.2.19.3	Example: Auditing Top-Level Statements	27-64
	27.2.19.4	Example: Comparison of Top-Level SQL Statement Audits	27-64
	27.2.19.5	How the Unified Audit Trail Captures Top-Level SQL Statements	27-70
27	.2.20 Unifi	ed Audit Policies or AUDIT Settings in a Multitenant Environment	27-70
	27.2.20.1	About Local, CDB Common, and Application Common Audit Policies	27-71
	27.2.20.2	Traditional Auditing in a Multitenant Environment	27-72
	27.2.20.3	Configuring a Local Unified Audit Policy or Common Unified Audit Policy	27-73
	27.2.20.4	Example: Local Unified Audit Policy	27-74
	27.2.20.5	Example: CDB Common Unified Audit Policy	27-75



27.2	2.20.6	Example: Application Common Unified Audit Policy	27-75
27.2	2.20.7	How Local or Common Audit Policies or Settings Appear in the Audit Trail	27-76
27.2.21	Alte	ring Unified Audit Policies	27-76
27.2	2.21.1	About Altering Unified Audit Policies	27-77
27.2	2.21.2	Altering a Unified Audit Policy	27-77
27.2	2.21.3	Example: Altering a Condition in a Unified Audit Policy	27-78
27.2	2.21.4	Example: Altering an Oracle Label Security Component in a Unified Audit Policy	27-78
27.	2.21.5	Example: Altering Roles in a Unified Audit Policy	27-79
27.2	2.21.6	Example: Dropping a Condition from a Unified Audit Policy	27-79
27.2	2.21.7	Example: Altering an Existing Unified Audit Policy Top-Level Statement Audits	27-79
27.2.22	Ena	bling and Applying Unified Audit Policies to Users and Roles	27-79
27.2	2.22.1	About Enabling Unified Audit Policies	27-80
27.2	2.22.2	Enabling a Unified Audit Policy	27-81
27.2	2.22.3	Example: Enabling a Unified Audit Policy	27-82
27.2.23	Disa	abling Unified Audit Policies	27-82
27.2	2.23.1	About Disabling Unified Audit Policies	27-82
27.2	2.23.2	Disabling a Unified Audit Policy	27-82
27.2	2.23.3	Example: Disabling a Unified Audit Policy	27-83
27.2.24	Dro	oping Unified Audit Policies	27-83
27.2	2.24.1	About Dropping Unified Audit Policies	27-83
27.2	2.24.2	Dropping a Unified Audit Policy	27-84
27.2	2.24.3	Example: Disabling and Dropping a Unified Audit Policy	27-84
27.2.25	Tuto	orial: Auditing Nondatabase Users	27-84
27.2	2.25.1	Step 1: Create the User Accounts and Ensure the User OE Is Active	27-85
27.2	2.25.2	Step 2: Create the Unified Audit Policy	27-85
27.2	2.25.3	Step 3: Test the Policy	27-86
27.2	2.25.4	Step 4: Remove the Components of This Tutorial	27-87
27.3 Audi	ting Ac	tivities with the Predefined Unified Audit Policies	27-87
27.3.1	Logo	n Failures Predefined Unified Audit Policy	27-88
27.3.2	Secu	re Options Predefined Unified Audit Policy	27-88
27.3.3	Orac	e Database Parameter Changes Predefined Unified Audit Policy	27-89
27.3.4	User	Account and Privilege Management Predefined Unified Audit Policy	27-90
27.3.5	Cente	er for Internet Security Recommendations Predefined Unified Audit Policy	27-90
27.3.6	Orac	e Database Real Application Security Predefined Audit Policies	27-91
27.3	3.6.1	System Administrator Operations Predefined Unified Audit Policy	27-91
27.3	3.6.2	Session Operations Predefined Unified Audit Policy	27-91
27.3.7		e Database Vault Predefined Unified Audit Policy for DVSYS and CSYS Schemas	27-92
27.3.8		e Database Vault Predefined Unified Audit Policy for Default Realms and mand Rules	27-92



	27.4 Audi	iting S	pecific Activities with Fine-Grained Auditing	27-93
	27.4.1	Abou	ut Fine-Grained Auditing	27-93
	27.4.2	Whe	ere Are Fine-Grained Audit Records Stored?	27-95
	27.4.3	Who	Can Perform Fine-Grained Auditing?	27-95
	27.4.4	Fine	-Grained Auditing on Tables or Views That Have Oracle VPD Policies	27-95
	27.4.5	Fine	-Grained Auditing in a Multitenant Environment	27-96
	27.4.6	Fine	-Grained Audit Policies with Editions	27-97
	27.4.7	Usin Polic	g the DBMS_FGA PL/SQL Package to Manage Fine-Grained Audit cies	27-97
	27.4	4.7.1	About the DBMS_FGA PL/SQL PL/SQL Package	27-98
	27.4	4.7.2	The DBMS_FGA PL/SQL Package with Editions	27-98
	27.4	4.7.3	The DBMS_FGA PL/SQL Package in a Multitenant Environment	27-99
	27.4	4.7.4	Creating a Fine-Grained Audit Policy	27-99
	27.4	4.7.5	Example: Using DBMS_FGA.ADD_POLICY to Create a Fine-Grained Audit Policy	27-102
	27.4	4.7.6	Disabling a Fine-Grained Audit Policy	27-103
	27.4	4.7.7	Enabling a Fine-Grained Audit Policy	27-103
	27.4	4.7.8	Dropping a Fine-Grained Audit Policy	27-104
	27.4.8	Tuto	rial: Adding an Email Alert to a Fine-Grained Audit Policy	27-104
	27.4	4.8.1	About This Tutorial	27-105
	27.4	4.8.2	Step 1: Install and Configure the UTL_MAIL PL/SQL Package	27-105
	27.4	4.8.3	Step 2: Create User Accounts	27-107
	27.4	4.8.4	Step 3: Configure an Access Control List File for Network Services	27-108
	27.4	4.8.5	Step 4: Create the Email Security Alert PL/SQL Procedure	27-108
	27.4	4.8.6	Step 5: Create and Test the Fine-Grained Audit Policy Settings	27-109
	27.4	4.8.7	Step 6: Test the Alert	27-109
	27.4	4.8.8	Step 7: Remove the Components of This Tutorial	27-110
	27.5 Audi	it Polic	cy Data Dictionary Views	27-111
28	Adminis	terin	g the Audit Trail	
	28.1 Man	aging	the Unified Audit Trail	28-1
	28.1.1	Whe	en and Where Are Audit Records Created?	28-2
	28.1.2	Activ	vities That Are Mandatorily Audited	28-2
	28.1.3	How	Do Cursors Affect Auditing?	28-4
	28.1.4	Writi	ng the Unified Audit Trail Records to the AUDSYS Schema	28-4
	28.1.5	Writi View	ng the Unified Audit Trail Records to SYSLOG or the Windows Event ver	28-5
	28.3	1.5.1	About Writing the Unified Audit Trail Records to SYSLOG or the Windows Event Viewer	28-5
	28.3	1.5.2	Enabling SYSLOG and Windows Event Viewer Captures for the Unified Audit Trail	28-6
	28.1.6	Whe	en Audit Records Are Written to the Operating System	28-7



	28.1.7	IVIOVI	ng Operating System Audit Records Into the Unified Audit Trail	28-8
	28.1.8	Expo	orting and Importing the Unified Audit Trail Using Oracle Data Pump	28-9
	28.1.9	Disal	bling Unified Auditing	28-10
	28.2 Arch	niving t	he Audit Trail	28-11
	28.2.1	Arch	iving the Traditional Operating System Audit Trail	28-11
	28.2.2	Arch	iving the Unified and Traditional Database Audit Trails	28-12
	28.3 Pur	ging Au	ıdit Trail Records	28-12
	28.3.1	Abou	ut Purging Audit Trail Records	28-13
	28.3.2	Sele	cting an Audit Trail Purge Method	28-13
	28.	3.2.1	Purging the Audit Trail on a Regularly Scheduled Basis	28-13
	28.	3.2.2	Manually Purging the Audit Trail at a Specific Time	28-14
	28.3.3	Sche	eduling an Automatic Purge Job for the Audit Trail	28-14
	28.	3.3.1	About Scheduling an Automatic Purge Job	28-14
	28.	3.3.2	Step 1: If Necessary, Tune Online and Archive Redo Log Sizes	28-15
	28.	3.3.3	Step 2: Plan a Timestamp and Archive Strategy	28-15
	28.	3.3.4	Step 3: Optionally, Set an Archive Timestamp for Audit Records	28-15
	28.	3.3.5	Step 4: Create and Schedule the Purge Job	28-17
	28.3.4	Manı	ually Purging the Audit Trail	28-19
	28.	3.4.1	About Manually Purging the Audit Trail	28-19
	28.	3.4.2	Using DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL to Manually Purge the Audit Trail	28-19
	28.3.5	Othe	r Audit Trail Purge Operations	28-21
	28.	3.5.1	Enabling or Disabling an Audit Trail Purge Job	28-21
	28.	3.5.2	Setting the Default Audit Trail Purge Job Interval for a Specified Purge Job	28-22
	28.	3.5.3	Deleting an Audit Trail Purge Job	28-23
		3.5.4	Clearing the Archive Timestamp Setting	28-23
	28.3.6		nple: Directly Calling a Unified Audit Trail Purge Operation	28-24
			Management Data Dictionary Views	28-25
Part	VII Ap	pend	lixes	
٨	Keening	ι Υοιιι	r Oracle Database Secure	
\wedge				
			Oracle Database Security Guidelines	A-1
			g Security Patches and Contacting Oracle Regarding Vulnerabilities	A-2
	A.2.1		loading Security Patches and Workaround Solutions	A-2
	A.2.2		acting Oracle Security Regarding Vulnerabilities in Oracle Database	A-2
			for Securing User Accounts and Privileges	A-3
			for Securing Roles	A-7
			for Securing Passwords	A-7
	A.6 Guid	eiines f	for Securing Data	A-10



	A.7 Guidelines for Securing the ORACLE_LOADER Access Driver	A-11
	A.8 Guidelines for Securing a Database Installation and Configuration	A-12
	A.9 Guidelines for Securing the Network	A-13
	A.9.1 Client Connection Security	A-13
	A.9.2 Network Connection Security	A-14
	A.9.3 Transport Layer Security Connection Security	A-17
	A.10 Guideline for Securing External Procedures	A-18
	A.11 Guidelines for Auditing	A-19
	A.11.1 Manageability of Audited Information	A-19
	A.11.2 Audits of Typical Database Activity	A-20
	A.11.3 Audits of Suspicious Database Activity	A-21
	A.11.4 Audits of Sensitive Data	A-21
	A.11.5 Recommended Audit Settings	A-21
	A.11.6 Best Practices for Querying the UNIFIED_AUDIT_TRAIL Data Dictionary View	A-22
	A.12 Addressing the CONNECT Role Change	A-23
	A.12.1 Why Was the CONNECT Role Changed?	A-23
	A.12.2 How the CONNNECT Role Change Affects Applications	A-24
	A.12.2.1 How the CONNECT Role Change Affects Database Upgrades	A-24
	A.12.2.2 How the CONNECT Role Change Affects Account Provisioning	A-24
	A.12.2.3 How the CONNECT Role Change Affects Applications Using New Databases	A-24
	A.12.3 How the CONNECT Role Change Affects Users	A-25
	A.12.3.1 How the CONNECT Role Change Affects General Users	A-25
	A.12.3.2 How the CONNECT Role Change Affects Application Developers	A-25
	A.12.3.3 How the CONNECT Role Change Affects Client Server Applications	A-25
	A.12.4 Approaches to Addressing the CONNECT Role Change	A-26
	A.12.4.1 Creating a New Database Role	A-26
	A.12.4.2 Restoring the CONNECT Privilege	A-27
	A.12.4.3 Data Dictionary View to Show CONNECT Grantees	A-28
	A.12.4.4 Least Privilege Analysis Studies	A-28
В	Data Encryption and Integrity Parameters	
	B.1 About Using sqlnet.ora for Data Encryption and Integrity	B-1
	B.2 Sample sqlnet.ora File	B-1
	B.3 Data Encryption and Integrity Parameters	B-3
	B.3.1 About the Data Encryption and Integrity Parameters	B-3
	B.3.2 SQLNET.ENCRYPTION_SERVER	B-4
	B.3.3 SQLNET.ENCRYPTION_CLIENT	B-5
	B.3.4 SQLNET.CRYPTO_CHECKSUM_SERVER	B-5
	B.3.5 SQLNET.CRYPTO_CHECKSUM_CLIENT	B-6
	B.3.6 SQLNET.ENCRYPTION_TYPES_SERVER	B-6



B.3.8 B.3.9	SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	
Kerbero	s, TLS, and RADIUS Authentication Parameters	
C.1 Para	meters for Clients and Servers Using Kerberos Authentication	
C.2 Para	meters for Clients and Servers Using Transport Layer Security	
C.2.1	Ways to Configure a Parameter for Transport Layer Security	
C.2.2	Transport Layer Security Authentication Parameters for Clients and Servers	
C.2.3	Cipher Suite Parameters for Transport Layer Security	
C.2.4	Supported Transport Layer Security Cipher Suites	
C.2.5	Transport Layer Security Version Parameters	
C.2.6	Transport Layer Security Client Authentication Parameters	
C.2.7	Transport Layer Security X.509 Server Match Parameters	
C.	2.7.1 SSL_SERVER_DN_MATCH	
C.	2.7.2 SSL_SERVER_CERT_DN	
C.2.8	Oracle Wallet Location	
C.3 Para	meters for Clients and Servers Using RADIUS Authentication	
C.3.1	sqlnet.ora File Parameters	
C.	3.1.1 SQLNET.AUTHENTICATION_SERVICES	
C.	3.1.2 SQLNET.RADIUS_ALTERNATE	
C.	3.1.3 SQLNET.RADIUS_ALTERNATE_PORT	
C.	3.1.4 SQLNET.RADIUS_ALTERNATE_TIMEOUT	
C.	3.1.5 SQLNET.RADIUS_ALTERNATE_RETRIES	
C.	3.1.6 SQLNET.RADIUS_AUTHENTICATION	
C.	3.1.7 SQLNET.RADIUS_AUTHENTICATION_INTERFACE	
C.	3.1.8 SQLNET.RADIUS_AUTHENTICATION_PORT	
C.	3.1.9 SQLNET.RADIUS_AUTHENTICATION_TIMEOUT	
C.	3.1.10 SQLNET.RADIUS_AUTHENTICATION_RETRIES	
C.	3.1.11 SQLNET.RADIUS_CHALLENGE_RESPONSE	
C.	3.1.12 SQLNET.RADIUS_CHALLENGE_KEYWORD	
C.	3.1.13 SQLNET.RADIUS_CLASSPATH	
C.	3.1.14 SQLNET.RADIUS_SECRET	
C.	3.1.15 SQLNET.RADIUS_SEND_ACCOUNTING	
C.3.2	Minimum RADIUS Parameters	
C.3.3	Initialization File Parameter for RADIUS	
	ing Authentication Devices Using RADIUS	
	the DADIIIC Challenge Decisions Used Interfere	
D.1 Abo	ut the RADIUS Challenge-Response User Interface	



E Oracle Database FIPS 140	0-2 Settings
----------------------------	--------------

E.1 About the	Oracle Database FIPS 140-2 Settings	E-1
E.2 Configurin	g FIPS 140-2 for Transparent Data Encryption and DBMS_CRYPTO	E-2
E.3 Configurat	tion of FIPS 140-2 for Transport Layer Security	E-3
	figuring the SSLFIPS_140 and SSLFIPS_LIB Parameters for Transport	
•	er Security	E-3
E.3.2 App	roved TLS Cipher Suites for FIPS 140-2	E-4
_	tion of FIPS 140-2 for Native Network Encryption	E-5
	ut Configuration of FIPS 140-2 for Native Network Encryption	E-5
	figuring the FIPS_140 Parameter for Native Network Encryption	E-5
E.5 Postinstall	ation Checks for FIPS 140-2	E-6
E.6 Verifying F	FIPS 140-2 Connections	E-6
E.6.1 Verif	fying FIPS 140-2 Connections for Transport Layer Security	E-6
E.6.2 Verit	fying FIPS 140-2 Connections for Network Native Encryption	E-7
	fying FIPS 140-2 Connections for Transparent Data Encryption and	
DBM	MS_CRYPTO	E-7
Managing P	Public Key Infrastructure (PKI) Elements	
F.1 Uses of the	e orapki Utility	F-1
F.2 orapki Utili	ty Syntax	F-2
F.3 Creating S	signed Certificates for Testing Purposes	F-2
F.4 Viewing a	Certificate	F-3
F.5 Importing a	a User-Supplied or Trusted Certificate into an Oracle Wallet	F-3
F.6 Controlling	MD5 and SHA-1 Certificate Use	F-3
F.7 Managing	Oracle Wallets with orapki Utility	F-4
F.7.1 Abou	ut Managing Wallets with orapki	F-4
F.7.2 Crea	ating, Viewing, and Modifying Wallets with orapki	F-4
F.7.2.1	Creating a PKCS#12 Wallet	F-5
F.7.2.2	Using OpenSSL to Create a PKCS#12 Wallet That Has a Certificate Chain	F-5
F.7.2.3	Creating an Auto-Login Wallet	F-6
F.7.2.4	Creating an Auto-Login Wallet That Is Associated with a PKCS#12 Wallet	F-6
F.7.2.5	Creating an Auto-Login Wallet That Is Local to the Computer and User Who Created It	F-7
F.7.2.6	Viewing a Wallet	F-7
F.7.2.7	Modifying the Password for a Wallet	F-7
F.7.2.8	Converting an Oracle Wallet to Use the AES256 Algorithm	F-7
	ng Certificates and Certificate Requests to Oracle Wallets with orapki	F-8
F.7.3.1	Adding a Certificate Request to an Oracle Wallet	F-8
F.7.3.2	Adding a Trusted Certificate to an Oracle Wallet	F-9



F

F.7.3.	3 Adding a Root Certificate to an Oracle Wallet	F-9
F.7.3.	4 Adding a User Certificate to an Oracle Wallet	F-9
F.7.3.	Verifying Credentials on the Hardware Device That Uses a PKCS#11 Wallet	F-9
F.7.3.	6 Adding PKCS#11 Information to an Oracle Wallet	F-10
F.7.4 Ex	xporting Certificates and Certificate Requests from Oracle Wallets with orapki	F-10
8 Manage	ement of Certificate Revocation Lists (CRLs) with orapki Utility	F-10
9 orapki L	Jsage	F-11
F.9.1 Ex	xample: Wallet with a Self-Signed Certificate and Export of the Certificate	F-11
F.9.2 Ex	xample: Creating a Wallet and a User Certificate	F-12
10 orapki	Utility Commands Summary	F-13
F.10.1	orapki cert create	F-14
F.10.2	orapki cert display	F-14
F.10.3	orapki crl delete Command	F-14
F.10.4 (orapki crl display	F-15
F.10.5	orapki crl hash	F-15
F.10.6	orapki crl list	F-16
F.10.7	orapki crl upload	F-16
F.10.8	orapki wallet add	F-17
F.10.9	orapki wallet convert	F-17
F.10.10	orapki wallet create	F-18
F.10.11	orapki wallet display	F-18
F.10.12	orapki wallet export	F-19



List of Tables

2-1	Predefined Oracle Database Administrative User Accounts	2-32
2-2	Predefined Oracle Database Non-Administrative User Accounts	2-35
2-3	Default Sample Schema User Accounts	2-36
2-4	Data Dictionary Views That Display Information about Users and Profiles	2-37
3-1	Password-Specific Settings in the Default Profile	3-7
3-2	Parameters Controlling Reuse of a Previous Password	3-11
3-3	Password Rollover Time Limits	3-20
3-4	Effect of SQLNET.ALLOWED_LOGON_VERSION_SERVER on Password Version Generation	3-41
3-5	Data Dictionary Views That Describe User Authentication	3-92
4-1	Roles to Allow Access to SYS Schema Objects	4-12
4-2	Properties of Roles and Their Description	4-25
4-3	Oracle Database Predefined Roles	4-30
4-4	System Privileges for Named Types	4-66
4-5	Privileges for Object Tables	4-68
4-6	Data Dictionary Views That Display Privilege and Role Information	4-86
5-1	Data Dictionary Views That Display Privilege Analysis Information	5-27
8-1	Parameters to Directly Retrieve Tokens	8-33
10-1	Data Dictionary Views That Display Information about Access Control Lists	10-22
12-1	Features Affected by the One Big Application User Model	12-2
12-2	Expected Behaviors for extproc Process Authentication and Impersonation Settings	12-14
12-3	How Privileges Relate to Schema Objects	12-24
12-4	SQL Statements Permitted by Database Object Privileges	12-25
13-1	Types of Application Contexts	13-5
13-2	Setting the DBMS_SESSION.SET_CONTEXT username and client_id Parameters	13-33
13-3	Data Dictionary Views That Display Information about Application Contexts	13-55
14-1	DBMS_RLS Procedures	14-9
14-2	DBMS_RLS.ADD_POLICY Policy Types	14-26
14-3	Oracle Virtual Private Database in Different User Models	14-50
14-4	Data Dictionary Views That Display Information about VPD Policies	14-51
15-1	DBMS_RLS.ADD_POLICY Parameters Used for TSDP Policies	15-23
15-2	Unified Audit Policy Settings Used for TSDP Policies	15-30
15-3	Fine-Grained Audit Policy Settings Used for TSDP Policies	15-32
15-4	TDE Column Encryption ENCRYPT Settings Used for TSDP Policies	15-34
15-5	Transparent Sensitive Data Protection Views	15-35
16-1	Data Dictionary Views for Encrypted Data Dictionary Credentials	16-6
17-1	DBMS_CRYPTO Package Feature Summary	17-10



17-2	SHA Hash Algorithms	17-10
17-3	Encryption and Decryption Algorithms	17-11
17-4	Other Algorithms	17-11
17-5	Data Dictionary Views That Display Information about Encrypted Data	17-20
18-1	Comparison of Native Network Encryption and Transport Layer Security	18-3
18-2	Two Forms of Network Attacks	18-3
18-3	Encryption and Data Integrity Negotiations	18-10
18-4	Valid Encryption Algorithms	18-12
19-1	CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_LEVEL Attributes	19-5
19-2	CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_TYPES Attributes	19-5
19-3	CONNECTION_PROPERTY_THIN_NET_CHECKSUM_LEVEL Attributes	19-6
19-4	CONNECTION_PROPERTY_THIN_NET_CHECKSUM_TYPES Attributes	19-6
19-5	CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_SERVICES Attributes	19-6
20-1	Authentication Methods and System Requirements	20-7
21-1	Kerberos Adapter Command-Line Utilities	21-2
21-2	Common Security Administrator/DBA Configuration and Administrative Tasks	21-4
22-1	Kerberos-Specific sqlnet.ora Parameters	22-9
22-2	Options for the okinit Utility	22-12
22-3	Options for the oklist Utility	22-13
22-4	Options for the okdstry Utility	22-14
22-5	okcreate Utility Options for Automatic Keytab Creation	22-15
23-1	Transport Layer Security Cipher Suites	23-17
23-2	SSL_DH Transport Layer Security Cipher Suites	23-18
24-1	RADIUS Authentication Components	24-3
26-1	Differences Between Mixed Mode Audting and Pure Unified Auditing	26-6
27-1	Administrative Users and Administrative Privileges	27-12
27-2	Object-Level Standard Database Action Audit Option	27-14
27-3	Auditing Behavior for READ ANY TABLE and SELECT ANY TABLE	27-21
27-4	Oracle Database Real Application Security User, Privilege, and Role Audit Events	27-35
27-5	Oracle Database Real Application Security Security Class and ACL Audit Events	27-36
27-6	Oracle Database Real Application Security Session Audit Events	27-37
27-7	Oracle Database Real Application Security ALL Events	27-38
27-8	Oracle Recovery Manager Columns in UNIFIED_AUDIT_TRAIL View	27-40
27-9	Oracle Database Vault Realm Audit Events	27-44
27-10	Oracle Database Vault Rule Set and Rule Audit Events	27-44
27-11	Oracle Database Vault Command Rule Audit Events	27-45
27-12	Oracle Database Vault Factor Audit Events	27-46



27-13	Oracle Database Vault Secure Application Role Audit Events	27-47
27-14	Oracle Database Vault Oracle Label Security Audit Events	27-48
27-15	Oracle Database Vault Oracle Data Pump Audit Events	27-48
27-16	Oracle Database Vault Enable and Disable Audit Events	27-48
27-17	Oracle Label Security Audit Events	27-52
27-18	Oracle Data Mining Audit Events	27-57
27-19	How Audit Policies Apply to the CDB Root, Application Root, and Individual PDBs	27-71
27-20	Views That Display Information about Audited Activities	27-111
28-1	Audit Record Field Names for SYSLOG and the Windows Event Viewer	28-5
28-2	Views That Display Information about Audit Trail Management Settings	28-25
A-1	Columns and Contents for DBA_CONNECT_ROLE_GRANTEES	A-28
B-1	Algorithm Type Selection	B-4
B-2	SQLNET.ENCRYPTION_SERVER Parameter Attributes	B-4
B-3	SQLNET.ENCRYPTION_CLIENT Parameter Attributes	B-5
B-4	SQLNET.CRYPTO_CHECKSUM_SERVER Parameter Attributes	B-5
B-5	SQLNET.CRYPTO_CHECKSUM_CLIENT Parameter Attributes	B-6
B-6	SQLNET.ENCRYPTION_TYPES_SERVER Parameter Attributes	B-6
B-7	SQLNET.ENCRYPTION_TYPES_CLIENT Parameter Attributes	B-7
B-8	SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER Parameter Attributes	B-8
B-9	SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT Parameter Attributes	B-8
C-1	Kerberos Authentication Parameters	C-1
C-2	TLS Authentication Parameters for Clients and Servers	C-3
C-3	Cipher Suite Parameters for Transport Layer Security	C-3
C-4	Transport Layer Security Version Parameters	C-4
C-5	Transport Layer Security Client Authentication Parameters	C-5
C-6	SSL_SERVER_DN_MATCH Parameter	C-6
C-7	SSL_SERVER_CERT_DN Parameter	C-7
C-8	Wallet Location Parameters	C-7
C-9	SQLNET.AUTHENTICATION_SERVICES Parameter Attributes	C-9
C-10	SQLNET.RADIUS_ALTERNATE Parameter Attributes	C-9
C-11	SQLNET.RADIUS_ALTERNATE_PORT Parameter Attributes	C-10
C-12	SQLNET.RADIUS_ALTERNATE_TIMEOUT Parameter Attributes	C-10
C-13	SQLNET.RADIUS_ALTERNATE_RETRIES Parameter Attributes	C-10
C-14	SQLNET.RADIUS_AUTHENTICATION Parameter Attributes	C-10
C-15	SQLNET.RADIUS_AUTHENTICATION_INTERFACE Parameter Attributes	C-11
C-16	SQLNET.RADIUS_AUTHENTICATION_PORT Parameter Attributes	C-11
C-17	SQLNET.RADIUS_AUTHENTICATION_TIMEOUT Parameter Attributes	C-11



C-18	SQLNET.RADIUS_AUTHENTICATION_RETRIES Parameter Attributes	C-12
C-19	SQLNET.RADIUS_CHALLENGE_RESPONSE Parameter Attributes	C-12
C-20	SQLNET.RADIUS_CHALLENGE_KEYWORD Parameter Attributes	C-12
C-21	SQLNET.RADIUS_CLASSPATH Parameter Attributes	C-13
C-22	SQLNET.RADIUS_SECRET Parameter Attributes	C-13
C-23	SQLNET.RADIUS_SEND_ACCOUNTING Parameter Attributes	C-13
E-1	How the DBFIPS_140 Initialization Parameter Affects Platforms	E-2
G-1	Availability of Unified Auditing Features Before and After Migration	G-1



Preface

Welcome to *Oracle Database Security Guide*. This guide describes how you can configure security for Oracle Database by using the default database features.

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

Oracle Database Security Guide is intended for database administrators (DBAs), security administrators, application developers, and others tasked with performing the following operations securely and efficiently.

It covers these areas:

- Designing and implementing security policies to protect the data of an organization, users, and applications from accidental, inappropriate, or unauthorized actions
- Creating and enforcing policies and practices of auditing and accountability for inappropriate or unauthorized actions
- Creating, maintaining, and terminating user accounts, passwords, roles, and privileges
- Developing applications that provide desired services securely in a variety of computational models, leveraging database and directory services to maximize both efficiency and ease of use

To use this document, you need a basic understanding of how and why a database is used, and basic familiarity with SQL.

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 customer access to and use of Oracle support services will be pursuant to the terms and conditions specified in their Oracle order for the applicable services.

Related Documents

For more security-related information, see these Oracle resources:

Oracle Database Advanced Security Guide



- Oracle Database Vault Administrator's Guide
- Oracle Label Security Administrator's Guide
- Oracle Key Vault documentation library
- Audit Vault and Database Firewall documentation library
- Oracle Data Masking and Subsetting documentation library
- Oracle Data Safe documentation library
- Oracle Database Security Assessment Tool
- Oracle Database PL/SQL Packages and Types Reference
- Oracle Database Reference
- Oracle Database SQL Language Reference
- Oracle Database Net Services Reference
- Oracle Database Administrator's Guide
- Oracle Database Concepts
- Oracle Multitenant Administrator's Guide

Many of the examples in this guide use the sample schemas of the seed PDB, which you can create when you install Oracle Database. See *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them yourself.

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 Security Guide

This preface contains:

- Changes in Oracle Database Security 19c
- Updates to Oracle Database Security 19c

Changes in Oracle Database Security 19c

Oracle Database Security Guide for Oracle Database 19c has new security features.

- Signature-Based Security for LOB Locators
 Starting with this release, you can configure signature-based security for large object (LOB) locators.
- Default User Accounts Now Schema Only
 Using the schema only account feature from Oracle Database release 18c, most of the
 Oracle Database supplied schemas (users) now have their passwords removed to prevent
 users from authenticating to these accounts.
- Privilege Analysis Documentation Moved to Oracle Database Security Guide
 The documentation for privilege analysis has moved from Oracle Database Vault
 Administrator's Guide to Oracle Database Security Guide.
- Ability to Grant or Revoke Administrative Privileges to and from Schema-Only Accounts
 Administrative privileges such as SYSOPER and SYSBACKUP can now be granted to schema-only (passwordless) accounts.
- Automatic Support for Both SASL and Non-SASL Active Directory Connections
 Starting with this release, both Simple Authentication and Security Layer (SASL) and
 Transport Layer Security (TLS) binds are supported for Microsoft Active Directory
 connections.
- Support for Oracle Native Encryption and TLS Authentication for Different Users Concurrently
 - In previous releases, Oracle Database prevented the use of both Oracle native encryption (also called Advanced Networking Option (ANO) encryption) and Transport Layer Security (TLS) authentication together.
- Support for Host Name-Based Partial DN Matching for Matching for Server Certificates
 This new support for partial DN matching adds the ability for the client to further verify the
 server certificate.
- Ability to Audit Only Top-Level SQL Statements
 The unified auditing top-level statements feature enables you to audit top level user (or, direct user) activities in the database but without collecting indirect user activity audit data.
- Improved Read Performance for the Unified Audit Trial
 The AUDSYS.AUD\$UNIFIED system table, which stores the unified audit trail records, has been redesigned to use partition pruning to improve read performance.



- SYSLOG Destination for Common Unified Audit Policies
 Available with Oracle Database release 19.9, certain predefined columns of unified audit
 - records from common unified audit policies can be written to the UNIX SYSLOG destination.
- PDB_GUID as Audit Record Field Name for SYSLOG and the Windows Event Viewer The audit record fields for SYSLOG and the Windows Event Viewer now have a new field, PDB GUID, to identify the pluggable database associated with a unified audit trail record.

Signature-Based Security for LOB Locators

Starting with this release, you can configure signature-based security for large object (LOB) locators.

This feature strengthens the security of Oracle Database LOBs, particularly when instances of LOB data types (CLOB and BLOB) are used in distributed environments.

LOB signature keys can be in both multitenant PDBs or in standalone, non-multitenant databases. You can enable the encryption of the LOB signature key credentials by executing the ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS SQL statement; otherwise, the credentials are stored in obfuscated format. If you choose to store the LOB signature key in encrypted format, then the database or PDB must have an open TDE keystore.

Related Topics

Securing LOBs with LOB Locator Signatures
 You can secure large objects (LOB) by regenerating their LOB locator signatures.

Default User Accounts Now Schema Only

Using the schema only account feature from Oracle Database release 18c, most of the Oracle Database supplied schemas (users) now have their passwords removed to prevent users from authenticating to these accounts.

This enhancement does not affect the sample schemas. Sample schemas are still installed with their default passwords.

For the default schemas that are schema only, administrators can still alter these accounts with passwords if they need to authenticate to the schema, but Oracle recommends changing the schemas back to a schema-only account afterward.

The benefit of this feature is that administrators no longer have to periodically rotate the passwords for these Oracle Database-provided schemas. This feature also reduces the security risk of attackers using default passwords to hack into these accounts.

Related Topics

- Predefined Schema User Accounts Provided by Oracle Database
 The Oracle Database installation process creates predefined administrative, non-administrative, and sample schema user accounts in the database.
- Schema-Only Accounts
 You can create schema-only accounts, that is, the schema user has no password.



Privilege Analysis Documentation Moved to Oracle Database Security Guide

The documentation for privilege analysis has moved from Oracle Database Vault Administrator's Guide to *Oracle Database Security Guide*.

See Oracle Database Licensing Information User Manual for privilege analysis licensing information.

Related Topics

- Performing Privilege Analysis to Identify Privilege Use
 Privilege analysis dynamically analyzes the privileges and roles that users use and do not
 use.
- Oracle Database Licensing Information User Manual

Ability to Grant or Revoke Administrative Privileges to and from Schema-Only Accounts

Administrative privileges such as SYSOPER and SYSBACKUP can now be granted to schema-only (passwordless) accounts.

Existing user accounts (active, rarely accessed, and unused users) that are currently granted administrative privileges can be altered to be schema-only accounts. This enhancement prevents administrators from having to manage the passwords of these accounts.

Related Topics

About Schema-Only Accounts
 A schema-only account cannot log in to the database but can proxy in a single session proxy.

Automatic Support for Both SASL and Non-SASL Active Directory Connections

Starting with this release, both Simple Authentication and Security Layer (SASL) and Transport Layer Security (TLS) binds are supported for Microsoft Active Directory connections.

For centrally managed users, the Oracle database will initially try to connect to Active Directory using SASL bind. If the Active Directory server rejects the SASL bind connection, then the Oracle database will automatically attempt the connection again without SASL bind but still secured with TLS.

The Active Directory administrator is responsible for configuring the connection parameters for Active Directory server, but does not need to configure the database to match this new Active Directory connection enhancement. The database will automatically adjust from using SASL to not using SASL bind.

Related Topics

About Configuring the Oracle Database-Microsoft Active Directory Connection
Before you configure this connection, you must have Microsoft Active Directory installed
and configured.



Support for Oracle Native Encryption and TLS Authentication for Different Users Concurrently

In previous releases, Oracle Database prevented the use of both Oracle native encryption (also called Advanced Networking Option (ANO) encryption) and Transport Layer Security (TLS) authentication together.

Starting with this release, you can set a new parameter,

SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS, to TRUE to ignore the SQLNET.ENCRYPTION_CLIENT or SQLNET.ENCRYPTION_SERVER when there is a conflict between the use of a TCPS client and either of these two parameters are set to required.

Related Topics

 Enabling Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently

Depending on the SQLNET.ENCRYPTION_CLIENT and SQLNET.ENCRYPTION_SERVER settings, you can configure Oracle Database to allow both Oracle native encryption and SSL authentication for different users concurrently.

Support for Host Name-Based Partial DN Matching for Matching for Server Certificates

This new support for partial DN matching adds the ability for the client to further verify the server certificate.

The earlier ability to perform a full DN match with the server certificate during the Transport Layer Security (TLS) handshake is still supported. The client supports both full and partial DN matching. If the server DN matching is enabled, then partial DN matching is the default.

Allowing partial and full DN matching for certificate verification enables more flexibility based on how the certificates were created.

Related Topics

About Configuring the Server DN Matching and Using TCP/IP with TLS on the Client
In addition to validating the server certificate's certificate chain, you can perform an extra
check through server DN matching.

Ability to Audit Only Top-Level SQL Statements

The unified auditing top-level statements feature enables you to audit top level user (or, direct user) activities in the database but without collecting indirect user activity audit data.

You can use this feature to audit only the top-level user directly issued events, without the overhead of indirect SQL statements. Top-level statements are SQL statements that users directly issue. These statements can be important for both security and compliance. SQL statements run from within PL/SQL procedures or functions are not considered top level, so they may be less relevant for auditing purposes.

Related Topics

Auditing Only Top-Level Statements
 You can audit top-level SQL or PL/SQL statements to limit the volume of audit records.



Improved Read Performance for the Unified Audit Trial

The AUDSYS.AUD\$UNIFIED system table, which stores the unified audit trail records, has been redesigned to use partition pruning to improve read performance.

This redesign entailed the addition of a new column to the AUDSYS.AUD\$UNIFIED table. The UNIFIED_AUDIT_TRAIL data dictionary view, which enables you to query the AUDSYS.AUD\$UNIFIED table audit records, now has the EVENT_TIMESTAMP_UTC column to correspond with the new AUDSYS.AUD\$UNIFIED table column. As part of this enhancement, the data type of the EVENT_TIMESTAMP column in the GV\$UNIFIED_AUDIT_TRAIL view has changed TIMESTAMP (6).

Oracle recommends that when you query the <code>UNIFIED_AUDIT_TRAIL</code> view, to include the <code>EVENT_TIMESTAMP_UTC</code> column in the <code>WHERE</code> clause to achieve partitioning pruning.

Related Topics

Best Practices for Querying the UNIFIED_AUDIT_TRAIL Data Dictionary View
 To get the best results from querying the UNIFIED_AUDIT_TRAIL data dictionary view, you should follow these guidelines.

SYSLOG Destination for Common Unified Audit Policies

Available with Oracle Database release 19.9, certain predefined columns of unified audit records from common unified audit policies can be written to the UNIX SYSLOG destination.

To enable this feature, you set <code>UNIFIED_AUDIT_COMMON_SYSTEMLOG</code>, a new CDB level <code>init.ora</code> parameter. This enhancement enables all audit records from common unified audit policies to be consolidated into a single destination.

This feature is available only on UNIX platforms, not Windows.

Related Topics

Enabling SYSLOG and Windows Event Viewer Captures for the Unified Audit Trail
You can write a subset of unified audit trail records to the UNIX SYSLOG or to the
Windows Event Viewer.

PDB_GUID as Audit Record Field Name for SYSLOG and the Windows Event Viewer

The audit record fields for SYSLOG and the Windows Event Viewer now have a new field, PDB GUID, to identify the pluggable database associated with a unified audit trail record.

In a multitenant database deployment, the pluggable database that generated a unified audit trail record must be identified in the audit trail. Starting with this release, the <code>SYSLOG</code> and Windows Event Viewer will have a new field, <code>PDB_GUID</code>, to capture this information. The data type is <code>VARCHAR2</code>.

Related Topics

About Writing the Unified Audit Trail Records to SYSLOG or the Windows Event Viewer
With this feature, you can copy some of the key unified audit fields to SYSLOG or the
Windows Event Viewer.



Updates to Oracle Database Security 19c

Oracle Database release 19c has several updates from the last update of release 19c.

- Authenticating and Authorizing IAM Users to Oracle Autonomous Database on Dedicated Exadata Infrastructure
 - Available for Oracle Database release 19.14, users now can authenticate and authorize IAM users to Oracle Autonomous Database on Dedicated Exadata Infrastructure.
- Enhancements for Identity and Access Management Integration with Oracle Database Environments
 - Available for Oracle Database release 19.16 are enhancements to the integration of Identity and Access Management (IAM) users with Oracle Database Environments.
- Identity and Access Management Integration with Oracle Autonomous Cloud Databases
 Available for Oracle Database release 19.13, Identity and Access Management (IAM)
 users can log in to an Oracle Autonomous Database Serverless using either password or
 token-based authentication.
- Database Integration Support for Non-Default Domains for Identity and Access
 Management with Identity Domains
 Starting with this release, Oracle Database supports non-default domains in tenancies with Identity and Access Management (IAM) with Identity Domains.
- Microsoft Azure Active Directory Integration with Additional Oracle Database Environments
 Including On-Premises
 Available for Oracle Database release 10.16. Microsoft Azure Active Directory (Azure AD)

Available for Oracle Database release 19.16, Microsoft Azure Active Directory (Azure AD) users can log in to additional Oracle Database environments with their Azure AD <code>OAuth2</code> access token.

- Microsoft Azure Active Directory Integration with Oracle Cloud Infrastructure Autonomous Databases
 - Available for Oracle Autonomous Database in June, 2022, Microsoft Azure Active Directory (Azure AD) users can log in to Oracle Cloud Infrastructure (OCI) Autonomous Database with their Azure AD OAuth2 access token.
- Gradual Database Password Rollover for Applications
 Available for Oracle Database release 19.12, an application can change its database passwords without an administrator having to schedule downtime.
- Ability to Use Multiple Kerberos Principals with a Single Database Client
 Available for Oracle Database release 19.10, when you configure Kerberos authentication
 for an Oracle Database client, you can specify multiple Kerberos principals with a single
 Oracle Database client.
- Updated Support for Micro Edition Suite (MES) for FIPS 140.2
 Available for Oracle Database release 19.10, Oracle Database supports Micro Edition Suite (MES) version 4.5 for FIPS 140.2.
- Support for DBMS_CRYPTO Asymmetric Key Operations
 Available for Oracle Database release 19.9, the DBMS_CRYPTO PL/SQL package supports
 asymmetric key operations, in addition to the existing support for symmetric key
 operations.
- SYSLOG Destination for Common Unified Audit Policies
 Available with Oracle Database release 19.9, certain predefined columns of unified audit records from common unified audit policies can be written to the UNIX SYSLOG destination.



- Security Update for Native Encryption
 - Oracle provides a patch that you can download to address necessary security enhancements that affect native network encryption environments in Oracle Database release 11.2 and later.
- Ability to Configure Transport Layer Security Connections without Client Wallets
 Available in Oracle Database release 19.14, an Oracle Database client will not be required
 to provide a wallet to hold well-known CA root certificates if they are available elsewhere in
 the local system.
- SSL_ALLOW_WEAK_DN_MATCH Parameter to Control the Behavior of SSL_SERVER_DN_MATCH

Available with Oracle Database release 19.23, you can use the SSL_ALLOW_WEAK_DN_MATCH parameter to control how SSL_SERVER_DN_MATCH allows the service name for partial distinguished name matching and to only check the database server certificate.

Authenticating and Authorizing IAM Users to Oracle Autonomous Database on Dedicated Exadata Infrastructure

Available for Oracle Database release 19.14, users now can authenticate and authorize IAM users to Oracle Autonomous Database on Dedicated Exadata Infrastructure.

Additional enhancements are as follows:

- Applications can now connect to an Autonomous Database instance by using end-user, instance, and resource principals.
- IAM users can now proxy to an Autonomous Database by using a database user schema.
- Database links are supported for IAM connections.

Related Topics

Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
 Identity and Access Management (IAM) users can be configured to connect to an Oracle Database as a service (Oracle DBaaS) instance.

Enhancements for Identity and Access Management Integration with Oracle Database Environments

Available for Oracle Database release 19.16 are enhancements to the integration of Identity and Access Management (IAM) users with Oracle Database Environments.

- Additional Oracle Database environments: The full list of supported Oracle Database environments is as follows:
 - Oracle Autonomous Database on Dedicated Exadata Infrastructure
 - Oracle Autonomous Database Serverless
 - Oracle Base Database Service
- Ability to use the IAM user name and password to retrieve an IAM token: Retrieving a
 token using an IAM user name and password or secure external password store (SEPS) is
 more secure than using the password verifier method of database access.



Related Topics

Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
 Identity and Access Management (IAM) users can be configured to connect to an Oracle
 Database as a service (Oracle DBaaS) instance.

Identity and Access Management Integration with Oracle Autonomous Cloud Databases

Available for Oracle Database release 19.13, Identity and Access Management (IAM) users can log in to an Oracle Autonomous Database Serverless using either password or token-based authentication.

An IAM ADMIN user can configure both the authentication and authorization of IAM users and IAM groups. The IAM user can log in to the Oracle Autonomous Database using tools such as SQL*Plus or SQLcl.

This enhancement provides the security advantages of both IAM and Oracle Database. For example, the Oracle Database gradual password rollover feature can be used in this configuration and update the application passwords without downtime.

Related Topics

Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
 Identity and Access Management (IAM) users can be configured to connect to an Oracle
 Database as a service (Oracle DBaaS) instance.

Database Integration Support for Non-Default Domains for Identity and Access Management with Identity Domains

Starting with this release, Oracle Database supports non-default domains in tenancies with Identity and Access Management (IAM) with Identity Domains.

The following releases are supported:

- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure

This update allows IAM users in non-default IAM domains to access the database with IAM database password verifiers or IAM access tokens. IAM users in default domains are already supported.

In previous releases, the IAM integration only worked with users and groups from the default domain, and did not support users and groups from custom, non-default domains.

Related Topics

Authenticating and Authorizing IAM Users for Oracle DBaaS Databases
 Identity and Access Management (IAM) users can be configured to connect to an Oracle
 Database as a service (Oracle DBaaS) instance.



Microsoft Azure Active Directory Integration with Additional Oracle Database Environments Including On-Premises

Available for Oracle Database release 19.16, Microsoft Azure Active Directory (Azure AD) users can log in to additional Oracle Database environments with their Azure AD <code>OAuth2</code> access token.

The previous release supported Azure AD integration support for Oracle Cloud Infrastructure (OCI) Autonomous Databases. This release has expanded Azure AD integration support to onpremises Oracle Database release 19.16 and later, but not for Oracle Database 21c.

You can use Azure AD <code>OAuth2</code> tokens to access the database. Azure AD users can access the database directly using their Azure AD token, and applications can use their service tokens to access the database.

Related Topics

 Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases

An Oracle Database can be configured for Microsoft Azure AD users to connect using single-sign on.

Microsoft Azure Active Directory Integration with Oracle Cloud Infrastructure Autonomous Databases

Available for Oracle Autonomous Database in June, 2022, Microsoft Azure Active Directory (Azure AD) users can log in to Oracle Cloud Infrastructure (OCI) Autonomous Database with their Azure AD <code>OAuth2</code> access token.

OCI Oracle Autonomous Database now can accept Azure AD <code>OAuth2</code> tokens to access the database. Azure AD users can access the database directly using their Azure AD token, and applications can use their service tokens to access the database.

You can use Azure AD <code>OAuth2</code> tokens to access the database. Azure AD users can access the database directly using their Azure AD token, and applications can use their service tokens to access the database.

Related Topics

 Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases

An Oracle Database can be configured for Microsoft Azure AD users to connect using single-sign on.

Gradual Database Password Rollover for Applications

Available for Oracle Database release 19.12, an application can change its database passwords without an administrator having to schedule downtime.

To accomplish this, a database administrator can associate a profile having a non-zero limit for the PASSWORD_ROLLOVER_TIME password profile parameter, new with this release, with an application schema. This allows the database password of the application user to be altered while allowing the older password to remain valid for the time specified by the PASSWORD_ROLLOVER_TIME limit. During the rollover period of time, the application instance can



use either the old password or the new password to connect to the database server. When the rollover time expires, only the new password is allowed.

Before this enhancement, an administrator normally took the application down when the application database password was being rotated. This is because the password update requires changes on both the database and the application side. With the gradual database password rollover enhancement, the application can continue to use the older password until the new password is configured in the application.

In addition to the new clause PASSWORD_ROLLOVER_TIME in the CREATE PROFILE and ALTER PROFILE statements, the ALTER USER statement has a new clause, EXPIRE PASSWORD ROLLOVER PERIOD. The ACCOUNT_STATUS column of the DBA_USERS and USER_USERS data dictionary views have several new statuses indicating values to indicate rollover status.

Related Topics

Managing Gradual Database Password Rollover for Applications
 A gradual database password rollover enables the database password of an application to be updated while avoiding application downtime while the new password is propagated to application clients, by allowing the older password to remain valid for a specified period.

Ability to Use Multiple Kerberos Principals with a Single Database Client

Available for Oracle Database release 19.10, when you configure Kerberos authentication for an Oracle Database client, you can specify multiple Kerberos principals with a single Oracle Database client.

To enable this functionality, you will need to create a separate credential cache for each user in the client and then use the connect string to specify the user.

In previous releases, you were restricted to one Kerberos principal for each Oracle Database client.

Related Topics

- Step 1C: Optionally, Specify Additional Kerberos Principals Using this this thin the state of th
- Oracle Database Net Services Reference

Updated Support for Micro Edition Suite (MES) for FIPS 140.2

Available for Oracle Database release 19.10, Oracle Database supports Micro Edition Suite (MES) version 4.5 for FIPS 140.2.

The Micro Edition Suite (MES) version 4.5 updates include four new CVEs in the RSA BSAFE MES library, support for the rules that FIPS 140.2 requires, and access to the updated NZ/ZT library from the Crypto Foundation.

This enhancement enables the Oracle Database FIPS 140.2 configuration to benefit from new features and security improvements available from the latest RSA BSAFE MES library.

Related Topics

Configuring FIPS 140-2 for Transparent Data Encryption and DBMS_CRYPTO
 The DBFIPS 140 initialization parameter configures FIPS mode.



Support for DBMS_CRYPTO Asymmetric Key Operations

Available for Oracle Database release 19.9, the DBMS_CRYPTO PL/SQL package supports asymmetric key operations, in addition to the existing support for symmetric key operations.

To implement the support for asymmetric key operations, the following procedures have been added to the $\tt DBMS$ CRYPTO package:

- PKENCRYPT
- PKDECRYPT
- SIGN
- VERIFY

Related Topics

- Asymmetric Key Operations with the DBMS_CRYPTO Package
 The DBMS_CRYPTO package provides four functions that enable you to perform asymmetric key operations for encryption, decryption, signing, and verification.
- Oracle Database PL/SQL Packages and Types Reference

SYSLOG Destination for Common Unified Audit Policies

Available with Oracle Database release 19.9, certain predefined columns of unified audit records from common unified audit policies can be written to the UNIX SYSLOG destination.

To enable this feature, you set <code>UNIFIED_AUDIT_COMMON_SYSTEMLOG</code>, a new CDB level <code>init.ora</code> parameter. This enhancement enables all audit records from common unified audit policies to be consolidated into a single destination.

This feature is available only on UNIX platforms, not Windows.

Related Topics

Enabling SYSLOG and Windows Event Viewer Captures for the Unified Audit Trail
You can write a subset of unified audit trail records to the UNIX SYSLOG or to the
Windows Event Viewer.

Security Update for Native Encryption

Oracle provides a patch that you can download to address necessary security enhancements that affect native network encryption environments in Oracle Database release 11.2 and later.

This patch is available in My Oracle Support note 2118136.2.

The supported algorithms that have been improved are as follows:

- Encryption algorithms: AES128, AES192 and AES256
- Checksumming algorithms: SHA1, SHA256, SHA384, and SHA512

Algorithms that are deprecated and should not be used are as follows:

- Encryption algorithms: DES, DES40, 3DES112, 3DES168, RC4_40, RC4_56, RC4_128, and RC4_256
- Checksumming algorithm: MD5



If your site requires the use of network native encryption, then you must download the patch that is described in My Oracle Support note 2118136.2. To enable a smooth transition for your Oracle Database installation, this patch provides two parameters that enable you to disable the weaker algorithms and start using the stronger algorithms. You will need to install this patch on both servers and clients in your Oracle Database installation.

An alternative to network native encryption is Transport Layer Security (TLS), which provides protection against person-in-the-middle attacks.

Related Topics

- Choosing Between Native Network Encryption and Transport Layer Security
 Oracle offers two ways to encrypt data over the network, native network encryption and
 Transport Layer Security (TLS).
- Improving Native Network Encryption Security
 Oracle provides a patch that will strengthen native network encryption security for both
 Oracle Database servers and clients.

Ability to Configure Transport Layer Security Connections without Client Wallets

Available in Oracle Database release 19.14, an Oracle Database client will not be required to provide a wallet to hold well-known CA root certificates if they are available elsewhere in the local system.

Transport Layer Security (TLS) encryption requires either one-way authentication or two-way authentication. In one-way authentication (the default), which is commonly used for HTTPS connections, the server certificate is verified using well-known root CA certificates that are already available in local systems. Starting in this release, you will no longer need to install and configure a wallet to hold a well-known root certificate if it is already available in the local system.

This enhancement greatly simplifies the Oracle Database client installation and the use of TLS protocol to encrypt Oracle Database client-server communications.

Related Topics

Transport Layer Security Connection without a Client Wallet
 A Transport Layer Security (TLS) connection that uses a common root certificate for the database server does not require a client wallet.

SSL_ALLOW_WEAK_DN_MATCH Parameter to Control the Behavior of SSL_SERVER_DN_MATCH

Available with Oracle Database release 19.23, you can use the SSL_ALLOW_WEAK_DN_MATCH parameter to control how SSL_SERVER_DN_MATCH allows the service name for partial distinguished name matching and to only check the database server certificate.

In this release, the behavior of the <code>SSL_SERVER_DN_MATCH</code> parameter has changed. Previously, only the database server certificate was checked for DN matching. With this release, the listener and server certificates are both checked. Also, the <code>SERVICE_NAME</code> setting in the connect string is not used to check during a partial DN match. The <code>HOSTNAME</code> setting can still be used for partial DN matching with the certificate DN and subject alternative name (SAN), on both the listener and server certificates. When set to <code>TRUE</code>, the <code>SSL_ALLOW_WEAK_DN_MATCH</code> parameter reverts <code>SSL_SERVER_DN_MATCH</code> to the older, pre-release <code>19c</code> behavior and enables DN matching



to only check the database server certificate (but not the listener) and enable the service name to be used for partial DN matching.

DN matching with both the listener and server certificates provides better security to ensure that the client is connecting to the correct database server. The service name setting is also removed from SSL_SERVER_DN_MATCH for better security and partial DN matching can still be performed with the host name connect string parameter with the certificate common names (CN) and subject alternative name (SAN) matching.

The SSL_ALLOW_WEAK_DN_MATCH, though new to this release, is marked as deprecated because it is considered a temporary solution.

Related Topics

 Use of the SSL_ALLOW_WEAK_DN_MATCH Parameter to Control SSL_SERVER_DN_MATCH

The SSL_ALLOW_WEAK_DN_MATCH parameter controls how the SSL_SERVER_DN_MATCH parameter allows the service name for partial distinguished name matching and check the database server certificate.



1

Introduction to Oracle Database Security

Oracle Database provides a rich set of default security features to manage user accounts, authentication, privileges, application security, encryption, network traffic, and auditing.

- About Oracle Database Security
 Use Oracle Database's security features to reduce risk and protect data from theft, destruction, or misuse.
- Additional Oracle Database Security Products
 In addition to the security resources that are available in a default database installation,
 Oracle Database provides several other database security products.

1.1 About Oracle Database Security

Use Oracle Database's security features to reduce risk and protect data from theft, destruction, or misuse.

A few popular areas to focus security efforts on include:

- User accounts. When you create user accounts, you can secure them in a variety of ways. You can also create password profiles to better secure password policies for your site. Managing Security for Oracle Database Users, describes how to manage user accounts.
- Authentication methods. Oracle Database provides several ways to configure
 authentication for users and database administrators. For example, you can authenticate
 users on the database level, from the operating system, and on the network. Configuring
 Authentication, describes how authentication in Oracle Database works. See also
 Configuring Centrally Managed Users with Microsoft Active Directory.
- Privileges and roles. You can use privileges and roles to restrict user access to data. The following chapters describe how to manage privileges and roles:
 - Configuring Privilege and Role Authorization
 - Performing Privilege Analysis to Identify Privilege Use
 - Managing Security for Definer's Rights and Invoker's Rights
 - Managing Fine-Grained Access in PL/SQL Packages and Types
 - Managing Security for a Multitenant Environment in Enterprise Manager
- Application security. The first step to creating a database application is to ensure that it is
 properly secure. Managing Security for Application Developers, discusses how to
 incorporate application security into your application security policies.
- User session information using application context. An application context is a namevalue pair that holds the session information. You can retrieve session information about a
 user, such as the user name or terminal, and restrict database and application access for
 that user based on this information. Using Application Contexts
 to Retrieve User Information, describes how to use application contexts.
- Database access on the row and column level using Virtual Private Database. A
 Virtual Private Database policy dynamically imbeds a WHERE predicate into SQL statements

the user issues. Using Oracle Virtual Private Database to Control Data Access, describes how to create and manage Virtual Private Database policies.

- Classify and protect data in different categories. You can find all table columns in a
 database that hold sensitive data (such as credit card or Social Security numbers), classify
 this data, and then create a policy that protects this data as a whole for a given class.
 Using Transparent Sensitive Data Protection, explains how to create Transparent
 Sensitive Data Protection policies.
- Network data encryption. Manually Encrypting Data, explains how to use the
 DBMS_CRYPTO PL/SQL package to encrypt data as it travels on the network to prevent
 unauthorized access to that data. You can configure native Oracle Net Services data
 encryption and integrity for both servers and clients, which are described in Configuring
 Oracle Database Native Network Encryption and Data Integrity.
- Thin JDBC client network configuration. You can configure thin Java Database
 Connectivity (JDBC) clients to securely connect to Oracle databases. Configuring the Thin
 JDBC Client Network, provides detailed information.
- Strong authentication. You can configure your databases to use strong authentication
 with Oracle authentication adapters that support various third-party authentication services,
 including SSL with digital certificates. Oracle Database provides the following strong
 authentication support:
 - Centralized authentication and single sign-on.
 - Kerberos
 - Remote Authentication Dial-in User Service (RADIUS)
 - Transport Layer Security (TLS) (formerly called Secure Sockets Layer)

The following chapters cover strong authentication:

- Introduction to Strong Authentication
- Strong Authentication Administration Tools
- Configuring Kerberos Authentication
- Configuring Transport Layer Security Authentication
- Configuring RADIUS Authentication
- Customizing the Use of Strong Authentication
- Auditing database activities. You can audit database activities in general terms, such as
 auditing all SQL statements, SQL privileges, schema objects, and network activity. Or, you
 can audit in a granular manner, such as when the IP addresses from outside the corporate
 network is being used. This chapter also explains how to purge the database audit trail.
 The following chapters describe how to configure and administer database auditing.
 - Introduction to Auditing
 - Configuring Audit Policies
 - Administering the Audit Trail

In addition, Keeping Your Oracle Database Secure, provides guidelines that you should follow when you secure your Oracle Database installation.

1.2 Additional Oracle Database Security Products

In addition to the security resources that are available in a default database installation, Oracle Database provides several other database security products.



These products are as follows:

- Oracle Advanced Security enables you to protect sensitive data by using Transparent Data Encryption and Oracle Data Redaction.
- Oracle Label Security applies classification labels to data, allowing you to filter user access to data at the row level.
- Oracle Database Vault provides fine-grained access control to your sensitive data, including protecting data from privileged users. For example, you can restrict database administrators from having access to employee information such as salaries.
- Oracle Data Safe enables you to analyze the sensitivity and risks of data in your Oracle databases, and based on these findings, create policies that mask sensitive data, create and monitor security controls, assess user security, and monitor user activity.
- Oracle Enterprise User Security enables you to manage user security at the enterprise level.
- Oracle Enterprise Manager Data Masking and Subsetting Pack can irreversibly replace
 the original sensitive data with fictitious data so that production data can be shared safely
 with IT developers or offshore business partners.
- Oracle Audit Vault and Database Firewall collects database audit data from sources such as Oracle Database audit trail tables, database operating system audit files, and database redo logs. Using Oracle Audit Vault and Database Firewall, you can create alerts on suspicious activities, and create reports on the history of privileged user changes, schema modifications, and even data-level access.
- Oracle Key Vault enables you to accelerate security and encryption deployments by centrally managing encryption keys, Oracle wallets, Java keystores, and credential files. It is optimized for Oracle wallets, Java keystores, and Oracle Advanced Security Transparent Data Encryption (TDE) master keys. Oracle Key Vault supports the OASIS KMIP standard. The full-stack, security-hardened software appliance uses Oracle Linux and Oracle Database technology for security, availability, and scalability, and can be deployed on your choice of compatible hardware.

In addition to these products, you can find the latest information about Oracle Database security, such as new products and important information about security patches and alerts, by visiting the Security Technology Center on Oracle Technology Network at

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



Part I

Managing User Authentication and Authorization

Part I describes how to manage user authentication and authorization.

Managing Security for Oracle Database Users

You can manage the security for Oracle Database users in many ways, such as enforcing restrictions on the way that passwords are created.

· Configuring Authentication

Authentication means to verify the identity of users or other entities that connect to the database.

Configuring Privilege and Role Authorization

Privilege and role authorization controls the permissions that users have to perform day-to-day tasks.

Performing Privilege Analysis to Identify Privilege Use

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

Configuring Centrally Managed Users with Microsoft Active Directory

Oracle Database can authenticate and authorize Microsoft Active Directory users with the database directly without intermediate directories or Oracle Enterprise User Security.

• Authenticating and Authorizing IAM Users for Oracle DBaaS Databases

Identity and Access Management (IAM) users can be configured to connect to an Oracle Database as a service (Oracle DBaaS) instance.

 Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases

An Oracle Database can be configured for Microsoft Azure AD users to connect using single-sign on.

Managing Security for Definer's Rights and Invoker's Rights

Invoker's rights and definer's rights have several security advantages when used to control access to privileges during user-defined procedure executions.

Managing Fine-Grained Access in PL/SQL Packages and Types

Oracle Database provides PL/SQL packages and types for fine-grained access to control access to external network services and wallets.

Managing Security for a Multitenant Environment in Enterprise Manager

You can manage common and local users and roles for a multitenant environment by using Oracle Enterprise Manager.



Managing Security for Oracle Database Users

You can manage the security for Oracle Database users in many ways, such as enforcing restrictions on the way that passwords are created.

About User Security

You can secure users accounts through strong passwords and by specifying special limits for the users.

Creating User Accounts

A user account can have restrictions such as profiles, a default role, and tablespace restrictions.

Altering User Accounts

The ALTER USER statement modifies user accounts, such their default tablespace or profile, or changing a user's password.

Configuring User Resource Limits

A resource limit defines the amount of system resources that are available for a user.

Dropping User Accounts

You can drop user accounts if the user is not in a session, and if the user has objects in the user's schema.

Predefined Schema User Accounts Provided by Oracle Database

The Oracle Database installation process creates predefined administrative, non-administrative, and sample schema user accounts in the database.

Database User and Profile Data Dictionary Views

Oracle Database provides a set of data dictionary views that provide information about the settings that you used to create users and profiles.

2.1 About User Security

You can secure users accounts through strong passwords and by specifying special limits for the users.

Each Oracle database has a list of valid database users. To access a database, a user must run a database application, and connect to the database instance using a valid user name defined in the database.

When you create user accounts, you can specify limits to the user account. You can also set limits on the amount of various system resources available to each user as part of the security domain of that user. Oracle Database provides a set of database views that you can query to find information such as resource and session information.

This section also describes profiles. Profiles provide a way to configure the resources for the database user. A profile is collection of attributes that apply to a user. It enables a single point of reference for any of multiple users that share those exact attributes.

Oracle Database provides a set of predefined administrative, non-administrative, and sample schema accounts. The Oracle Database installation guides provide a listing of these accounts. To find the status of these accounts, query the USERNAME and ACCOUNT_STATUS columns of the DBA_USERS data dictionary view.

Related Topics

Configuring Privilege and Role Authorization

Privilege and role authorization controls the permissions that users have to perform day-to-day tasks.

2.2 Creating User Accounts

A user account can have restrictions such as profiles, a default role, and tablespace restrictions.

About Common Users and Local Users

In a multitenant environment, CDB common users and application common have access to their respective containers, and local users are specific to a PDB.

Who Can Create User Accounts?

Users who has been granted the CREATE USER system privilege can create user accounts, including user accounts to be used as proxy users.

- Creating a New User Account That Has Minimum Database Privileges
 When you create a new user account, you should enable this user to access the database.
- Restrictions on Creating the User Name for a New Account
 When you specify a name for a user account, be aware of restrictions such as naming
 conventions and whether the name is unique.
- Assignment of User Passwords

The IDENTIFIED BY clause of the CREATE USER statement assigns the user a password.

Default Tablespace for the User

A default tablespace stores objects that users create.

Tablespace Quotas for a User

The tablespace quota defines how much space to provide for a user's tablespace.

Temporary Tablespaces for the User

A temporary tablespace contains transient data that persists only for the duration of a user session.

Profiles for the User

A profile is a set of limits, defined by attributes, on database resources and password access to the database.

Creation of a Common User or a Local User

The CREATE USER SQL statement can be used to create both common (CDB and application) users and local users.

Creating a Default Role for the User

A default role is automatically enabled for a user when the user creates a session.

2.2.1 About Common Users and Local Users

In a multitenant environment, CDB common users and application common have access to their respective containers, and local users are specific to a PDB.

About Common Users

Oracle provides two types of common users: CDB common users and application common users.



- How Plugging in PDBs Affects CDB Common Users
 Plugging a non-CDB into a CDB as a PDB affects both Oracle-supplied administrative and user-created accounts and privileges.
- About Local Users
 In a multitenant environment, a local user is a database user that exists only in a single PDB.

2.2.1.1 About Common Users

Oracle provides two types of common users: CDB common users and application common users.

A CDB common user is a database user whose single identity and password are known in the CDB root and in every existing and future pluggable database (PDB), including any application roots. All Oracle-supplied administrative user accounts, such as SYS and SYSTEM, are CDB common users and can navigate across the system container. CDB common users can have different privileges in different PDBs. For example, the user SYSTEM can switch between PDBs and use the privileges that are granted to SYSTEM in the current PDB. However, if one of the PDBs is Oracle Database Vault-enabled, then the Database Vault restrictions, such as SYSTEM not being allowed to create user accounts, apply to SYSTEM when this user is connected to that PDB. Oracle does not recommend that you change the privileges of the Oracle-supplied CDB common users.

A CDB common user can perform all tasks that an application common user can perform, provided that appropriate privileges have been granted to that user.

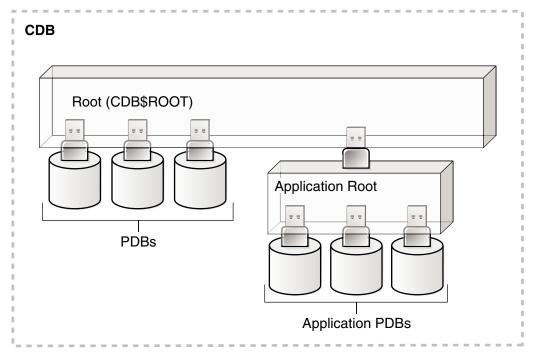
An application common user is a user account that is created in an application root, and is common only within this application container. In other words, the application common user does not have access to the entire CDB environment like CDB common users. An application common user is responsible for activities such as creating (which includes plugging), opening, closing, unplugging, and dropping application PDBs. This user can create application common objects in the application root. You can create an application common user only when you are connected to an application root. The ability for users to access the application common objects is subject to the same privileges as local and CDB common objects. For example, a local user in a PDB that is associated with an application root has access to only the objects in that PDB for which the user has privileges. In the application root itself, you can commonly grant a privilege on a CDB common object that will apply across the application container.

Both of these types of common users are responsible for managing the common objects in their respective roots. If the CDB common user or the application common user has the appropriate privileges, then this user can perform operations in PDBs as well, such as granting privileges to local users. These users can also locally grant common users different privileges in each container.

Both CDB and application common users can perform the following activities:

Granting privileges to common users or common roles. That is, a CDB common user can
grant a privilege to a common user or role, and the scope within which this privilege
applies is determined by the container (CDB root, application root, or PDB) in which the
statement is issued and whether the privilege is granted commonly (in the CDB root or the
application root). A CDB common user connected to an application root can commonly
grant a privilege on a CDB common object, and that privilege will apply across the
application container.

The following diagram illustrates the access hierarchy with CDB common users, application common users, and local users:



CDB common users are defined in the CDB root and may be able to access all PDBs within the CDB, including application roots and their application PDBs. Application common users are defined in the application root and have access to the PDBs that belong to the application container. Local users in either the CDB PDBs or the application PDBs have access only to the PDBs in which the local user resides.

• The state of a PDB can be altered by a suitably privileged user who can issue the ALTER PLUGGABLE DATABASE statement from the CDB root, from an application root (if a PDB is an application PDB that belongs to the application container), or from a PDB itself.

One difference between CDB common users and application common users is that only a CDB common user can run an ALTER DATABASE statement that specifies the recovery clauses that apply to the entire CDB.

See Also:

- About Creating Common User Accounts
- About Commonly and Locally Granted Privileges for more information about how privileges work in with PDBs
- Oracle Database Concepts for more conceptual information about CDB common users and application common users

2.2.1.2 How Plugging in PDBs Affects CDB Common Users

Plugging a non-CDB into a CDB as a PDB affects both Oracle-supplied administrative and user-created accounts and privileges.

This affects the passwords of these CDB common user accounts, and privileges of all accounts in the newly plugged-in database.

The following actions take place:

- The Oracle-supplied administrative accounts are merged with the existing common user accounts.
- User-created accounts are merged with the existing user-created common user accounts.
- The passwords of the existing CDB common user accounts take precedence over the passwords for the accounts from the non-CDB.
- If you had modified the privileges of a user account in its original non-CDB, then these privileges are saved, but they only apply to the PDB that was created when the PDB was plugged into the CDB, as locally granted privileges. For example, suppose you had granted the user SYSTEM a role called hr_mgr in the non-CDB db1. After the db1 database has been added to a CDB, then SYSTEM can only use the hr_mgr role in the db1 PDB, and not in any other PDBs.

The following two scenarios are possible when you plug a PDB (for example, pdb_1) from one CDB (cdb_1) to a another CDB (cdb_2):

- cdb_1 has the common user c##cdb1_user. cdb_2 does not have this user.
 c##cdb1_user remains in PDB_1 but this account is locked. To resurrect this account, you must close pdb_1, recreate common user c##cdb1_user in the root of cdb_2, and then reopen pdb_1.
- cdb 1 and cdb 2 both have common user c##common_user.

Both c##common_user accounts are merged. c##common_user retains its password in cdb 2. Any privileges assigned to it in cdb 2 but not in cdb 1 are retained locally in pdb 1.

2.2.1.3 About Local Users

In a multitenant environment, a local user is a database user that exists only in a single PDB.

Local users can have administrative privileges, but these privileges apply only in the PDB in which the local user account was created. A local user account has the following characteristics, which distinguishes it from common user accounts:

- Local user accounts cannot create common user accounts or commonly grant them
 privileges. A common user with the appropriate privileges can create and modify common
 or local user accounts and grant and revoke privileges, commonly or locally. A local user
 can create and modify local user accounts or locally grant privileges to common or local
 users in a given PDB.
- You can grant local user accounts common roles. However, the privileges associated with the common role only apply to the local user's PDB.
- The local user account must be unique only within its PDB.
- With the appropriate privileges, a local user can access objects in a common user's schema. For example, a local user can access a table within the schema of a common user if the common user has granted the local user privileges to access it.
- You can editions-enable a local user account but not a common user account.

Related Topics

- About Creating Local User Accounts
 Be aware of local user account restrictions such as where they can be created, naming conventions, and objects stored in their schemas.
- Oracle Database Concepts



2.2.2 Who Can Create User Accounts?

Users who has been granted the CREATE USER system privilege can create user accounts, including user accounts to be used as proxy users.

Because the CREATE USER system privilege is a powerful privilege, a database administrator or security administrator is usually the only user who has this system privilege.

If you want to create users who themselves have the privilege to create users, then include the WITH ADMIN OPTION clause in the GRANT statement. For example:

```
GRANT CREATE USER TO 1brown WITH ADMIN OPTION;
```

As with all user accounts to whom you grant privileges, grant these privileges to trusted users only.

In a multitenant environment, you must have the commonly granted CREATE USER system privilege to create common user accounts. To create local user accounts, you must have a commonly granted CREATE USER privilege or a locally granted CREATE USER privilege in the PDB in which the local user account will be created.



As a security administrator, you should create your own roles and assign only those privileges that are needed. For example, many users formerly granted the CONNECT privilege did not need the additional privileges CONNECT used to provide. Instead, only CREATE SESSION was actually needed. By default, the SET CONTAINER privilege is granted to CONNECT role.

Creating organization-specific roles gives an organization detailed control of the privileges it assigns, and protects it in case Oracle Database changes the roles that it defines in future releases.

Related Topics

Configuring Privilege and Role Authorization
 Privilege and role authorization controls the permissions that users have to perform day-to-day tasks.

2.2.3 Creating a New User Account That Has Minimum Database Privileges

When you create a new user account, you should enable this user to access the database.

1. Use the CREATE USER statement to create a new user account.

For example:

CREATE USER jward
IDENTIFIED BY password
DEFAULT TABLESPACE example
QUOTA 10M ON example
TEMPORARY TABLESPACE temp
QUOTA 5M ON system
PASSWORD EXPIRE;



Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

This example creates a local user account and specifies the user password, default tablespace, temporary tablespace where temporary segments are created, tablespace quotas, and profile.

2. At minimum, grant the user the CREATE SESSION privilege so that the user can access the database instance.

GRANT CREATE SESSION TO jward;

A newly created user cannot connect to the database until he or she has the CREATE SESSION privilege. If the user must access Oracle Enterprise Manager, then you should also grant the user the SELECT ANY DICTIONARY privilege.

Related Topics

Restrictions on Creating the User Name for a New Account

When you specify a name for a user account, be aware of restrictions such as naming conventions and whether the name is unique.

Assignment of User Passwords

The IDENTIFIED BY clause of the CREATE USER statement assigns the user a password.

Default Tablespace for the User

A default tablespace stores objects that users create.

Tablespace Quotas for a User

The tablespace quota defines how much space to provide for a user's tablespace.

Temporary Tablespaces for the User

A temporary tablespace contains transient data that persists only for the duration of a user session.

Profiles for the User

A profile is a set of limits, defined by attributes, on database resources and password access to the database.

Creation of a Common User or a Local User

The CREATE USER SQL statement can be used to create both common (CDB and application) users and local users.

2.2.4 Restrictions on Creating the User Name for a New Account

When you specify a name for a user account, be aware of restrictions such as naming conventions and whether the name is unique.

Uniqueness of User Names

Each user has an associated schema; within a schema, each schema object must have a unique name.

User Names in a Multitenant Environment

Within each PDB, a user name must be unique with respect to other user names and roles in that PDB.

Case Sensitivity for User Names

How you create a user name controls the case sensitivity in which the user name is stored in the database.



2.2.4.1 Uniqueness of User Names

Each user has an associated schema; within a schema, each schema object must have a unique name.

Oracle Database will prevent you from creating a user name if it is already exists. You can check existing names by querying the USERNAME column of the DBA USERS data dictionary view.

2.2.4.2 User Names in a Multitenant Environment

Within each PDB, a user name must be unique with respect to other user names and roles in that PDB.

Note the following restrictions:

• For common user names, names for user-created common users must begin with a common user prefix. By default, for CDB common users, this prefix is C##. For application common users, this prefix is an empty string. This means that there are no restrictions on the name that can be assigned to an application common user other than that it cannot start with the prefix reserved for CDB common users. For example, you could name a CDB common user c##hr_admin and an application common user hr_admin.

The COMMON_USER_PREFIX parameter in CDB\$ROOT defines the common user prefix. You can change this setting, but do so only with great care.

- For local user names, the name cannot start with C## (or C##)
- A user and a role cannot have the same name.

2.2.4.3 Case Sensitivity for User Names

How you create a user name controls the case sensitivity in which the user name is stored in the database.

For example:

CREATE USER jward

```
IDENTIFIED BY password
DEFAULT TABLESPACE data_ts
QUOTA 100M ON test_ts
QUOTA 500K ON data_ts
TEMPORARY TABLESPACE temp_ts
PROFILE clerk
CONTAINER = CURRENT;
```

User jward is stored in the database in upper-case letters. For example:

```
SELECT USERNAME FROM ALL_USERS;

USERNAME

JWARD
```

However, if you enclose the user name in double quotation marks, then the name is stored using the case sensitivity that you used for the name. For example:

```
CREATE USER "jward" IDENTIFIED BY password;
```



So, when you query the ALL_USERS data dictionary view, you will find that the user account is stored using the case that you used to create it.

```
SELECT USERNAME FROM ALL_USERS;

USERNAME

-----
jward
...
```

User JWARD and user jward are both stored in the database as separate user accounts. Later on, if you must modify or drop the user that you had created using double quotation marks, then you must enclose the user name in double quotation marks.

For example:

```
DROP USER "jward";
```

2.2.5 Assignment of User Passwords

The IDENTIFIED BY clause of the CREATE USER statement assigns the user a password.

Ensure that you create a secure password.

In the example in Creating a New User Account That Has Minimum Database Privileges, the new local user is authenticated using the database. In this case, the connecting user must supply the correct password to the database to connect successfully.

```
CREATE USER jward

IDENTIFIED BY password

DEFAULT TABLESPACE data_ts
QUOTA 100M ON test_ts
QUOTA 500K ON data_ts

TEMPORARY TABLESPACE temp_ts
PROFILE clerk
CONTAINER = CURRENT;
```

Related Topics

- Minimum Requirements for Passwords
 Oracle provides a set of minimum requirements for passwords.
- Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.

2.2.6 Default Tablespace for the User

A default tablespace stores objects that users create.

- About Assigning a Default Tablespace for a User Each user should have a default tablespace.
- DEFAULT TABLESPACE Clause for Assigning a Default Tablespace
 The DEFAULT TABLESPACE clause in the CREATE USER statement assigns a default tablespace to the user.

2.2.6.1 About Assigning a Default Tablespace for a User

Each user should have a default tablespace.

When a schema object is created in the user's schema and the DDL statement does not specify a tablespace to contain the object, the Oracle Database stores the object in the user's default tablespace.

Tablespaces enable you to separate user data from system data, such as the data that is stored in the SYSTEM tablespace. You use the CREATE USER or ALTER USER statement to assign a default tablespace to a user. The default setting for the default tablespaces of all users is the SYSTEM tablespace. If a user does not create objects, and has no privileges to do so, then this default setting is fine. However, if a user is likely to create any type of object, then you should specifically assign the user a default tablespace, such as the USERS tablespace. Using a tablespace other than SYSTEM reduces contention between data dictionary objects and user objects for the same data files. In general, do not store user data in the SYSTEM tablespace.

You can use the CREATE TABLESPACE SQL statement to create a permanent default tablespace other than SYSTEM at the time of database creation, to be used as the database default for permanent objects. By separating the user data from the system data, you reduce the likelihood of problems with the SYSTEM tablespace, which can in some circumstances cause the entire database to become nonfunctional. This default permanent tablespace is not used by system users, that is, SYS, SYSTEM, and OUTLN, whose default permanent tablespace is SYSTEM. A tablespace designated as the default permanent tablespace cannot be dropped. To accomplish this goal, you must first designate another tablespace as the default permanent tablespace. You can use the ALTER TABLESPACE SQL statement to alter the default permanent tablespace to another tablespace. Be aware that this will affect all users or objects created after the ALTER DDL statement is executed.

You can also set a user default tablespace during user creation, and change it later with the ALTER USER statement. Changing the user default tablespace affects only objects created after the setting is changed.

When you specify the default tablespace for a user, also specify a quota on that tablespace.

2.2.6.2 DEFAULT TABLESPACE Clause for Assigning a Default Tablespace

The DEFAULT TABLESPACE clause in the CREATE USER statement assigns a default tablespace to the user.

In the following CREATE USER statement, the default tablespace for local user jward is data ts:

```
CREATE USER jward
IDENTIFIED BY password
DEFAULT TABLESPACE data_ts
QUOTA 100M ON test_ts
QUOTA 500K ON data_ts
TEMPORARY TABLESPACE temp_ts
PROFILE clerk
CONTAINER = CURRENT;
```

Related Topics

Tablespace Quotas for a User
 The tablespace quota defines how much space to provide for a user's tablespace.

2.2.7 Tablespace Quotas for a User

The tablespace quota defines how much space to provide for a user's tablespace.

- About Assigning a Tablespace Quota for a User
 - You can assign each user a tablespace quota for any tablespace, except a temporary tablespace.
- CREATE USER Statement for Assigning a Tablespace Quota

 The QUOTA clause of the CREATE USER statement assigns the quotas for a tablespace.
- Restriction of the Quota Limits for User Objects in a Tablespace
 You can restrict the quota limits for user objects in a tablespace so that the current quota is
 zero.
- Grants to Users for the UNLIMITED TABLESPACE System Privilege
 To permit a user to use an unlimited amount of any tablespace in the database, grant the user the UNLIMITED TABLESPACE system privilege.

2.2.7.1 About Assigning a Tablespace Quota for a User

You can assign each user a tablespace quota for any tablespace, except a temporary tablespace.

Assigning a quota accomplishes the following:

- Users with privileges to create certain types of objects can create those objects in the specified tablespace.
- Oracle Database limits the amount of space that can be allocated for storage of a user's objects within the specified tablespace to the amount of the quota.

By default, a user has no quota on any tablespace in the database. If the user has the privilege to create a schema object, then you must assign a quota to allow the user to create objects. At a minimum, assign users a quota for the default tablespace, and additional quotas for other tablespaces in which they can create objects. The maximum amount of space that you can assign for a tablespace is 2 TB. If you need more space, then specify UNLIMITED for the QUOTA clause.

You can assign a user either individual quotas for a specific amount of disk space in each tablespace or an unlimited amount of disk space in all tablespaces. Specific quotas prevent a user's objects from using too much space in the database.

You can assign quotas to a user tablespace when you create the user, or add or change quotas later. (You can find existing user quotas by querying the <code>USER_TS_QUOTAS</code> view.) If a new quota is less than the old one, then the following conditions remain true:

- If a user has already exceeded a new tablespace quota, then the objects of a user in the tablespace cannot be allocated more space until the combined space of these objects is less than the new quota.
- If a user has not exceeded a new tablespace quota, or if the space used by the objects of the user in the tablespace falls under a new tablespace quota, then the user's objects can be allocated space up to the new quota.

2.2.7.2 CREATE USER Statement for Assigning a Tablespace Quota

The QUOTA clause of the CREATE USER statement assigns the quotas for a tablespace.

The following CREATE USER statement assigns quotas for the test_ts and data_ts tablespaces:

CREATE USER jward
IDENTIFIED BY password
DEFAULT TABLESPACE data ts



QUOTA 500K ON data_ts
QUOTA 100M ON test_ts
TEMPORARY TABLESPACE temp_ts
PROFILE clerk
CONTAINER = CURRENT;

2.2.7.3 Restriction of the Quota Limits for User Objects in a Tablespace

You can restrict the quota limits for user objects in a tablespace so that the current quota is zero.

To restrict the quote limits, use the ALTER USER SQL statement.

After a quota of zero is assigned, the objects of the user in the tablespace remain, and the user can still create new objects, but the existing objects will not be allocated any new space. For example, you could not insert data into one of this user's existing tables. The operation will fail with an ORA-1536 space quota exceeded for tablespace %s error.

2.2.7.4 Grants to Users for the UNLIMITED TABLESPACE System Privilege

To permit a user to use an unlimited amount of any tablespace in the database, grant the user the UNLIMITED TABLESPACE system privilege.

The UNLIMITED TABLESPACE privilege overrides all explicit tablespace quotas for the user. If you later revoke the privilege, then you must explicitly grant quotas to individual tablespaces. You can grant this privilege only to users, not to roles.

Before granting the UNLIMITED TABLESPACE system privilege, consider the consequences of doing so.

Advantage:

You can grant a user unlimited access to all tablespaces of a database with one statement.

Disadvantages:

- The privilege overrides all explicit tablespace quotas for the user.
- You cannot selectively revoke tablespace access from a user with the UNLIMITED
 TABLESPACE privilege. You can grant selective or restricted access only after revoking the
 privilege.

2.2.8 Temporary Tablespaces for the User

A temporary tablespace contains transient data that persists only for the duration of a user session.

- About Assigning a Temporary Tablespace for a User You should assign each user a temporary tablespace.
- TEMPORARY TABLESPACE Clause for Assigning a Temporary Tablespace
 The TEMPORARY TABLESPACE clause in the CREATE USER statement assigns a user a temporary tablespace.

2.2.8.1 About Assigning a Temporary Tablespace for a User

You should assign each user a temporary tablespace.



When a user executes a SQL statement that requires a temporary segment, Oracle Database stores the segment in the temporary tablespace of the user. These temporary segments are created by the system when performing sort or join operations. Temporary segments are owned by SYS, which has resource privileges in all tablespaces.

To create a temporary tablespace, you can use the CREATE TEMPORARY TABLESPACE SQL statement.

If you do not explicitly assign the user a temporary tablespace, then Oracle Database assigns the user the default temporary tablespace that was specified at database creation, or by an ALTER DATABASE statement at a later time. If there is no default temporary tablespace explicitly assigned, then the default is the SYSTEM tablespace or another permanent default established by the system administrator. Assigning a tablespace to be used specifically as a temporary tablespace eliminates file contention among temporary segments and other types of segments.



If your SYSTEM tablespace is locally managed, then users must be assigned a specific default (locally managed) temporary tablespace. They may not be allowed to default to using the SYSTEM tablespace because temporary objects cannot be placed in locally managed permanent tablespaces.

You can set the temporary tablespace for a user at user creation, and change it later using the ALTER USER statement. You can also establish tablespace groups instead of assigning individual temporary tablespaces.

Related Topics

Oracle Database Administrator's Guide

2.2.8.2 TEMPORARY TABLESPACE Clause for Assigning a Temporary Tablespace

The TEMPORARY TABLESPACE clause in the CREATE USER statement assigns a user a temporary tablespace.

In the following example, the temporary tablespace of jward is temp_ts, a tablespace created explicitly to contain only temporary segments.

```
CREATE USER jward
IDENTIFIED BY password
DEFAULT TABLESPACE data_ts
QUOTA 100M ON test_ts
QUOTA 500K ON data_ts
TEMPORARY TABLESPACE temp_ts
PROFILE clerk
CONTAINER = CURRENT;
```

2.2.9 Profiles for the User

A profile is a set of limits, defined by attributes, on database resources and password access to the database.

The profile can be applied to multiple users, enabling them to share these attributes.

You can specify a profile when you create a user. The PROFILE clause of the CREATE USER statement assigns a user a profile. If you do not specify a profile, then Oracle Database assigns the user a default profile.

For example:

```
CREATE USER jward

IDENTIFIED BY password

DEFAULT TABLESPACE data_ts

QUOTA 100M ON test_ts

QUOTA 500K ON data_ts

TEMPORARY TABLESPACE temp_ts

PROFILE clerk

CONTAINER = CURRENT;
```

In a multitenant environment, different profiles can be assigned to a common user in the root and in a PDB. When the common user logs in to the PDB, a profile whose setting applies to the session depends on whether the settings are password-related or resource-related.

- Password-related profile settings are fetched from the profile that is assigned to the common user in the root. For example, suppose you assign a common profile c##prof (in which FAILED_LOGIN_ATTEMPTS is set to 1) to common user c##admin in the root. In a PDB that user is assigned a local profile local_prof (in which FAILED_LOGIN_ATTEMPTS is set to 6.) Common user c##admin is allowed only one failed login attempt when he or she tries to log in to the PDB where loc prof is assigned to him.
- Resource-related profile settings specified in the profile assigned to a user in a PDB get used without consulting resource-related settings in a profile assigned to the common user in the root. For example, if the profile <code>local_prof</code> that is assigned to user <code>c##admin</code> in a PDB has <code>SESSIONS_PER_USER</code> set to 2, then <code>c##admin</code> is only allowed only 2 concurrent sessions when he or she logs in to the PDB <code>loc_prof</code> is assigned to him, regardless of value of this setting in a profile assigned to him in the root.

Related Topics

Managing Resources with Profiles

A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.

2.2.10 Creation of a Common User or a Local User

The CREATE USER SQL statement can be used to create both common (CDB and application) users and local users.

- About Creating Common User Accounts
 - Be aware of common user account restrictions such as where they can be created, naming conventions, and objects stored in their schemas.
- CREATE USER Statement for Creating a Common User Account
 The CREATE USER statement CONTAINER=ALL clause can be used to create a common user account.
- About Creating Local User Accounts
 - Be aware of local user account restrictions such as where they can be created, naming conventions, and objects stored in their schemas.
- CREATE USER Statement for Creating a Local User Account
 The CREATE USER statement CONTAINER clause can be used to create a local user account.

2.2.10.1 About Creating Common User Accounts

Be aware of common user account restrictions such as where they can be created, naming conventions, and objects stored in their schemas.

To create a common user account, follow these rules:

- To create a CDB common user, you must be connected to the CDB root and have the commonly granted CREATE USER system privilege.
- To create an application common user, you must be connected to the application root and have the commonly granted CREATE USER system privilege.
- You can run the CREATE USER ... CONTAINER = ALL statement to create an application common user in the application root. Afterward, you must synchronize the application so that this user can be visible in the application PDB. For example, for an application named saas sales app:

ALTER PLUGGABLE DATABASE APPLICATION saas sales app SYNC;

- The name that you give the common user who connects to the CDB root must begin with the prefix that is defined in the <code>COMMON_USER_PREFIX</code> parameter in the CDB root, which by default is <code>C##</code>. (You can modify this parameter, but only do so with great caution.) It must contain only ASCII or EBCDIC characters. This naming requirement does not apply to the names of existing Oracle-supplied user accounts, such as <code>SYS</code> or <code>SYSTEM</code>. To find the names of existing user accounts, query the <code>ALL_USERS</code>, <code>CDB_USERS</code>, <code>DBA_USERS</code>, and <code>USER</code> <code>USERS</code> data dictionary views.
- The name that you give the common user who connects to the application root must follow the naming conventions for standard user accounts. By default, the COMMON_USER_PREFIX parameter in the application root is set to an empty string. In other words, you can create a user named hr admin in the application root but not a user named c##hr admin.
- To explicitly designate a user account as a CDB or an application common user, in the CREATE USER statement, specify the CONTAINER=ALL clause. If you are logged into the CDB or application root, and if you omit the CONTAINER clause from your CREATE USER statement, then the CONTAINER=ALL clause is implied.
- Do not create objects in the schemas of common users for a CDB. Instead, you can create
 application common objects. These are objects whose metadata, and in case of data links
 or extended data links, data, is shared between all application PDBs that belong to the
 application container. You must create the application common object in the root of an
 application container.
- If you specify the DEFAULT TABLESPACE, TEMPORARY TABLESPACE, QUOTA...ON, and PROFILE clauses in the CREATE USER statement for a CDB or an application common user account, then you must ensure that these objects—tablespaces, tablespace groups, and profiles—exist in all containers of the CDB for a CDB common user, or in the application root and all PDBs of an application container for an application common user.

2.2.10.2 CREATE USER Statement for Creating a Common User Account

The CREATE USER statement CONTAINER=ALL clause can be used to create a common user account.

You must be in the CDB root to create a CDB common user account and the application root to create an application common user account.



The following example shows how to create a CDB common user account from the CDB root by using the CONTAINER clause, and then granting the user the SET CONTAINER and CREATE SESSION privileges. Common users must have the SET CONTAINER system privilege to navigate between containers. When you create the account, there is a single common password for this common user across all containers.

```
CONNECT SYSTEM
Enter password: password
Connected.

CREATE USER c##hr_admin
IDENTIFIED BY password
DEFAULT TABLESPACE data_ts
QUOTA 100M ON test_ts
QUOTA 500K ON data_ts
TEMPORARY TABLESPACE temp_ts
CONTAINER = ALL;

GRANT SET CONTAINER, CREATE SESSION TO c##hr_admin
CONTAINER = ALL;
```

The next example shows how to create an application common user in the application root (app_root) by using the CONTAINER clause, and then granting the user the SET CONTAINER, and CREATE SESSION system privileges. Finally, to synchronize this user so that it is visible in the application PDBs, the ALTER PLUGGABLE DATABASE APPLICATION APP\$CON SYNC statement is run.

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

CREATE USER app_admin
IDENTIFIED BY password
DEFAULT TABLESPACE data_ts
QUOTA 100M ON temp_ts
QUOTA 500K ON data_ts
TEMPORARY TABLESPACE temp_ts
CONTAINER = ALL;

GRANT SET CONTAINER, CREATE SESSION TO app_admin CONTAINER = ALL;
CONNECT SYSTEM@app_hr_pdb
Enter password: password
Connected.

ALTER PLUGGABLE DATABASE APPLICATION APP$CON SYNC;
```

Related Topics

- About Common Users
 - Oracle provides two types of common users: CDB common users and application common users.
- Creating a Common User Account in Enterprise Manager
 A common user is a user that exists in the root and can access PDBs in the CDB.

2.2.10.3 About Creating Local User Accounts

Be aware of local user account restrictions such as where they can be created, naming conventions, and objects stored in their schemas.

To create a local user account, follow these rules:

- To create a local user account, you must be connected to the PDB in which you want to create the account, and have the CREATE USER privilege.
- The name that you give the local user must not start with a prefix that is reserved for common users, which by default is C## for CDB common users.
- You can include CONTAINER=CURRENT in the CREATE USER statement to specify the user as a local user. If you are connected to a PDB and omit this clause, then the CONTAINER=CURRENT clause is implied.
- You cannot have common users and local users with the same name. However, you can
 use the same name for local users in different PDBs. To find the names of existing user
 accounts, query the ALL_USERS, CDB_USERS, DBA_USERS, and USER_USERS data dictionary
 views.
- Both common and local users connected to a PDB can create local user accounts, as long as they have the appropriate privileges.

2.2.10.4 CREATE USER Statement for Creating a Local User Account

The CREATE USER statement CONTAINER clause can be used to create a local user account.

You must create the local user account in the PDB where you want this account to reside.

The following example shows how to create a local user account using the CONTAINER clause.

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

CREATE USER kmurray
IDENTIFIED BY password
DEFAULT TABLESPACE data_ts
QUOTA 100M ON test_ts
QUOTA 500K ON data_ts
TEMPORARY TABLESPACE temp_ts
PROFILE hr_profile
CONTAINER = CURRENT;
```

Related Topics

- Creating a Common User Account in Enterprise Manager
 A common user is a user that exists in the root and can access PDBs in the CDB.
- About Local Users
 In a multitenant environment, a local user is a database user that exists only in a single PDB.

2.2.11 Creating a Default Role for the User

A default role is automatically enabled for a user when the user creates a session.

You can assign a user zero or more default roles. You cannot set default roles for a user in the CREATE USER statement. When you first create a user, the default role setting for the user is ALL, which causes all roles subsequently granted to the user to be default roles.

Use the ALTER USER statement to change the default roles for the user.

For example:

```
GRANT USER rdale clerk_mgr;
ALTER USER rdale DEFAULT ROLE clerk mgr;
```

Before a role can be made the default role for a user, that user must have been already granted the role.

Related Topics

Managing User Roles

A user role is a named collection of privileges that you can create and assign to other users.

2.3 Altering User Accounts

The ALTER USER statement modifies user accounts, such their default tablespace or profile, or changing a user's password.

- About Altering User Accounts
 - Changing user security settings affects the future user sessions, not the current session.
- Methods of Altering Common or Local User Accounts
 You can use the ALTER USER statement or the PASSWORD command to alter both common
 and local user accounts.
- Changing Non-SYS User Passwords
 Users can change their own passwords but to change other users' passwords, they must have the correct privileges.
- Changing the SYS User Password
 To change the SYS user password, you can use the ALTER USER statement, the PASSWORD command, or the ORAPWD command line utility.

2.3.1 About Altering User Accounts

Changing user security settings affects the future user sessions, not the current session.

In most cases, you can alter user security settings with the ALTER USER SQL statement. Users can change their own passwords. However, to change any other option of a user security domain, you must have the ALTER USER system privilege. Security administrators are typically the only users that have this system privilege, as it allows a modification of *any* user security domain. This privilege includes the ability to set tablespace quotas for a user on any tablespace in the database, even if the user performing the modification does not have a quota for a specified tablespace.

In a multitenant environment, you must have the commonly granted ALTER USER system privilege to alter common user accounts. To alter local user accounts, you must have a commonly granted ALTER USER privilege or a locally granted ALTER USER privilege in the PDB in which the local user account resides.

2.3.2 Methods of Altering Common or Local User Accounts

You can use the ALTER USER statement or the PASSWORD command to alter both common and local user accounts.

You cannot change an existing common user account to be a local user account, or a local user account to be made into a common user account. In this case, you must create a new account, as either a common user account or a local user account.

The following example shows how to use the ALTER USER statement to restrict user c##hr_admin's ability to view V\$SESSION rows to those that pertain to sessions that are connected to CDB\$ROOT, and to the emp db and hr db PDBs.

```
CONNECT SYSTEM
Enter password: password
Connected.

ALTER USER c##hr_admin
DEFAULT TABLESPACE data_ts
TEMPORARY TABLESPACE temp_ts
QUOTA 100M ON data_ts
QUOTA 0 ON test_ts
SET CONTAINER_DATA = (emp_db, hr_db) FOR V$SESSION
CONTAINER = CURRENT;
```

The ALTER USER statement here changes the security settings for the user c##hr_admin as follows:

- DEFAULT TABLESPACE and TEMPORARY TABLESPACE are set explicitly to data_ts and temp ts, respectively.
- QUOTA 100M gives the data ts tablespace 100 MB.
- QUOTA 0 revokes the quota on the temp ts tablespace.
- SET CONTAINER_DATA enables user c##hr_admin to have access to data related to the emp_db and hr_db PDBs as well as the root when he queries the V\$SESSION view from the root.

To change passwords, you can use ALTER USER, but Oracle recommends that you use the PASSWORD command to change passwords, for both non-SYS and SYS user accounts.

Related Topics

- Oracle Database SQL Language Reference
- About Changing Non-SYS User Passwords
 Users can use either the PASSWORD command or ALTER USER statement to change a
 password.
- About Changing the SYS User Password
 The method of changing the SYS password that you choose will depend on how your database is configured (for example, how the REMOTE_LOGIN_PASSWORDFILE initialization parameter is set).

2.3.3 Changing Non-SYS User Passwords

Users can change their own passwords but to change other users' passwords, they must have the correct privileges.

- About Changing Non-SYS User Passwords
 Users can use either the PASSWORD command or ALTER USER statement to change a password.
- Using the PASSWORD Command or ALTER USER Statement to Change a Password Most users can change their own passwords with the SQL*Plus PASSWORD command or the ALTER USER SQL statement.

2.3.3.1 About Changing Non-SYS User Passwords

Users can use either the PASSWORD command or ALTER USER statement to change a password.

No special privileges (other than those to connect to the database and create a session) are required for a user to change his or her own password. Encourage users to change their passwords frequently. You can find existing users for the current database instance by querying the ALL USERS view.

For better security, use the PASSWORD command to change the account's password. The ALTER USER statement displays the new password on the screen, where it can be seen by any overly curious coworkers. The PASSWORD command does not display the new password, so it is only known to you, not to your co-workers. The PASSWORD command also encrypts the password on the network. ALTER USER will send the password in clear text, so you should not use it unless the network connection between the client and database is encrypted or the session is a local session not routed over the network.

Users must have the Password and Alter user privilege to switch between methods of authentication. Usually, only an administrator has this privilege.

Related Topics

- Minimum Requirements for Passwords
 Oracle provides a set of minimum requirements for passwords.
- Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.
- Configuring Authentication
 Authentication means to verify the identity of users or other entities that connect to the database.

2.3.3.2 Using the PASSWORD Command or ALTER USER Statement to Change a Password

Most users can change their own passwords with the SQL*Plus PASSWORD command or the ALTER USER SQL statement.

In a multitenant environment, a CDB common user must change his or her password in the CDB root, and an application common user must change his or her password in the application root.

- Use one of the following methods to change a user's password:
 - To use the SQL*Plus PASSWORD command to change a password, supply the user's name, and when prompted, enter the new password.

For example:

PASSWORD andy Changing password for andy New password: password Retype new password: password

 To use the ALTER USER SQL statement change a password, include the IDENTIFIED BY clause.

For example:

ALTER USER andy IDENTIFIED BY password;



2.3.4 Changing the SYS User Password

To change the SYS user password, you can use the ALTER USER statement, the PASSWORD command, or the ORAPWD command line utility.

- About Changing the SYS User Password
 The method of changing the SYS password that you choose will depend on how your database is configured (for example, how the REMOTE_LOGIN_PASSWORDFILE initialization parameter is set).
- ORAPWD Utility for Changing the SYS User Password
 The ORAPWD utility enables you to change the SYS user password.

2.3.4.1 About Changing the SYS User Password

The method of changing the SYS password that you choose will depend on how your database is configured (for example, how the REMOTE_LOGIN_PASSWORDFILE initialization parameter is set).

You an use the Password command, the Alter user statement, or the Orapwd utility to change sys password.

As with non-SYS user accounts, there are good reasons for using PASSWORD to change the SYS user account. PASSWORD does not show the new password on the screen, and PASSWORD also encrypts the password over the network. ALTER USER will send the password in clear text, so you should not use it unless the network connection between the client and database is encrypted or the session is a local session not routed over the network. Hence, you should use PASSWORD for remote connections.

The ALTER USER statement has the following advantages over using ORAPWD:

- It enables you to change the SYS user password from within the Oracle database instance.
- In an Oracle Data Guard environment, it propagates the SYS password change to Oracle Data Guard instances.

Be aware that Oracle Real Application Clusters (Oracle RAC) databases using a shared password file will have REMOTE_LOGIN_PASSWORDFILE = SHARED, which prevents ALTER USER from updating SYS password. If the password file is not shared and the password is changed, then you must copy the password file to all the nodes in the Oracle RAC cluster.

If the <code>REMOTE_LOGIN_PASSWORDFILE</code> initialization parameter is set and you want to use <code>ALTERUSER</code> to change the <code>SYS</code> password, then note the following:

- Ensure that the REMOTE_LOGIN_PASSWORDFILE initialization parameter is set to EXCLUSIVE. Otherwise, the SYS user password change (or any administrative user password change) attempt will fail.
- If REMOTE_LOGIN_PASSWORDFILE is null or set to NONE, then the password change attempt fails with an ORA-01994 error.
- If REMOTE_LOGIN_PASSWORDFILE is set to SHARED, then using the ALTER USER statement to change the password fails with an ORA-28046 error.

If you want to use ORAPWD to change the SYS password, then note the following:

 Before you can change the password of the SYS user account, a password file must exist for this account.



• If the instance initialization parameter REMOTE_LOGIN_PASSWORDFILE is set to SHARED or is null, then you must use ORAPWD to change the SYS password.

The following applies to both the ALTER USER and ORAPWD methods of changing the SYS user password:

- New accounts are created with the SHA-2 (SHA-512) verifier. SYS user verifiers are generated based on the sqlnet.ora parameter ALLOWED_LOGON_VERSION_SERVER. You can identify these accounts by querying the PASSWORD_VERSIONS column of the DBA_USERS data dictionary view. (These verifiers are listed as 12C in the PASSWORD_VERSIONS column of the DBA_USERS view output.)
- In an Oracle Real Application Clusters (Oracle RAC) environment, store the password in the ASM disk group so that it can be shared by multiple Oracle RAC instances.

Related Topics

- Ensuring Against Password Security Threats by Using the 12C Password Version
 The 12C password version enables users to create complex passwords that meet
 compliance standards.
- Oracle Database Administrator's Guide

2.3.4.2 ORAPWD Utility for Changing the SYS User Password

The ORAPWD utility enables you to change the SYS user password.

You can use the <code>ORAPWD</code> utility with the <code>INPUT_FILE</code> parameter to change the <code>SYS</code> user password. To migrate the password files to a specific format, include the <code>FORMAT</code> option. By default, the format is 12.2 if you do not specify the <code>FORMAT</code> option.

To set a new password for the SYS user using the ORAPWD utility, set the SYS option to Y (yes), use the INPUT_FILE parameter to specify the current password file name, and use the FILE parameter to create the password file to which the original password file is migrated. For example:

```
ORAPWD INPUT_FILE='orapworcl' FILE='orapwd' SYS=Y Enter password for SYS: new_password
```

Replace <code>new_password</code> with a password that is secure. If you do not want to migrate the password file to a different format, then you can specify the same format as the <code>input_file</code>. For example, assuming that the input file <code>orapworcl</code> format is 12 and you want to change the <code>SYS</code> user password:

```
ORAPWD INPUT_FILE='orapworcl' FILE='orapwd' FORMAT=12 SYS=Y Enter password for SYS: new_password
```

Related Topics

- Oracle Database Administrator's Guide
- Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.

2.4 Configuring User Resource Limits

A resource limit defines the amount of system resources that are available for a user.



About User Resource Limits

You can set limits on the amount of system resources available to each user as part of the security domain of that user.

Types of System Resources and Limits

You can limit several types of system resources, including CPU time and logical reads, at the session level, call level, or both.

Values for Resource Limits of Profiles

Before you create profiles and set resource limits, you should determine appropriate values for each resource limit.

Managing Resources with Profiles

A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.

2.4.1 About User Resource Limits

You can set limits on the amount of system resources available to each user as part of the security domain of that user.

By doing so, you can prevent the uncontrolled consumption of valuable system resources such as CPU time.

This resource limit feature is very useful in large, multiuser systems, where system resources are very expensive. Excessive consumption of these resources by one or more users can detrimentally affect the other users of the database. In single-user or small-scale multiuser database systems, the system resource feature is not as important, because user consumption of system resources is less likely to have a detrimental impact.

You manage user resource limits by using Database Resource Manager. You can set password management preferences using profiles, either set individually or using a default profile for many users. Each Oracle database can have an unlimited number of profiles. Oracle Database allows the security administrator to enable or disable the enforcement of profile resource limits universally.

Setting resource limits causes a slight performance degradation when users create sessions, because Oracle Database loads all resource limit data for each user upon each connection to the database.

Related Topics

Oracle Database Administrator's Guide

2.4.2 Types of System Resources and Limits

You can limit several types of system resources, including CPU time and logical reads, at the session level, call level, or both.

Limits to the User Session Level

When a user connects to a database, a session is created. Sessions use CPU time and memory, on which you can set limits.

Limits to Database Call Levels

Each time a user runs a SQL statement, Oracle Database performs several steps to process the statement.

Limits to CPU Time

When SQL statements and other calls are made to an Oracle database, CPU time is necessary to process the call.

- Limits to Logical Reads
 Input/output (I/O) is one of the most expensive operations in a database system.
- Limits to Other Resources
 You can control limits for user concurrent sessions and idle time.

2.4.2.1 Limits to the User Session Level

When a user connects to a database, a session is created. Sessions use CPU time and memory, on which you can set limits.

You can set several resource limits at the session level. If a user exceeds a session-level resource limit, then Oracle Database terminates (rolls back) the current statement and returns a message indicating that the session limit has been reached. At this point, all previous statements in the current transaction are intact, and the only operations the user can perform are COMMIT, ROLLBACK, or disconnect (in this case, the current transaction is committed). All other operations produce an error. Even after the transaction is committed or rolled back, the user cannot accomplish any more work during the current session.

2.4.2.2 Limits to Database Call Levels

Each time a user runs a SQL statement, Oracle Database performs several steps to process the statement.

During the SQL statement processing, several calls are made to the database as a part of the different execution phases. To prevent any one call from using the system excessively, Oracle Database lets you set several resource limits at the call level.

If a user exceeds a call-level resource limit, then Oracle Database halts the processing of the statement, rolls back the statement, and returns an error. However, all previous statements of the current transaction remain intact, and the user session remains connected.

2.4.2.3 Limits to CPU Time

When SQL statements and other calls are made to an Oracle database, CPU time is necessary to process the call.

Average calls require a small amount of CPU time. However, a SQL statement involving a large amount of data or a runaway query can potentially use a large amount of CPU time, reducing CPU time available for other processing.

To prevent uncontrolled use of CPU time, you can set fixed or dynamic limits on the CPU time for each call and the total amount of CPU time used for Oracle Database calls during a session. The limits are set and measured in CPU one-hundredth seconds (0.01 seconds) used by a call or a session.

2.4.2.4 Limits to Logical Reads

Input/output (I/O) is one of the most expensive operations in a database system.

SQL statements that are I/O-intensive can monopolize memory and disk use and cause other database operations to compete for these resources.

To prevent single sources of excessive I/O, you can limit the logical data block reads for each call and for each session. Logical data block reads include data block reads from both memory and disk. The limits are set and measured in number of block reads performed by a call or during a session.

2.4.2.5 Limits to Other Resources

You can control limits for user concurrent sessions and idle time.

Limits to other resources are as follows:

- You can limit the number of concurrent sessions for each user. Each user can create
 only up to a predefined number of concurrent sessions.
- You can limit the idle time for a session. If the time between calls in a session reaches the idle time limit, then the current transaction is rolled back, the session is terminated, and the resources of the session are returned to the system. The next call receives an error that indicates that the user is no longer connected to the instance. This limit is set as a number of elapsed minutes.



Shortly after a session is terminated because it has exceeded an idle time limit, the process monitor (PMON) background process cleans up after the terminated session. Until PMON completes this process, the terminated session is still counted in any session or user resource limit.

• You can limit the elapsed connect time for each session. If the duration of a session exceeds the elapsed time limit, then the current transaction is rolled back, the session is dropped, and the resources of the session are returned to the system. This limit is set as a number of elapsed minutes.

Note:

Oracle Database does not constantly monitor the elapsed idle time or elapsed connection time. Doing so reduces system performance. Instead, it checks every few minutes. Therefore, a session can exceed this limit slightly (for example, by 5 minutes) before Oracle Database enforces the limit and terminates the session.

You can limit the amount of private System Global Area (SGA) space (used for private SQL areas) for a session. This limit is only important in systems that use the shared server configuration. Otherwise, private SQL areas are located in the Program Global Area (PGA). This limit is set as a number of bytes of memory in the SGA of an instance. Use the characters K or M to specify kilobytes or megabytes.

2.4.3 Values for Resource Limits of Profiles

Before you create profiles and set resource limits, you should determine appropriate values for each resource limit.

You can base the resource limit values on the type of operations a typical user performs. For example, if one class of user does not usually perform a high number of logical data block reads, then use the ALTER RESOURCE COST SQL statement to set the LOGICAL_READS_PER_SESSION setting conservatively.

Usually, the best way to determine the appropriate resource limit values for a given user profile is to gather historical information about each type of resource usage. For example, the

database or security administrator can use the AUDIT SESSION clause to gather information about the limits CONNECT TIME, LOGICAL READS PER SESSION.

In an Oracle Data Guard environment, an active standby database is opened in read-only mode. This allows user connections on it in the same way as on a primary database. Hence, all the password resource-related limits of a given user profile will work independently between them, except for the ones that imply or require a user password change in the standby database; this task cannot be performed in a database that is opened in read-only mode.

You can gather statistics for other limits using the Monitor feature of Oracle Enterprise Manager (or SQL*Plus), specifically the Statistics monitor.

2.4.4 Managing Resources with Profiles

A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.

About Profiles

A profile is a collection of attributes that apply to a user.

ora stig profile User Profile

The <code>ora_stig_profile</code> user profile is designed for Security Technical Implementation Guide compliance.

Creating a Profile

A profile can encompass limits for a specific category, such as limits on passwords or limits on resources.

Creating a CDB Profile or an Application Profile

The CREATE PROFILE or ALTER PROFILE statement CONTAINER=ALL clause can create a profile in a CDB or application root.

Assigning a Profile to a User

After you create a profile, you can assign it to users.

Dropping Profiles

You can drop a profile, even if it is currently assigned to a user.

2.4.4.1 About Profiles

A profile is a collection of attributes that apply to a user.

The **profile** is used to enable a single point of reference for multiple users who share these attributes.

You should assign a profile to each user. Each user can have only one profile, and creating a new one supersedes an earlier assignment.

You can create and manage user profiles only if resource limits are a requirement of your database security policy. To use profiles, first categorize the related types of users in a database. Just as roles are used to manage the privileges of related users, profiles are used to manage the resource limits of related users. Determine how many profiles are needed to encompass all categories of users in a database and then determine appropriate resource limits for each profile.

User profiles in Oracle Internet Directory contain attributes pertinent to directory usage and authentication for each user. Similarly, profiles in Oracle Label Security contain attributes useful in label security user administration and operations management. Profile attributes can include restrictions on system resources. You can use Database Resource Manager to set these types of resource limits.

In a multitenant environment, profiles are useful for the administration and operations performed in the container databases (CDBs) and application containers, as well as their associated pluggable databases (PDBs). For both CDB and application containers, if you define a common profile, then the profile applies to the entire container and not outside this container. If you create a local profile, then it applies to that PDB only.

Profile resource limits are enforced only when you enable resource limitation for the associated database. Enabling this limitation can occur either before starting the database (using the RESOURCE_LIMIT initialization parameter) or while it is open (using the ALTER SYSTEM statement).

Though password parameters reside in profiles, they are unaffected by RESOURCE_LIMIT or ALTER SYSTEM and password management is always enabled. In Oracle Database, Database Resource Manager primarily handles resource allocations and restrictions.

Any authorized database user can create, assign to users, alter, and drop a profile at any time (using the CREATE USER or ALTER USER statement). Profiles can be assigned only to users and not to roles or other profiles. Profile assignments do not affect current sessions; instead, they take effect only in subsequent sessions.

To find information about current profiles, query the DBA PROFILES view.



Oracle Database Administrator's Guide for detailed information about managing resources

2.4.4.2 ora stig profile User Profile

The <code>ora_stig_profile</code> user profile is designed for Security Technical Implementation Guide compliance.

The ora_stig_profile user profile addresses STIG requirements such as the need for a password complexity function, maximum failed login attempts, reuse time, and other requirements. The definition for this profile is as follows:

```
CREATE PROFILE ora_stig_profile

password_life_time 60

password_grace_time 5

password_reuse_time 365

password_reuse_max 10

failed_login_attempts 3

password_lock_time unlimited
inactive_account_time 35

idle_time 15

password_verify function ora12c stig verify function;
```

2.4.4.3 Creating a Profile

A profile can encompass limits for a specific category, such as limits on passwords or limits on resources.

To create a profile, you must have the CREATE PROFILE system privilege. To find all existing profiles, you can query the DBA_PROFILES view.

• Use the CREATE PROFILE statement to create a profile.

For example, to create a profile that defines password limits:

```
CREATE PROFILE password_prof LIMIT
FAILED_LOGIN_ATTEMPTS 6
PASSWORD_LIFE_TIME 60
PASSWORD_REUSE_TIME 60
PASSWORD_REUSE_MAX 5
PASSWORD_LOCK_TIME 1/24
PASSWORD_GRACE_TIME 10
PASSWORD VERIFY FUNCTION DEFAULT;
```

The following example shows how to create a resource limits profile.

```
CREATE PROFILE app_user LIMIT

SESSIONS_PER_USER UNLIMITED

CPU_PER_SESSION UNLIMITED

CPU_PER_CALL 3500

CONNECT_TIME 50

LOGICAL_READS_PER_SESSION DEFAULT

LOGICAL_READS_PER_CALL 1200

PRIVATE_SGA 20K

COMPOSITE LIMIT 7500000;
```

Related Topics

Oracle Database SQL Language Reference

2.4.4.4 Creating a CDB Profile or an Application Profile

The CREATE PROFILE or ALTER PROFILE statement CONTAINER=ALL clause can create a profile in a CDB or application root.

You cannot create local profiles in the CDB root or the application root. The profile that you create will be applied to all PDBs that are associated with the CDB root or the application root. Create the profile using the same parameters that you would in a non-multitenant environment.

To create a profile in a CDB root or an application root, optionally include the CONTAINER=ALL clause in the CREATE PROFILE or ALTER PROFILE statement.

The CONTAINER=ALL clause is optional because it is the default when the statement is processed.

For example:

```
CREATE PROFILE password_prof LIMIT
FAILED_LOGIN_ATTEMPTS 6
PASSWORD_LIFE_TIME 60
PASSWORD_REUSE_TIME 60
PASSWORD_REUSE_MAX 5
PASSWORD_LOCK_TIME 1/24
PASSWORD_GRACE_TIME 10
PASSWORD_VERIFY_FUNCTION_DEFAULT
CONTAINER=ALL;
```

2.4.4.5 Assigning a Profile to a User

After you create a profile, you can assign it to users.

You can assign a profile to a user who has already been assigned a profile, but the most recently assigned profile takes precedence. When you assign a profile to an external user or a global user, the password parameters do not take effect for that user.

To find the profiles that are currently assigned to users, you can query the DBA_USERS view.

Use the ALTER USER statement to assign the profile to a user.

For example:

ALTER USER psmith PROFILE app user;

2.4.4.6 Dropping Profiles

You can drop a profile, even if it is currently assigned to a user.

When you drop a profile, the drop does not affect currently active sessions. Only sessions that were created after a profile is dropped use the modified profile assignments. To drop a profile, you must have the DROP PROFILE system privilege. You cannot drop the default profile.

• Use the SQL statement DROP PROFILE to drop a profile. To drop a profile that is currently assigned to a user, use the CASCADE option.

For example:

DROP PROFILE clerk CASCADE;

Any user currently assigned to a profile that is dropped is automatically is assigned to the DEFAULT profile. The DEFAULT profile cannot be dropped.

Related Topics

Oracle Database SQL Language Reference

2.5 Dropping User Accounts

You can drop user accounts if the user is not in a session, and if the user has objects in the user's schema.

- About Dropping User Accounts
 - Before you drop a user account, you must ensure that you have the appropriate privileges for doing so.
- Terminating a User Session
 - A user who is connected to a database cannot be dropped.
- About Dropping a User After the User Is No Longer Connected to the Database
 After a user is disconnected from the database, you can use the DROP USER statement to
 drop the user.
- Dropping a User Whose Schema Contains Objects
 Before you drop a user whose schema contains objects, carefully investigate the implications of dropping these schema objects.

2.5.1 About Dropping User Accounts

Before you drop a user account, you must ensure that you have the appropriate privileges for doing so.

To drop a user account in any environment, you must have the DROP USER system privilege. In a multitenant environment, you must have the commonly granted DROP USER system privilege to drop common user accounts. To drop local user accounts, you must have a commonly granted DROP USER privilege or a locally granted DROP USER privilege in the PDB in which the local user account resides.

When you drop a user account, Oracle Database removes the user account and associated schema from the data dictionary. It also immediately drops all schema objects contained in the user schema, if any.

Note:

- If a user schema and associated objects must remain but the user must be denied access to the database, then revoke the CREATE SESSION privilege from the user.
- Do not attempt to drop the SYS or SYSTEM user. Doing so corrupts your database.

2.5.2 Terminating a User Session

A user who is connected to a database cannot be dropped.

You must first terminate the user session (or the user can exit the session) before you can drop the user.

Query the V\$SESSION dynamic view to find the session ID of the user whose session you
want to terminate.

For example:

```
SELECT SID, SERIAL#, USERNAME FROM V$SESSION;

SID SERIAL# USERNAME

127 55234 ANDY
```

2. Use the ALTER SYSTEM SQL statement to stop the session for the user, based on the SID and SERIAL# settings of the V\$SESSION view.

For example:

```
ALTER SYSTEM KILL SESSION '127, 55234';
```

2.5.3 About Dropping a User After the User Is No Longer Connected to the Database

After a user is disconnected from the database, you can use the DROP USER statement to drop the user.

To drop a user and all the user schema objects (if any), you must have the DROP USER system privilege. Because the DROP USER system privilege is powerful, a security administrator is typically the only type of user that has this privilege.

If the schema of the user contains any dependent schema objects, then use the CASCADE option to drop the user and all associated objects and foreign keys that depend on the tables of the user successfully. If you do not specify CASCADE and the user schema contains dependent objects, then an error message is returned and the user is not dropped.

2.5.4 Dropping a User Whose Schema Contains Objects

Before you drop a user whose schema contains objects, carefully investigate the implications of dropping these schema objects.

1. Query the DBA OBJECTS data dictionary view to find the objects that are owned by the user.

For example:

```
SELECT OWNER, OBJECT NAME FROM DBA OBJECTS WHERE OWNER LIKE 'ANDY';
```

Enter the user name in capital letters. Pay attention to any unknown cascading effects. For example, if you intend to drop a user who owns a table, then check whether any views or procedures depend on that particular table.

2. Use the DROP USER SQL statement with the CASCADE clause to drop the user and all associated objects and foreign keys that depend on the tables that the user owns.

For example:

DROP USER andy CASCADE;

2.6 Predefined Schema User Accounts Provided by Oracle Database

The Oracle Database installation process creates predefined administrative, non-administrative, and sample schema user accounts in the database.

- About the Predefined Schema User Accounts
 - The predefined schema accounts are either created automatically when you run standard Oracle scripts or they are accounts that represent a fictional company.
- Predefined Administrative Accounts
 - A default Oracle Database installation provides predefined administrative accounts to manage commonly used features, such as auditing.
- Predefined Non-Administrative User Accounts
 - A default Oracle Database installation provides non-administrative user accounts to manage features such as Oracle Spatial.
- Predefined Sample Schema User Accounts
 Oracle Database creates a set of sample user accounts if you install the sample schemas.

2.6.1 About the Predefined Schema User Accounts

The predefined schema accounts are either created automatically when you run standard Oracle scripts or they are accounts that represent a fictional company.

The predefined schema accounts are in two categories:

- The predefined administrative and non-administrative schema accounts are created automatically when you run standard scripts such as the various <code>cat.*sql</code> scripts. You can find these accounts by querying the <code>USERNAME</code> and <code>ORACLE_MAINTAINED</code> columns of the <code>ALL_USERS</code> data dictionary view. If the output for <code>ORACLE_MAINTAINED</code> is <code>Y</code>, then you must not modify the user account except by running the script that was used to create it.
- The HR sample schema user account is installed by default. A set of additional schema user accounts (OE, PM, IX, and SH, along with HR) is available on GitHub. These schema

accounts represent different divisions of a fictional company that manufactures various products. You can find the status of these accounts by querying the <code>DBA_USERS</code> data dictionary view. Because the <code>ORACLE_MAINTAINED</code> column output for these accounts is N, you can modify these accounts without re-running the scripts that were used to create them.

By default, most of these accounts are authenticated as schema only accounts, except for the sample schema accounts, which are locked and expired during the database installation process. When using these accounts, you can configure them to be authenticated in other ways (such as with password authentication), but Oracle recommends that for better security, to keep these accounts as schema only accounts.

Related Topics

- Oracle Database Sample Schemas
- Schema-Only Accounts
 You can create schema-only accounts, that is, the schema user has no password.

2.6.2 Predefined Administrative Accounts

A default Oracle Database installation provides predefined administrative accounts to manage commonly used features, such as auditing.

These are accounts that have special privileges required to administer areas of the database, such as the CREATE ANY TABLE OF ALTER SESSION privilege, or EXECUTE privileges on packages owned by the SYS schema. The default tablespace for administrative accounts is either SYSTEM or SYSAUX. In a multitenant environment, the predefined administrative accounts reside in the root database.

To protect these accounts from unauthorized access, the installation process expires and locks most of these accounts, except where noted in the following table. As the database administrator, you are responsible for unlocking and resetting these accounts.

Table 2-1 lists the predefined administrative user accounts, which Oracle Database automatically creates when you run standard scripts (such as the various <code>cat*.sql</code> scripts). You can find a complete list of user accounts that are created and maintained by Oracle by querying the <code>USERNAME</code> and <code>ORACLE_MAINTAINED</code> columns of the <code>ALL_USERS</code> data dictionary view. If the output for <code>ORACLE_MAINTAINED</code> is <code>Y</code>, then you must not modify the user account except by running the script that was used to create it.

To find the status of an account, such as whether it is open, locked, or expired, query the ACCOUNT_STATUS column of the DBA_USERS data dictionary view. If the account is schema only, then the status is NONE.

Table 2-1 Predefined Oracle Database Administrative User Accounts

User Account	Description	
ANONYMOUS	An account that allows HTTP access to Oracle XML DB. It is used in place of the APEX_PUBLIC_USER account when the Embedded PL/SQL Gateway (EPG) is installed in the database.	
	EPG is a Web server that can be used with Oracle Database. It provides the necessary infrastructure to create dynamic applications.	
APPQOSSYS	Used for storing and managing all data and metadata required by Oracle Quality of Service Management.	



Table 2-1 (Cont.) Predefined Oracle Database Administrative User Accounts

User Account	Description	
AUDSYS	The internal account used by the unified audit feature to store unified audit trail records.	
	See When and Where Are Audit Records Created?.	
CTXSYS	The account used to administer Oracle Text. Oracle Text enables you to build text query applications and document classification applications. It provides indexing, word and theme searching, and viewing capabilities for text.	
	See Oracle Text Application Developer's Guide.	
DBSNMP	The account used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database.	
	See Enterprise Manager Cloud Control Administrator's Guide.	
DBSFWUSER	The account used to run the DBMS_SFW_ACL_ADMIN package.	
	See Oracle Database PL/SQL Packages and Types Reference.	
DVF	The account owned by Oracle Database Vault that contains public functions to retrieve Database Vault factor values.	
	See Oracle Database Vault Administrator's Guide	
DVSYS	Oracle Database Vault account that is associated with the DV_OWNER (for administrative configurations) and DV_ACCTMGR (for account management) roles.	
	See Oracle Database Vault Administrator's Guide	
GGSYS	The internal account used by Oracle GoldenGate. It should not be unlocked or used for a database login.	
	See Oracle Database Global Data Services Concepts and Administration Guide	
GSMADMIN_INTERNAL	The internal account that owns the Global Data Services schema. It should not be unlocked or used for a database login.	
	See Oracle Database Global Data Services Concepts and Administration Guide	
GSMCATUSER	The account used by Global Service Manager to connect to the Global Data Services catalog.	
	See Oracle Database Global Data Services Concepts and Administration Guide	
GSMROOTUSER	An account that is used to log into CDB\$ROOT for CDBs in a sharding configuration. This user not used in GDS configurations. Any connections to CDB\$ROOT in a CDB are with GSMROOTUS	
GSMUSER	The account used by Global Service Manager to connect to the database.	
	See Oracle Database Global Data Services Concepts and Administration Guide	
LBACSYS	The account used to administer Oracle Label Security (OLS). It is created only when you install the Label Security custom option.	
	See Oracle Label Security Administrator's Guide.	
MDSYS	The Oracle Spatial and Oracle Multimedia Locator administrator account.	
	See Oracle Spatial and Graph Developer's Guide.	
OJVMSYS	The account that is used with the Java Naming and Directory Interface (JNDI) support with Oracle JVM support. This account owns database tables that store the following details about JVM objects: namespace metadata, bound names, attributes, permissions, and stored object representations.	
	See Oracle Database Java Developer's Guide.	
OLAPSYS	The account that owns the OLAP Catalog (CWMLite). This account has been deprecated, but is retained for backward compatibility.	
ORDDATA	This account contains the Oracle Multimedia DICOM data model. See <i>Oracle Multimedia DICOM Developer's Guide</i> for more information.	



Table 2-1 (Cont.) Predefined Oracle Database Administrative User Accounts

User Account	Description
ORDPLUGINS	The Oracle Multimedia user. Plug-ins supplied by Oracle and third-party, format plug-ins are installed in this schema.
	Oracle Multimedia enables Oracle Database to store, manage, and retrieve images, audio, video, DICOM format medical images and other objects, or other heterogeneous media data integrated with other enterprise information.
	See Oracle Multimedia User's Guide .
ORDSYS	The Oracle Multimedia administrator account.
	See Oracle Multimedia User's Guide .
OUTLN	The account that supports plan stability. Plan stability enables you to maintain the same execution plans for the same SQL statements. OUTLN acts as a role to centrally manage metadata associated with stored outlines.
REMOTE_SCHEDULER_AGE NT	The account to disable remote jobs on a database. This account is created during the remote scheduler agent configuration. You can disable the capability of a database to run remote jobs by dropping this user.
	See Oracle Database Administrator's Guide.
SI_INFORMTN_SCHEMA	The account that stores the information views for the SQL/MM Still Image Standard.
	See Oracle Multimedia User's Guide .
	Note: The SI_INFORMTN_SCHEMA account is deprecated in Oracle Database 12c release 2 (12.2).
SYS	An account used to perform database administration tasks.
	See Oracle Database 2 Day DBA.
SYS\$UMF	The account used to administer Remote Management Framework, including the remote Automatic Workload Repository (AWR).
	See Oracle Database Performance Tuning Guide.
SYSBACKUP	The account used to perform Oracle Recovery Manager recovery and backup operations. See Oracle Database Backup and Recovery User's Guide.
SYSDG	The account used to perform Oracle Data Guard operations.
	See Oracle Data Guard Concepts and Administration.
SYSKM	The account used to manage Transparent Data Encryption.
	See Oracle Database Advanced Security Guide.
SYSRAC	The account used to manage Oracle Real Application Clusters.
	See Oracle Real Application Clusters Administration and Deployment Guide.
SYSTEM	A default generic database administrator account for Oracle databases.
	For production systems, Oracle recommends creating individual database administrator accounts and not using the generic SYSTEM account for database administration operations.
	See Oracle Database 2 Day DBA.
WMSYS	The account used to store the metadata information for Oracle Workspace Manager. See Oracle Database Workspace Manager Developer's Guide.
XDB	The account used for storing Oracle XML DB data and metadata. For better security, never unlock the XDB user account.
	Oracle XML DB provides high-performance XML storage and retrieval for Oracle Database data
	See Oracle XML DB Developer's Guide.



Note:

If you create an Oracle Automatic Storage Management (Oracle ASM) instance, then the ASMSNMP account is created. Oracle Enterprise Manager uses this account to monitor ASM instances to retrieve data from ASM-related data dictionary views. The ASMSNMP account status is set to OPEN upon creation, and it is granted the SYSDBA administrative privilege.

2.6.3 Predefined Non-Administrative User Accounts

A default Oracle Database installation provides non-administrative user accounts to manage features such as Oracle Spatial.

Table 2-2 lists the predefined non-administrative user accounts that Oracle Database automatically creates when you run standard scripts (such as the various <code>cat*.sql</code> scripts). You can find a complete list of user accounts that are created and maintained by Oracle by querying the <code>USERNAME</code> and <code>ORACLE_MAINTAINED</code> columns of the <code>ALL_USERS</code> data dictionary view. If the output for <code>ORACLE_MAINTAINED</code> is <code>Y</code>, then you must not modify the user account except by running the script that was used to create it.

Non-administrative user accounts only have the minimum privileges needed to perform their jobs. Their default tablespace is USERS. In a multitenant environment, the predefined non-administrative accounts reside in the root database

To protect these accounts from unauthorized access, the installation process locks and expires these accounts immediately after installation, except where noted in the following table. As the database administrator, you are responsible for unlocking and resetting these accounts.

To find the status of an account, such as whether it is open, locked, or expired, query the ACCOUNT_STATUS column of the DBA_USERS data dictionary view. If the account is schema only, then the status is NONE.

Table 2-2 Predefined Oracle Database Non-Administrative User Accounts

User Account	Description
DIP	The Oracle Directory Integration and Provisioning (DIP) account that is installed with Oracle Label Security. This profile is created automatically as part of the installation process for Oracle Internet Directory-enabled Oracle Label Security.
	See Oracle Label Security Administrator's Guide.
MDDATA	The schema used by Oracle Spatial for storing Geocoder and router data.
	Oracle Spatial provides a SQL schema and functions that enable you to store, retrieve, update, and query collections of spatial features in an Oracle database.
	See Oracle Spatial and Graph Developer's Guide.
ORACLE_OCM	The account used with Oracle Configuration Manager. This feature enables you to associate the configuration information for the current Oracle Database instance with My Oracle Support. Then when you log a service request, it is associated with the database instance configuration information.
	See Oracle Database Installation Guide for your platform.
XS\$NULL	An internal account that represents the absence of database user in a session and the actual session user is an application user supported by Oracle Real Application Security. XS\$NULL has no privileges and does not own any database object. No one can authenticate as XS\$NULL, nor can authentication credentials ever be assigned to XS\$NULL.



2.6.4 Predefined Sample Schema User Accounts

Oracle Database creates a set of sample user accounts if you install the sample schemas.

The sample schema user accounts are all non-administrative accounts, and their tablespace is users.

To protect these accounts from unauthorized access, the installation process locks and expires these accounts immediately after installation. As the database administrator, you are responsible for unlocking and resetting these accounts.

Table 2-3 lists the sample schema user accounts, which represent different divisions of a fictional company that manufactures various products. You can find the status of these accounts by querying the <code>DBA_USERS</code> data dictionary view. Because the <code>ORACLE_MAINTAINED</code> column output for these accounts is N, you can modify these accounts without re-running the scripts that were used to create them.

To find the status of an account, such as whether it is open, locked, or expired, query the ACCOUNT_STATUS column of the DBA_USERS data dictionary view. If the account is schema only, then the status is NONE.

Table 2-3 Default Sample Schema User Accounts

User Account	Description
HR	The account used to manage the ${\tt HR}$ (Human Resources) schema. This schema stores information about the employees and the facilities of the company.
OE	The account used to manage the OE (Order Entry) schema. This schema stores product inventories and sales of the company's products through various channels.
PM	The account used to manage the PM (Product Media) schema. This schema contains descriptions and detailed information about each product sold by the company.
IX	The account used to manage the ${\tt IX}$ (Information Exchange) schema. This schema manages shipping through business-to-business (B2B) applications.
SH	The account used to manage the SH (Sales) schema. This schema stores business statistics to facilitate business decisions.

In addition to the sample schema accounts, Oracle Database provides another sample schema account, SCOTT. The SCOTT schema contains the tables EMP, DEPT, SALGRADE, and BONUS. The SCOTT account is used in examples throughout the Oracle Database documentation set. When you install Oracle Database, the SCOTT account is locked and expired.

Related Topics

Oracle Database Sample Schemas

2.7 Database User and Profile Data Dictionary Views

Oracle Database provides a set of data dictionary views that provide information about the settings that you used to create users and profiles.

Data Dictionary Views That List Information About Users and Profiles
 Oracle Database provides a set of data dictionary views that contain information about database users and profiles.



Query to Find All Users and Associated Information

The DBA_USERS data dictionary view shows all users and their associated information as defined in the database.

Query to List All Tablespace Quotas

The \DBA_TS_QUOTAS data dictionary view lists all tablespace quotas assigned to each user.

Query to List All Profiles and Assigned Limits

The DBA_PROFILE view lists all profiles in the database and associated settings for each limit in each profile.

Query to View Memory Use for Each User Session

The V\$SESSION dynamic view lists the memory use for each user session.

2.7.1 Data Dictionary Views That List Information About Users and Profiles

Oracle Database provides a set of data dictionary views that contain information about database users and profiles.

Table 2-4 lists these data dictionary views.

Table 2-4 Data Dictionary Views That Display Information about Users and Profiles

View	Description
ALL_OBJECTS	Describes all objects accessible to the current user
ALL_USERS	Lists users visible to the current user, but does not describe them
DBA_PROFILES	Displays all profiles and their limits
DBA_TS_QUOTAS	Describes tablespace quotas for users
DBA_OBJECTS	Describes all objects in the database
DBA_USERS	Describes all users of the database
DBA_USERS_WITH_DEFPWD	Lists all user accounts that have default passwords
PROXY_USERS	Describes users who can assume the identity of other users
RESOURCE_COST	Lists the cost for each resource in terms of CPUs for each session, reads for each session, connection times, and SGA
USER_PASSWORD_LIMITS	Describes the password profile parameters that are assigned to the user
USER_RESOURCE_LIMITS	Displays the resource limits for the current user
USER_TS_QUOTAS	Describes tablespace quotas for users
USER_OBJECTS	Describes all objects owned by the current user
USER_USERS	Describes only the current user
V\$SESSION	Lists session information for the current database session
V\$SESSTAT	Displays user session statistics
V\$STATNAME	Displays decoded statistic names for the statistics shown in the V\$SESSTAT view

The following sections present examples of using these views. These examples assume that the following statements have been run. The users are all local users.

CREATE PROFILE clerk LIMIT SESSIONS_PER_USER 1 IDLE TIME 30



```
CONNECT TIME 600;
CREATE USER jfee
IDENTIFIED BY password
DEFAULT TABLESPACE example
TEMPORARY TABLESPACE temp
QUOTA 500K ON example
PROFILE clerk
CONTAINER = CURRENT;
CREATE USER dcranney
IDENTIFIED BY password
DEFAULT TABLESPACE example
TEMPORARY TABLESPACE temp
QUOTA unlimited ON example
CONTAINER = CURRENT;
CREATE USER userscott
IDENTIFIED BY password
CONTAINER = CURRENT;
```

Related Topics

Oracle Database Reference

2.7.2 Query to Find All Users and Associated Information

The DBA_USERS data dictionary view shows all users and their associated information as defined in the database.

For detailed information about the DBA USERS view, see Oracle Database Reference.

For example:

```
col username format a11
col profile format a10
col account_status format a19
col authentication type format a29
```

SELECT USERNAME, PROFILE, ACCOUNT STATUS, AUTHENTICATION TYPE FROM DBA USERS;

USERNAME	PROFILE	ACCOUNT_STATUS	${\tt AUTHENTICATION_TYPE}$
SYS	DEFAULT	OPEN	PASSWORD
SYSTEM	DEFAULT	OPEN	PASSWORD
USERSCOTT	DEFAULT	OPEN	PASSWORD
JFEE	CLERK	OPEN	GLOBAL
DCRANNEY	DEFAULT	OPEN	EXTERNAL

2.7.3 Query to List All Tablespace Quotas

The DBA_TS_QUOTAS data dictionary view lists all tablespace quotas assigned to each user.

For detailed information about this view, see Oracle Database Reference.

For example:

```
SELECT * FROM DBA_TS_QUOTAS;

TABLESPACE USERNAME BYTES MAX_BYTES BLOCKS MAX_BLOCKS
```



EXAMPLE	JFEE	0	512000	0	250
EXAMPLE	DCRANNEY	0	-1	0	-1

When specific quotas are assigned, the exact number is indicated in the MAX_BYTES column. This number is always a multiple of the database block size, so if you specify a tablespace quota that is not a multiple of the database block size, then it is rounded up accordingly. Unlimited quotas are indicated by -1.

2.7.4 Query to List All Profiles and Assigned Limits

The DBA_PROFILE view lists all profiles in the database and associated settings for each limit in each profile.

For example:

SELECT * FROM DBA_PROFILES
 ORDER BY PROFILE;

PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT
CLERK	COMPOSITE LIMIT	KERNEL	DEFAULT
CLERK	FAILED LOGIN ATTEMPTS	PASSWORD	DEFAULT
CLERK	PASSWORD_LIFE_TIME	PASSWORD	DEFAULT
CLERK	PASSWORD REUSE TIME	PASSWORD	DEFAULT
CLERK	PASSWORD REUSE MAX	PASSWORD	DEFAULT
CLERK	PASSWORD_VERIFY_FUNCTION	PASSWORD	DEFAULT
CLERK	PASSWORD_LOCK_TIME	PASSWORD	DEFAULT
CLERK	PASSWORD_GRACE_TIME	PASSWORD	DEFAULT
CLERK	PRIVATE_SGA	KERNEL	DEFAULT
CLERK	CONNECT_TIME	KERNEL	600
CLERK	IDLE_TIME	KERNEL	30
CLERK	LOGICAL_READS_PER_CALL	KERNEL	DEFAULT
CLERK	LOGICAL_READS_PER_SESSION	KERNEL	DEFAULT
CLERK	CPU_PER_CALL	KERNEL	DEFAULT
CLERK	CPU_PER_SESSION	KERNEL	DEFAULT
CLERK	SESSIONS_PER_USER	KERNEL	1
DEFAULT	COMPOSITE_LIMIT	KERNEL	UNLIMITED
DEFAULT	PRIVATE_SGA	KERNEL	UNLIMITED
DEFAULT	SESSIONS_PER_USER	KERNEL	UNLIMITED
DEFAULT	CPU_PER_CALL	KERNEL	UNLIMITED
DEFAULT	LOGICAL_READS_PER_CALL	KERNEL	UNLIMITED
DEFAULT	CONNECT_TIME	KERNEL	UNLIMITED
DEFAULT	IDLE_TIME	KERNEL	UNLIMITED
DEFAULT	LOGICAL_READS_PER_SESSION	KERNEL	UNLIMITED
DEFAULT	CPU_PER_SESSION	KERNEL	UNLIMITED
DEFAULT	FAILED_LOGIN_ATTEMPTS	PASSWORD	10
DEFAULT	PASSWORD_LIFE_TIME	PASSWORD	180
DEFAULT	PASSWORD_REUSE_MAX		UNLIMITED
DEFAULT	PASSWORD_LOCK_TIME		1
DEFAULT	PASSWORD_GRACE_TIME	PASSWORD	7
DEFAULT	PASSWORD_VERIFY_FUNCTION		UNLIMITED
DEFAULT	PASSWORD_REUSE_TIME	PASSWORD	UNLIMITED
DEFAULT	INACTIVE_ACCOUNT_TIME		UNLIMITED
DEFAULT	PASSWORD_ROLLOVER_TIME	PASSWORD	0

34 rows selected.

To find the default profile values, you can run the following query:

SELECT * FROM DBA PROFILES WHERE PROFILE = 'DEFAULT';



PROFILE	RESOURCE_NAME	RESOURCE_TYPE	LIMIT
	GOVED TIME		
DEFAULT	COMPOSITE_LIMIT	KERNEL	UNLIMITED
DEFAULT	SESSIONS_PER_USER	KERNEL	UNLIMITED
DEFAULT	CPU_PER_SESSION	KERNEL	UNLIMITED
DEFAULT	CPU_PER_CALL	KERNEL	UNLIMITED
DEFAULT	LOGICAL_READS_PER_SESSION	KERNEL	UNLIMITED
DEFAULT	LOGICAL_READS_PER_CALL	KERNEL	UNLIMITED
DEFAULT	IDLE_TIME	KERNEL	UNLIMITED
DEFAULT	CONNECT_TIME	KERNEL	UNLIMITED
DEFAULT	PRIVATE_SGA	KERNEL	UNLIMITED
DEFAULT	FAILED_LOGIN_ATTEMPTS	PASSWORD	10
DEFAULT	PASSWORD_LIFE_TIME	PASSWORD	180
DEFAULT	PASSWORD_REUSE_TIME	PASSWORD	UNLIMITED
DEFAULT	PASSWORD_REUSE_MAX	PASSWORD	UNLIMITED
DEFAULT	PASSWORD_VERIFY_FUNCTION	PASSWORD	NULL
DEFAULT	PASSWORD_LOCK_TIME	PASSWORD	1
DEFAULT	PASSWORD_GRACE_TIME	PASSWORD	7

16 rows selected.

Related Topics

Oracle Database Reference

2.7.5 Query to View Memory Use for Each User Session

The V\$SESSION dynamic view lists the memory use for each user session.

For detailed information on this view, see Oracle Database Reference.

The following query lists all current sessions, showing the Oracle Database user and current User Global Area (UGA) memory use for each session:

```
SELECT USERNAME, VALUE || 'bytes' "Current UGA memory"
  FROM V$SESSION sess, V$SESSTAT stat, V$STATNAME name
WHERE sess.SID = stat.SID
  AND stat.STATISTIC# = name.STATISTIC#
  AND name.NAME = 'session uga memory';
USERNAME
                         Current UGA memory
_____
                         18636bytes
                         17464bytes
                          19180bytes
                          18364bytes
                          39384bytes
                          35292bytes
                          17696bytes
                          15868bytes
USERSCOTT
                          42244bytes
SYS
                          98196bytes
```

30648bytes

11 rows selected.

SYSTEM

To see the maximum UGA memory allocated to each session since the instance started, replace 'session uga memory' in the preceding query with 'session uga memory max'.

Configuring Authentication

Authentication means to verify the identity of users or other entities that connect to the database.

About Authentication

Authentication means verifying the identity of a user, device, or other entity who wants to use data, resources, or applications.

Configuring Password Protection

You can secure user passwords in a variety of ways, such as controlling the password creation requirements or using password management policies.

Authentication of Database Administrators

You can authenticate database administrators by using strong authentication, from the operating system, or from the database using passwords.

Database Authentication of Users

Database authentication of users entails using information within the database itself to perform the authentication.

Schema-Only Accounts

You can create schema-only accounts, that is, the schema user has no password.

Operating System Authentication of Users

Oracle Database can authenticate by using information that is maintained by the operating system.

Network Authentication of Users

You can authenticate users over a network by using Transport Layer Security with third-party services.

Configuring Operating System Users for a PDB

The DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure configures user accounts to be operating system users for a PDB.

Global User Authentication and Authorization

Global user authentication and authorization enables you to centralize the management of user-related information.

Configuring an External Service to Authenticate Users and Passwords

An external service (the operating system or the network) can administer passwords and authenticate users.

Multitier Authentication and Authorization

Oracle Database secures middle-tier applications by limiting privileges, preserving client identities through all tiers, and auditing actions by clients.

Administration and Security in Clients, Application Servers, and Database Servers In a multitier environment, an application server provides data for clients and serves as

In a multitier environment, an application server provides data for clients and serves as an interface to one or more database servers.

Preserving User Identity in Multitiered Environments

You can use middle tier servers for proxy authentication and client identifiers to identify application users who are not known to the database.

User Authentication Data Dictionary Views

Oracle Database provides data dictionary views that list information about user authentication, such as roles that users have or profiles they use.

3.1 About Authentication

Authentication means verifying the identity of a user, device, or other entity who wants to use data, resources, or applications.

Validating this identity establishes a trust relationship for further interactions. Authentication also enables accountability by making it possible to link access and actions to specific identities. After authentication, authorization processes can allow or limit the levels of access and action permitted to that entity.

You can authenticate both database and nondatabase users for an Oracle database. For simplicity, the same authentication method is generally used for all database users, but Oracle Database allows a single database instance to use any or all methods. Oracle Database requires special authentication procedures for database administrators, because they perform special database operations. Oracle Database also encrypts passwords during transmission to ensure the security of network authentication.

After authentication, authorization processes can allow or limit the levels of access and action permitted to that entity.

Related Topics

Configuring Privilege and Role Authorization
 Privilege and role authorization controls the permissions that users have to perform day-to-day tasks.

3.2 Configuring Password Protection

You can secure user passwords in a variety of ways, such as controlling the password creation requirements or using password management policies.

- What Are the Oracle Database Built-in Password Protections?
 Oracle Database provides a set of built-in password protections designed to protect your users' passwords.
- Minimum Requirements for Passwords
 Oracle provides a set of minimum requirements for passwords.
- Creating a Password by Using the IDENTIFIED BY Clause
 SQL statements that accept the IDENTIFIED BY clause also enable you to create passwords.
- Using a Password Management Policy
 A password management policy can create and enforce a set of restrictions that can better secure user passwords.
- Managing Gradual Database Password Rollover for Applications
 A gradual database password rollover enables the database password of an application to be updated while avoiding application downtime while the new password is propagated to application clients, by allowing the older password to remain valid for a specified period.
- Managing the Complexity of Passwords
 Oracle Database provides a set of functions that you can use to manage the complexity of passwords.



- Managing Password Case Sensitivity
 - You can manage the password case sensitivity for passwords from user accounts from previous releases.
- Ensuring Against Password Security Threats by Using the 12C Password Version
 The 12C password version enables users to create complex passwords that meet
 compliance standards.
- Managing the Secure External Password Store for Password Credentials
 The secure external password store is a client-side wallet that is used to store password credentials.
- Managing Passwords for Administrative Users
 The passwords of administrative users have special protections, such as password files and password complexity functions.

3.2.1 What Are the Oracle Database Built-in Password Protections?

Oracle Database provides a set of built-in password protections designed to protect your users' passwords.

These password protections are as follows:

- Password encryption. Oracle Database automatically and transparently encrypts passwords during network (client-to-server and server-to-server) connections, using Advanced Encryption Standard (AES) before sending them across the network. However, a password that is specified within a SQL statement (such as CREATE USER user_name IDENTIFIED BY password;) is still transmitted across the network in clear text in the network trace files. For this reason, you should have native network encryption enabled or configure Transport Layer Security (TLS) encryption.
- Password complexity checking. In a default installation, Oracle Database provides the
 oral2c_verify_function and oral2c_strong_verify_function password verification
 functions to ensure that new or changed passwords are sufficiently complex to prevent
 intruders who try to break into the system by guessing passwords. You must manually
 enable password complexity checking. You can further customize the complexity of your
 users' passwords.
- Preventing passwords from being broken. If a user tries to log in to Oracle Database multiple times using an incorrect password, Oracle Database delays each login by one second. This protection applies for attempts made from different IP addresses or multiple client connections. This feature significantly decreases the number of passwords that an intruder would be able to try within a fixed time period when attempting to log in. The failed login delay slows down each failed login attempt, increasing the overall time that is required to perform a password-guessing attack, because such attacks usually require a very large number of failed login attempts.

For non-administrative logins, Oracle Database protects against concurrent password guessing attacks by setting an exclusive lock for the failed login delay. This prevents an intruder from attempting to sidestep the failed login delay when the intruder tries the next concurrent guess in a different database session as soon as the first guess fails and is delayed.

By holding an exclusive lock on the account that is being attacked, Oracle Database mitigates concurrent password guessing attacks, but this can simultaneously leave the account vulnerable to denial-of-service (DoS) attacks. To remedy this problem, you should create a password profile where the FAILED_LOGIN_ATTEMPTS parameter is set to UNLIMITED, and then apply this password profile to the user account. The value UNLIMITED for the FAILED_LOGIN_ATTEMPTS parameter setting disables failed login delays and does not



limit the number of failed login attempts. For these types of accounts, Oracle recommends that you use a long random password.

The concurrent password-guessing attack protection does not apply to administrative user connections, because these kinds of connections must remain available at all times and be immune to denial-of-service attacks. Hence, Oracle recommends that you choose long passwords for any administrative privileged account.

- Enforced case sensitivity for passwords. Passwords are case sensitive. For example, the password hpp5620qr fails if it is entered as hpp5620qR or hpp5620qr. Case sensitivity affects password files and database links.
- Passwords hashed using the 12C password version. To verify the user's password and enforce case sensitivity in password creation, Oracle Database uses the 12C password version, which is based on a de-optimized algorithm that involves Password-Based Key Derivation Function (PBKDF2) and the SHA-512 cryptographic hash functions.

Related Topics

Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.

3.2.2 Minimum Requirements for Passwords

Oracle provides a set of minimum requirements for passwords.

Passwords can be at most 30 bytes long. There are a variety of ways that you can secure passwords, ranging from requiring passwords to be of a sensible length to creating custom password complexity verification scripts that enforce the password complexity policy requirements that apply at your site.

Related Topics

Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.

3.2.3 Creating a Password by Using the IDENTIFIED BY Clause

SQL statements that accept the IDENTIFIED BY clause also enable you to create passwords.

• To create passwords for users, use the CREATE USER, ALTER USER, GRANT CREATE SESSION, or CREATE DATABASE LINK SQL statement.

The following SQL statements create passwords with the IDENTIFIED BY clause.

```
CREATE USER psmith IDENTIFIED BY password;
GRANT CREATE SESSION TO psmith IDENTIFIED BY password;
ALTER USER psmith IDENTIFIED BY password;
CREATE DATABASE LINK AUTHENTICATED BY psmith IDENTIFIED BY password;
```

Related Topics

About Password Complexity Verification

Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.

3.2.4 Using a Password Management Policy

A password management policy can create and enforce a set of restrictions that can better secure user passwords.

About Managing Passwords

Database security systems that depend on passwords require that passwords be kept secret at all times.

Finding User Accounts That Have Default Passwords

The DBA_USERS_WITH_DEFPWD data dictionary view can find user accounts that use default passwords.

Password Settings in the Default Profile

A profile is a collection of parameters that sets limits on database resources.

Using the ALTER PROFILE Statement to Set Profile Limits

You can modify profile limits such as failed login attempts, password lock times, password reuse, and several other settings.

Disabling and Enabling the Default Password Security Settings

Oracle provides scripts that you can use to disable and enable the default password security settings.

Automatically Locking Inactive Database User Accounts

The INACTIVE_ACCOUNT_TIME profile parameter locks a user account that has not logged in to the database instance in a specified number of days.

- Automatically Locking User Accounts After a Specified Number of Failed Log-in Attempts
 Oracle Database can lock a user's account after a specified number of consecutive failed
 log-in attempts.
- Example: Locking an Account with the CREATE PROFILE Statement

The CREATE PROFILE statement can lock user accounts if a user's attempt to log in violates the CREATE PROFILE settings.

Explicitly Locking a User Account

When you explicitly lock a user account, the account cannot be unlocked automatically. Only a security administrator can unlock the account.

Controlling the User Ability to Reuse Previous Passwords

You can ensure that users do not reuse previous passwords for an amount of time or for a number of password changes.

About Controlling Password Aging and Expiration

You can specify a password lifetime, after which the password expires.

- Using the CREATE PROFILE or ALTER PROFILE Statement to Set a Password Lifetime When you set a lifetime for a password, the user must create a new password when this lifetime ends.
- Checking the Status of a User Account

You can check the status of any account, whether it is open, in grace, or expired.

Password Change Life Cycle

After a password is created, it follows a life cycle and grace period in four phases.

PASSWORD LIFE TIME Profile Parameter Low Value

Be careful if you set the PASSWORD_LIFE_TIME parameter of CREATE PROFILE or ALTER PROFILE to a low value (for example, 1 day).

3.2.4.1 About Managing Passwords

Database security systems that depend on passwords require that passwords be kept secret at all times.

Because passwords are vulnerable to theft and misuse, Oracle Database uses a password management policy. Database administrators and security officers control this policy through user profiles, enabling greater control of database security.

You can use the CREATE PROFILE statement to create a user profile. The profile is assigned to a user with the CREATE USER OF ALTER USER statement.

3.2.4.2 Finding User Accounts That Have Default Passwords

The DBA_USERS_WITH_DEFPWD data dictionary view can find user accounts that use default passwords.

When you create a database, most of the default accounts are locked with the passwords expired. If you have upgraded from an earlier release of Oracle Database, then you may have user accounts that have default passwords. These are default accounts that are created when you create a database, such as the HR, OE, and SCOTT accounts.

For greater security, you should change the passwords for these accounts. Using a default password that is commonly known can make your database vulnerable to attacks by intruders.

1. Log in to the database instance using SQL*Plus with the SYSDBA administrative privilege.

For example:

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

2. Query the DBA USERS WITH DEFPWD data dictionary view.

For example, to find both the names of accounts that have default passwords and the status of the account:

3. Change the passwords for any accounts that the DBA USERS WITH DEFPWD view lists.

Oracle recommends that you do **not** assign these accounts passwords that they may have had in previous releases of Oracle Database.

For example:

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

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

3.2.4.3 Password Settings in the Default Profile

A profile is a collection of parameters that sets limits on database resources.

If you assign the profile to a user, then that user cannot exceed these limits. You can use profiles to configure database settings such as sessions per user, logging and tracing features, and so on. Profiles can also control user passwords. To find information about the current password settings in the profile, you can query the DBA PROFILES data dictionary view.



Table 3-1 lists the password-specific parameter settings in the default profile.

Table 3-1 Password-Specific Settings in the Default Profile

Parameter	Default Setting	Description
INACTIVE_ACCOUNT_TIME	UNLIMITED	Locks the account of a database user who has not logged in to the database instance in a specified number of days.
FAILED_LOGIN_ATTEMPTS	10	Sets the maximum times a user try to log in and to fail before locking the account.
		 When you set this parameter, take into consideration users who may log in using the CONNECT THROUGH privilege.
		You can set limits on the number of times an unauthorized user (possibly an intruder) attempts to log in to Oracle Call Interface (OCI) applications by using the SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter.
PASSWORD_GRACE_TIME	7	Sets the number of days that a user has to change their password before it expires.
PASSWORD_LIFE_TIME	180	Sets the number of days the user can use their current password.
PASSWORD_LOCK_TIME	1	Sets the number of days an account will be locked after the specified number of consecutive failed login attempts. After the time passes, then the account becomes unlocked. This user's profile parameter is useful to help prevent brute force attacks on user passwords but not to increase the maintenance burden on administrators.
		Even after the value set by PASSWORD_LOCK_TIME shows that the password has expired, the DBA_USERS data dictionary view will show that the account is locked. However, after the user connects, the information in DBA_USERS is updated with the correct OPEN status.
PASSWORD_REUSE_MAX	UNLIMITED	Sets the number of password changes required before the current password can be reused.
PASSWORD_REUSE_TIME	UNLIMITED	Sets the number of days before which a password cannot be reused.
PASSWORD_ROLLOVER_TIME	0	Enables the gradual database password rollover time.

Related Topics

- Managing Resources with Profiles
 - A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.
- Automatically Locking Inactive Database User Accounts
 - The INACTIVE_ACCOUNT_TIME profile parameter locks a user account that has not logged in to the database instance in a specified number of days.



- Configuration of the Maximum Number of Authentication Attempts
 The SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter sets the number of authentication attempts before the database will drop a failed connection.
- Automatically Locking User Accounts After a Specified Number of Failed Log-in Attempts
 Oracle Database can lock a user's account after a specified number of consecutive failed
 log-in attempts.
- About Controlling Password Aging and Expiration
 You can specify a password lifetime, after which the password expires.
- Controlling the User Ability to Reuse Previous Passwords
 You can ensure that users do not reuse previous passwords for an amount of time or for a number of password changes.
- Managing Resources with Profiles
 A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.
- Managing Resources with Profiles
 A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.

3.2.4.4 Using the ALTER PROFILE Statement to Set Profile Limits

You can modify profile limits such as failed login attempts, password lock times, password reuse, and several other settings.

These settings are described in Table 3-1. For greater security, use the default settings that are described in this table, based on your needs.

Use the ALTER PROFILE statement to modify a user's profile limits.

For example:

```
ALTER PROFILE prof LIMIT

FAILED_LOGIN_ATTEMPTS 9

PASSWORD_LOCK_TIME 10

INACTIVE ACCOUNT TIME 21;
```

3.2.4.5 Disabling and Enabling the Default Password Security Settings

Oracle provides scripts that you can use to disable and enable the default password security settings.

If your applications use the default password security settings from Oracle Database 10g release 2 (10.2), then you can revert to these settings until you modify the applications to use the default password security settings from Oracle Database 11g or later.

- Modify your applications to conform to the password security settings from Oracle Database 11q or later.
- 2. Update your database to use the security configuration that suits your business needs, using one of the following methods:
 - Manually update the database security configuration.
 - Run the secconf.sql script to apply the default password settings from Oracle
 Database 11g or later. You can customize this script to have different security settings
 if you like, but remember that the settings listed in the original script are Oracle recommended settings.



If you created your database manually, then you should run the <code>secconf.sql</code> script to apply the Oracle default password settings to the database. Databases that have been created with Database Configuration Assistant (DBCA) will have these settings, but manually created databases do not.

The secconf.sql script is in the <code>\$ORACLE_HOME/rdbms/admin</code> directory. The secconf.sql script affects both password and audit settings. It has no effect on other security settings.

3.2.4.6 Automatically Locking Inactive Database User Accounts

The INACTIVE_ACCOUNT_TIME profile parameter locks a user account that has not logged in to the database instance in a specified number of days.

Users are considered active users if they log in periodically. The INACTIVE_ACCOUNT_TIME timing is based on the number of days after the last time a user successfully logs in.

 To lock user accounts automatically after a specified number of days, set the INACTIVE_ACCOUNT_TIME profile parameter in the CREATE PROFILE or ALTER PROFILE statement.

Note the following:

- The default value for INACTIVE ACCOUNT TIME is UNLIMITED.
- You must specify a whole number for the number of days. The minimum setting is 15 and the maximum is 24855.
- To set the user's account to have an unlimited inactivity time, set the INACTIVE ACCOUNT TIME to UNLIMITED.
- To set the user's account to use the time specified by the default profile, set INACTIVE ACCOUNT TIME to DEFAULT.
- You can set this parameter for all database authenticated users, including administrative users, but not for external or global authenticated users.
- In a read-only database, the last successful login is not considered in the
 INACTIVE_ACCOUNT_TIME timing. It is not possible to lock a user account in a read-only
 database (except by performing consecutive failed logins equal in number to the
 account's FAILED LOGIN ATTEMPTS password profile setting).
- For a newly created user account, the timing begins at account creation time. When this user logs out and then logs again, the timing starts when the user successfully logs in.
- In a multitenant environment, the INACTIVE_ACCOUNT_TIME setting applies to the last time a common user logs in to the root. A common user is considered active if this user logs in to any of the PDBs or the root.
- For a proxy user account login, the INACTIVE_ACCOUNT_TIME begins the timing when the proxy user logs in successfully.

For example, to create a profile that locks an account after 60 days of being inactive:

```
CREATE PROFILE time_limit LIMIT INACTIVE ACCOUNT TIME 60;
```



3.2.4.7 Automatically Locking User Accounts After a Specified Number of Failed Login Attempts

Oracle Database can lock a user's account after a specified number of consecutive failed log-in attempts.

• To lock user accounts automatically after a specified time interval or to require database administrator intervention to be unlocked, set the PASSWORD_LOCK_TIME profile parameter in the CREATE PROFILE or ALTER PROFILE statement.

For example, to set the time interval to 10 days:

```
PASSWORD LOCK TIME = 10
```

Note the following:

- You can lock accounts manually, so that they must be unlocked explicitly by a database administrator.
- You can specify the permissible number of failed login attempts by using the CREATE PROFILE statement. You can also specify the amount of time an account remains locked.
- Each time the user unsuccessfully logs in, Oracle Database increases the delay exponentially with each login failure.
- If you do not specify a time interval for unlocking the account, then PASSWORD_LOCK_TIME assumes the value specified in a default profile. (The recommended value is 1 day.) If you specify PASSWORD_LOCK_TIME as UNLIMITED, then you must explicitly unlock the account by using an ALTER USER statement. For example, assuming that PASSWORD_LOCK_TIME UNLIMITED is specified for johndoe, then you use the following statement to unlock the johndoe account:

```
ALTER USER johndoe ACCOUNT UNLOCK;
```

- After a user successfully logs into an account, Oracle Database resets the unsuccessful login attempt count for the user. If it is non-zero, then the count is set to zero.
- In a multitenant environment, a locked CDB common user account will be locked across all PDBs in the CDB. A locked application common user account will be locked across all PDBs that are associated with the application root.

3.2.4.8 Example: Locking an Account with the CREATE PROFILE Statement

The CREATE PROFILE statement can lock user accounts if a user's attempt to log in violates the CREATE PROFILE settings.

Example 3-1 sets the maximum number of failed login attempts for the user johndoe to 10 (the default), and the amount of time the account locked to 30 days. The account will unlock automatically after 30 days.

Example 3-1 Locking an Account with the CREATE PROFILE Statement

```
CREATE PROFILE prof LIMIT
FAILED_LOGIN_ATTEMPTS 10
PASSWORD_LOCK_TIME 30
```

ALTER USER johndoe PROFILE prof;

3.2.4.9 Explicitly Locking a User Account

When you explicitly lock a user account, the account cannot be unlocked automatically. Only a security administrator can unlock the account.

In a multitenant environment, after you have locked a CDB common user account in the CDB root, this user cannot log in to any PDB that is associated with this root, nor can this account be unlocked in a PDB. In addition, you can lock a CDB common account locally in a PDB, which will prevent the CDB common user from logging in to that PDB. Similarly, an application common user account that is locked in the application root cannot log in to any PDB associated with the application root, nor can the application common user be unlocked in an application PDB. You can explicitly lock an application common user locally in an application PDB.

To explicitly lock a user account, use the CREATE USER or ALTER USER statement.

For example, the following statement locks the user account, susan:

ALTER USER susan ACCOUNT LOCK;

3.2.4.10 Controlling the User Ability to Reuse Previous Passwords

You can ensure that users do not reuse previous passwords for an amount of time or for a number of password changes.

• To ensure that users cannot reuse their passwords for a specified period of time, configure the rules for password reuse with the CREATE PROFILE or ALTER PROFILE statements.

The following table lists the CREATE PROFILE and ALTER PROFILE parameters that control ability of a user to reuse a previous password.

Table 3-2 Parameters Controlling Reuse of a Previous Password

Parameter Name	Description and Use	
PASSWORD_REUSE_TIME	Requires either of the following:	
	 A number specifying how many days (or a fraction of a day) between the earlier use of a password and its next use The word UNLIMITED 	
PASSWORD_REUSE_MAX	Requires either of the following:	
	 An integer to specify the number of password changes required before a password can be reused The word UNLIMITED 	

If you do not specify a parameter, then the user can reuse passwords at any time, which is not a good security practice.

If neither parameter is <code>UNLIMITED</code>, then password reuse is allowed, but only after meeting both conditions. The user must have changed the password the specified number of times, and the specified number of days must have passed since the previous password was last used.

For example, suppose that the profile of user A had PASSWORD_REUSE_MAX set to 10 and PASSWORD_REUSE_TIME set to 30. User A cannot reuse a password until he or she has reset the password 10 times, and until 30 days had passed since the password was last used.

If either parameter is specified as UNLIMITED, then the user can never reuse a password.



If you set both parameters to UNLIMITED, then Oracle Database ignores both, and the user can reuse any password at any time.



If you specify DEFAULT for either parameter, then Oracle Database uses the value defined in the DEFAULT profile, which sets all parameters to UNLIMITED. Oracle Database thus uses UNLIMITED for any parameter specified as DEFAULT, unless you change the setting for that parameter in the DEFAULT profile.

Related Topics

Oracle Database SQL Language Reference

3.2.4.11 About Controlling Password Aging and Expiration

You can specify a password lifetime, after which the password expires.

This means that the next time the user logs in with the current, correct password, he or she is prompted to change the password. By default, there are no complexity or password history checks, so users can still reuse any previous or weak passwords. You can control these factors by setting the Password_Reuse_time, Password_reuse_max, and Password_verify_function parameters.

In addition, you can set a grace period, during which each attempt to log in to the database account receives a warning message to change the password. If the user does not change it by the end of that period, then Oracle Database expires the account.

As a database administrator, you can manually set the password state to be expired, which sets the account status to EXPIRED. The user must then follow the prompts to change the password before the logon can proceed.

For example, in SQL*Plus, suppose user SCOTT tries to log in with the correct credentials, but his password has expired. User SCOTT will then see the ORA-28001: The password has expired error and be prompted to change his password, as follows:

Changing password for scott
New password: new_password
Retype new password: new_password
Password changed.

Related Topics

- Controlling the User Ability to Reuse Previous Passwords
 You can ensure that users do not reuse previous passwords for an amount of time or for a number of password changes.
- About Password Complexity Verification
 Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.



3.2.4.12 Using the CREATE PROFILE or ALTER PROFILE Statement to Set a Password Lifetime

When you set a lifetime for a password, the user must create a new password when this lifetime ends.

Use the CREATE PROFILE or ALTER PROFILE statement to specify a lifetime for passwords.

The following example demonstrates how to create and assign a profile to user johndoe, and the PASSWORD_LIFE_TIME clause specifies that johndoe can use the same password for 180 days before it expires.

```
CREATE PROFILE prof LIMIT

FAILED_LOGIN_ATTEMPTS 4

PASSWORD_GRACE_TIME 3

PASSWORD_LIFE_TIME 180;

ALTER USER johndoe PROFILE prof;
```

Related Topics

Password Change Life Cycle
 After a password is created, it follows a life cycle and grace period in four phases.

3.2.4.13 Checking the Status of a User Account

You can check the status of any account, whether it is open, in grace, or expired.

• To check the status of a user account, query the ACCOUNT_STATUS column of the DBA_USERS data dictionary view.

For example:

```
SELECT ACCOUNT STATUS FROM DBA USERS WHERE USERNAME = 'username';
```

3.2.4.14 Password Change Life Cycle

After a password is created, it follows a life cycle and grace period in four phases.

The following diagram shows the life cycle of the password lifetime and grace period.

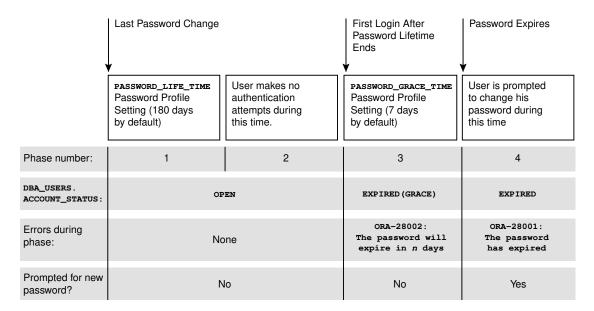


Figure 3-1 Password Change Life Cycle

In this figure:

- **Phase 1:** After the user account is created, or the password of an existing account is changed, the password lifetime period begins.
- Phase 2: This phase represents the period of time after the password lifetime ends but before the user logs in again with the correct password. The correct credentials are needed for Oracle Database to update the account status. Otherwise, the account status will remain unchanged. Oracle Database does not have any background process to update the account status. All changes to the account status are driven by the Oracle Database server process on behalf of authenticated users.
- Phase 3: When the user finally does log in, the grace period begins. Oracle Database then updates the DBA_USERS.EXPIRY_DATE column to a new value using the current time plus the value of the PASSWORD_GRACE_TIME setting from the account's password profile. At this point, the user receives an ORA-28002 warning message about the password expiring in the near future (for example, ORA-28002 The password will expire within 7 days if PASSWORD_GRACE_TIME is set to 7 days), but the user can still log in without changing the password. The DBA_USERS.EXPIRY_DATE column shows the time in the future when the user will be prompted to change their password.
- Phase 4: After the grace period (Phase 3) ends, the ORA-28001: The password has expired error appears, and the user is prompted to change the password after entering the current, correct password before the authentication can proceed. If the user has an Oracle Active Data Guard configuration, where there is a primary and a stand-by database, and the authentication attempt is made on the standby database (which is a read-only database), then the ORA-28032: Your password has expired and the database is set to read-only error appears. The user should log into the primary database and change the password there.

During any of these four phases, you can query the DBA_USERS data dictionary view to find the user's account status in the DBA_USERS.ACCOUNT_STATUS column.

In the following example, the profile assigned to johndoe includes the specification of a grace period: PASSWORD GRACE TIME = 3 (the recommended value). The first time johndoe tries to log

in to the database after 90 days (this can be *any* day after the 90th day, that is, the 91st day, 100th day, or another day), he receives a warning message that his password will expire in 3 days. If 3 days pass, and if he does not change his password, then the password expires. After this, he receives a prompt to change his password on any attempt to log in.

```
CREATE PROFILE prof LIMIT

FAILED_LOGIN_ATTEMPTS 4

PASSWORD_LIFE_TIME 90

PASSWORD_GRACE_TIME 3;

ALTER USER johndoe PROFILE prof;
```

A database administrator or a user who has the ALTER USER system privilege can explicitly expire a password by using the CREATE USER and ALTER USER statements. The following statement creates a user with an expired password. This setting forces the user to change the password before the user can log in to the database.

```
CREATE USER jbrown
IDENTIFIED BY password
...
PASSWORD EXPIRE;
```

There is no "password unexpire" clause for the CREATE USER statement, but an account can be "unexpired" by changing the password on the account.

3.2.4.15 PASSWORD LIFE TIME Profile Parameter Low Value

Be careful if you set the PASSWORD_LIFE_TIME parameter of CREATE PROFILE or ALTER PROFILE to a low value (for example, 1 day).

The PASSWORD_LIFE_TIME limit of a profile is measured from the last time that an account's password is changed, or the account creation time if the password has never been changed. These dates are recorded in the PTIME (password change time) and CTIME (account creation time) columns of the SYS.USER\$ system table. The PASSWORD_LIFE_TIME limit is not measured starting from the timestamp of the last change to the PASSWORD_LIFE_TIME profile parameter, as may be initially thought. Therefore, any accounts affected by the changed profile whose last password change time was more than PASSWORD_LIFE_TIME days ago immediately expire and enter their grace period on their next connection, issuing the ORA-28002: The password will expire within n days warning.

As a database administrator, you can find an account's last password change time as follows:

```
ALTER SESSION SET NLS_DATE_FORMAT='DD-MON-YYYY HH24:MI:SS';
SELECT PTIME FROM SYS.USER$ WHERE NAME = 'user name'; -- Password change time
```

To find when the account was created and the password expiration date, issue the following query:

```
SELECT CREATED, EXPIRY DATE FROM DBA USERS WHERE USERNAME = 'user name';
```

If the user who is assigned this profile is currently logged in when you set the PASSWORD_LIFE_TIME parameter and remains logged in, then Oracle Database does not change the user's account status from OPEN to EXPIRED (GRACE) when the currently listed expiration date passes. The timing begins only when the user logs into the database. You can check the user's last login time as follows:

```
SELECT LAST LOGIN FROM DBA USERS WHERE USERNAME = 'user name';
```



When making changes to a password profile, a database administrator must be aware that if some of the users who are subject to this profile are currently logged in to the Oracle database while their password profile is being updated by the administrator, then those users could potentially remain logged in to the system even beyond the expiration date of their password. You can find the currently logged in users by querying the USERNAME column of the V\$SESSION view.

This is because the expiration date of a user's password is based on the timestamp of the last password change on their account plus the value of the PASSWORD_LIFE_TIME password profile parameter set by the administrator. It is *not* based on the timestamp of the last change to the password profile itself.

Note the following:

- If the user is not logged in when you set PASSWORD_LIFE_TIME to a low value, then the user's account status does not change until the user logs in.
- You can set the PASSWORD_LIFE_TIME parameter to UNLIMITED, but this only affects
 accounts that have not entered their grace period. After the grace period expires, the user
 must change the password.

3.2.5 Managing Gradual Database Password Rollover for Applications

A gradual database password rollover enables the database password of an application to be updated while avoiding application downtime while the new password is propagated to application clients, by allowing the older password to remain valid for a specified period.

- About Managing Gradual Database Password Rollover for Applications
 You can configure a gradual database password rollover process to begin for database
 application clients when the database administrator changes the database password for
 the application.
- Password Change Life Cycle During a Gradual Database Password Rollover
 After a password is created or changed, it follows a life cycle and grace period in four
 phases.
- Enabling the Gradual Database Password Rollover
 To enable the gradual database password rollover, you must configure the PASSWORD ROLLOVER TIME user profile parameter.
- Changing a Password to Begin the Gradual Database Password Rollover Period
 After you have set a non-zero PASSWORD_ROLLOVER_TIME value, change the user's
 password and update the password with all the applications.
- Changing a Password During the Gradual Database Password Rollover Period After the rollover period has begun, you can still change the password.
- Ending the Password Rollover Period
 There are multiple ways in which you can end the password rollover period.
- Database Behavior During the Gradual Password Rollover Period
 Users can perform their standard password changes and logins during the password rollover period.
- Database Server Behavior After the Password Rollover Period Ends
 Oracle Database performs clean-up operations after the gradual database password rollover period ends.



- Guideline for Handling Compromised Passwords
 If a database account password is suspected of being compromised, then you should change the password immediately.
- How Gradual Database Password Rollover Works During Oracle Data Pump Exports
 When a user is exported while they are in the password rollover period, only the verifier
 corresponding to their new password is exported.
- Using Gradual Database Password Rollover in an Oracle Data Guard Environment
 In an Oracle Data Guard environment, you must set the ADG_ACCOUNT_INFO_TRACKING
 environment variable to GLOBAL to use gradual database password rollover.
- Finding Users Who Still Use Their Old Passwords
 You can perform a query that makes use of the AUTHENTICATION_TYPE field for a LOGIN
 audit record to find users who still use their old passwords.

3.2.5.1 About Managing Gradual Database Password Rollover for Applications

You can configure a gradual database password rollover process to begin for database application clients when the database administrator changes the database password for the application.

When the database or application administrator changes the password for the application in the database, the applications must be updated with the new database password. Setting the PASSWORD_ROLLOVER_TIME parameter in the user's profile enables a password change to take place without having to risk downtime or application outages that could occur as a result of an application attempting to use an outdated password. The password rollover takes place seamlessly from the server and works with all existing supported client versions.

The gradual database password rollover feature is designed for database accounts (service accounts) for applications. The application could be a single server (database client) or scaled out to multiple servers with multiple database clients. It is not designed for administrative users; hence, administrative users are restricted from using this feature, no matter which profile they are associated with. You cannot grant administrative privileges to users who have a password rollover-enabled profile.

You can configure the gradual database password rollover for native password-authenticated user connections. If you convert a password database account to a NO AUTHENTICATION account, then Oracle Database deletes the password and verifiers that are associated with this account. When a password-authenticated user account is converted to a GLOBAL, an EXTERNAL or a NO AUTHENTICATION account, then the user implicitly exits the password rollover period. Gradual password rollover supports the 11g password version and later.

You also can configure the gradual database password rollover for environments that use connected user database links. In this case, when you configure the gradual database password rollover, ensure that you also put the target account into rollover on the target of the connected user database link, and then roll over the target accounts on these links as well. To put the target account into rollover, you would use this syntax:

ALTER USER username IDENTIFIED BY same_new_rollover_password;

You cannot configure the gradual database password rollover for the following kinds of connections:

- Direct logins for Oracle Real Application Security users
- Kerberos-, certificate-, or RADIUS-based externally authenticated connections
- Centrally managed user (CMU) connections



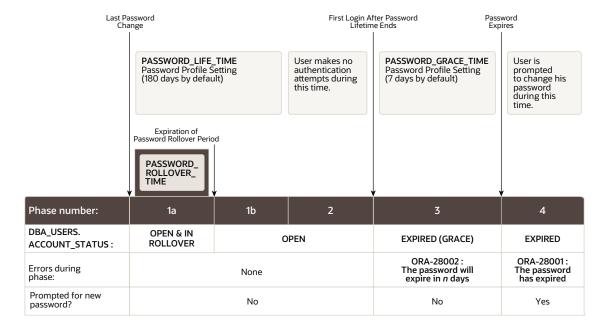
- Administrative connections that use external password files
- The Oracle Data Guard connection between the primary and the standby

3.2.5.2 Password Change Life Cycle During a Gradual Database Password Rollover

After a password is created or changed, it follows a life cycle and grace period in four phases.

The following diagram shows the life cycle of the password lifetime and grace period.

Figure 3-2 Password Change Life Cycle During a Gradual Database Password Rollover



In this figure:

- Phase 1: The password lifetime begins after the user account is created or when the
 password has been changed. When the password of an existing account is changed, and
 the user's profile has a non-zero PASSWORD_ROLLOVER_TIME value, then the password
 lifetime is composed of two phases, 1a and 1b:
 - Phase 1a begins with the password change. During Phase 1a, the user can log in using either the old password or the new password. The duration of phase 1a is normally Password_Rollover_Time, but if the administrator was able to update the password in all client applications sooner than this, they can decide to end the password rollover period sooner by issuing the following command, which makes the new password the only one that is accepted.

ALTER USER username EXPIRE PASSWORD ROLLOVER PERIOD;

- Phase 1b corresponds to the time remaining after the password rollover period expires
 until the end of PASSWORD_LIFE_TIME. During Phase 1b, the user can log in using only
 the new password.
- **Phase 2:** This phase represents the period of time *after* the password lifetime ends but *before* the user logs in again with the correct password. The correct credentials are needed for Oracle Database to update the account status. Otherwise, the account status will remain unchanged. Oracle Database does not have any background process to update



the account status. All changes to the account status are driven by the Oracle Database server process on behalf of authenticated users.

- Phase 3: When the user finally does log in, the grace period begins. Oracle Database then updates the DBA_USERS.EXPIRY_DATE column to a new value using the current time plus the value of the PASSWORD_GRACE_TIME setting from the account's password profile. At this point, the user receives an ORA-28002 warning message about the password expiring in the near future (for example, ORA-28002 The password will expire within 7 days if PASSWORD_GRACE_TIME is set to 7 days), but the user can still log in without changing the password. The DBA_USERS.EXPIRY_DATE column shows the time in the future when the user will be prompted to change their password.
- Phase 4: After the grace period (Phase 3) ends, the ORA-28001: The password has expired error appears, and the user is prompted to change the password after entering the current, correct password before the authentication can proceed. If the user has an Oracle Active Data Guard configuration, where there is a primary and a stand-by database, and the authentication attempt is made on the standby database (which is a read-only database), then the ORA-28032: Your password has expired and the database is set to read-only error appears. The user should log into the primary database and change the password there.

During any of these four phases, you can query the DBA_USERS data dictionary view to find the user's account status in the DBA_USERS.ACCOUNT_STATUS column.

In the following example, the profile assigned to <code>johndoe</code> includes the specification of a grace period: <code>PASSWORD_GRACE_TIME = 3</code> (the recommended value). The first time <code>johndoe</code> tries to log in to the database after 90 days (this can be <code>any</code> day after the 90th day, that is, the 91st day, 100th day, or another day), he receives a warning message that his password will expire in 3 days. If 3 days pass, and if he does not change his password, then the password expires. After this, he receives a prompt to change his password on any attempt to log in.

```
CREATE PROFILE prof LIMIT

FAILED_LOGIN_ATTEMPTS 4

PASSWORD_LIFE_TIME 90

PASSWORD_GRACE_TIME 3;

ALTER USER johndoe PROFILE prof;
```

A database administrator or a user who has the ALTER USER system privilege can explicitly expire a password by using the CREATE USER and ALTER USER statements. The following statement creates a user with an expired password. This setting forces the user to change the password before the user can log in to the database.

```
CREATE USER jbrown
IDENTIFIED BY password
...
PASSWORD EXPIRE;
```

There is no "password unexpire" clause for the CREATE USER statement, but an account can be "unexpired" by changing the password on the account.

3.2.5.3 Enabling the Gradual Database Password Rollover

To enable the gradual database password rollover, you must configure the PASSWORD ROLLOVER TIME user profile parameter.

• To configure the gradual database password rollover, set the PASSWORD_ROLLOVER_TIME parameter in the CREATE PROFILE or ALTER PROFILE statement.

For example, to set the gradual password rollover time period to 1 day:

```
CREATE PROFILE prof LIMIT
...
PASSWORD ROLLOVER TIME 1;
```

Note the following:

- You specify the rollover time period in days, but you can specify hours if you want. For example, enter 1/24 to specify 1 hour, or 6/24 (or 1/4) to specify 6 hours.
- The minimum value for an active rollover time is 1 hour. The maximum value is 60 days or the lower value of the PASSWORD_LIFE_TIME or PASSWORD_GRACE_TIME parameter. If PASSWORD_GRACE_TIME is set to 0 (zero), then it will be ignored with respect to any limits with PASSWORD_ROLLOVER_TIME. The following table describes these limits:

Table 3-3 Password Rollover Time Limits

Profile Name	PASSWORD_LIFE_TI ME	PASSWORD_GRACE _TIME	PASSWORD_ROLLO VER_TIME
Default	180	7	* Minimum: 1/24 (1 hour)
			* Maximum: 7 (days)
ORA_STIG_PROFILE	60	5	* Minimum: 1/24 (1 hour)
			* Maximum: 5 (days)
User Custom Profile	365	90	* Minimum: 1/24 (1 hour)
			* Maximum: 60 (days)

- The default setting for PASSWORD ROLLOVER TIME is 0 or NULL, which disables it.
- To find database accounts that are currently in the password rollover process, query the ACCOUNT_STATUS column of the DBA_USERS data dictionary view. The status will be IN ROLLOVER.
- The password rollover period begins the moment the administrator changes the password for the database account.

3.2.5.4 Changing a Password to Begin the Gradual Database Password Rollover Period

After you have set a non-zero PASSWORD_ROLLOVER_TIME value, change the user's password and update the password with all the applications.

Use the ALTER USER statement to provision a new rollover password for the application. After the user's new password is provisioned in the database, you can update the password on the application servers. You must complete the password updates before the PASSWORD ROLLOVER TIME period ends.



You can check the user's password rollover status by querying the ACCOUNT_STATUS column of the DBA_USERS data dictionary view. A user account that is within the rollover period will have a status of IN ROLLOVER.

Use the CREATE USER and ALTER USER statements to configure the user, the associated profile, and the password rollover period. CREATE USER allows the administrator to create a new application service account that is associated with a profile with password rollover.
 ALTER USER is more likely where an existing user is associated with a new or modified profile. To alter the profile, use the ALTER PROFILE statement.

The following example CREATE USER creates a new user u1 with password p1 and a profile prof1, with PASSWORD_ROLLOVER_TIME configured. The ALTER USER statement changes the user's password to begin password rollover period. To check the user status, query the DBA USERS data dictionary view.

Create the profile prof1.

```
CREATE PROFILE prof1
LIMIT
PASSWORD ROLLOVER TIME 1;
```

2. Create the user u1 and associate this user with the prof1 profile.

```
CREATE USER u1 IDENTIFIED BY p1 PROFILE prof1;
```

3. Alter the user's password.

```
ALTER USER u1 IDENTIFIED BY p2;
```

4. Query the DBA USER data dictionary view to check the user's rollover status.

```
SELECT USERNAME, ACCOUNT_STATUS FROM DBA_USERS WHERE USERNAME = 'U1';

USERNAME ACCOUNT_STATUS

U1 OPEN & IN ROLLOVER
```

3.2.5.5 Changing a Password During the Gradual Database Password Rollover Period

After the rollover period has begun, you can still change the password.

For example, suppose you inadvertently mistype the password. The following procedure enables you to correct the password even though the rollover process has already begun.

 To change a password after the rollover process has begun, use the ALTER USER statement, with or without the REPLACE clause.

For example, suppose user u1 has the original password p1, p2 is the new password that started the rollover process, and you want to switch to using another password p3 instead of password p2. Any of the following statements work:

```
ALTER USER u1 IDENTIFIED BY p3;

ALTER USER u1 IDENTIFIED BY p3 REPLACE p1;
```

ALTER USER u1 IDENTIFIED BY p3 REPLACE p2;

After you have changed the password to p3, the user can log in using either p1 or p3. An attempt to log in using p2 returns an ORA-1017 Invalid Username/Password error, and is recorded as a failed login attempt. Similarly, after a subsequent password change from p3 to p4 during the rollover period, the user can log in using either p1 or p4. Attempts to log in using either p2 or p3 will return an ORA-1017 Invalid Username/Password error, and are recorded as failed login attempts.

The rollover start time is fixed the first time a user changes their password. The start time is not affected by further password changes during the password rollover period. This design limits the length of time the old password can be used to the PASSWORD_ROLLOVER_TIME period after the password is changed outside of the password rollover period.

3.2.5.6 Ending the Password Rollover Period

There are multiple ways in which you can end the password rollover period.

For example, suppose p1 is the original password for user u1, and p2 is the new password that has been updated to all clients.

- Use one of the following methods to end the password rollover period:
 - Let the password rollover period expire on its own. For example, if the password rollover period is 1 day, wait for 1 day and the password rollover period will expire automatically.
 - As either the user or an administrator, execute the following statement to manually end the password rollover period:

```
ALTER USER u1 EXPIRE PASSWORD ROLLOVER PERIOD;
```

As an administrator, expire the password by executing the ALTER USER username
 PASSWORD EXPIRE statement. The next time the user logs in, he or she will be required
 to change their password.

Beginning with the first connection attempt after the password rollover period expires, Oracle Database drops the earlier password p1. Any attempt to login using the old password p1 returns an ORA-1017 Invalid Username/Password error, and is recorded as a failed login attempt. In effect, connections after the rollover period are authenticated with only the new password, and connections that are attempted with the old password are recorded as failed login attempts. The failed login attempts could lock an account after a sufficient number of consecutive logon attempts with the old password.

Connection attempts to read-only database servers after PASSWORD_ROLLOVER_TIME expires will require new password (p2). The password change to p2 will be made effective for all database clients.

3.2.5.7 Database Behavior During the Gradual Password Rollover Period

Users can perform their standard password changes and logins during the password rollover period.

The following database behavior is implemented during the rollover period:

The user can log in to the database using either the new or the old password. This
effectively increases the lifetime of the old password by the time set with
PASSWORD ROLLOVER TIME.



- Passwords can be changed by using the following methods:
 - An administrator or the user changes his or her own password by using the ALTER USER statement.
 - The user changes his or her own password by using the SQL*Plus password command.
 - The user's password is programmatically changed when the Oracle Call Interface (Oracle OCI) OCIPasswordChange function is executed.
- Oracle Database does not send any special messages to the database clients that indicate
 that the user account is in the password rollover period. This design avoids any errors from
 applications that may not be equipped to handle error and warning messages when a user
 logs in.
- Too many failed login attempts move the user account into a timed lock state, depending
 on the value of profile limit PASSWORD_LOCK_TIME. After the timed lock period expires, the
 state of the password rollover period determines what happens when the user attempts to
 log in.
- User administrators can perform other password lifecyle related actions as usual, such as ACCOUNT LOCK, ACCOUNT UNLOCK, EXPIRE PASSWORD operations.
- The password limits that have been set by the PASSWORD_REUSE_TIME and PASSWORD_REUSE_MAX in the user profile continue to be honored during the rollover period. Any password changes during the rollover period are validated against password change history and added into the password change history.
- Expiring a user account does not affect the password rollover status. As with locked accounts, Oracle Database maintains the password verifiers in their current state. The user can log in using either old or new password (p1 or p2). However, after the user successfully changes their password (to p3), the user is allowed to log in only using the newest password (p3). Both the old passwords are treated as expired.
- Oracle Data Pump exports the password hashes (also known as verifiers) for the latest
 password for user accounts in the password rollover period. For example, if a user u1 has
 an old password p1 and new password p2, then Oracle Data Pump exports password
 hashes for password p2 only.

3.2.5.8 Database Server Behavior After the Password Rollover Period Ends

Oracle Database performs clean-up operations after the gradual database password rollover period ends.

After the password rollover period expires, only the new password is allowed and the old password stops working. Attempting to use the old password returns an ORA-1017 Invalid Username/Password error, and is recorded as a failed login attempt. Connections after the password rollover period will only use the new password, and attempts to use the previous passwords will fail for both read-only and read-write databases. Failed login attempts could lock the user account depending on how many consecutive login attempts have been made to use the old password, based on the FAILED LOGIN ATTEMPTS limit in the password profile.

3.2.5.9 Guideline for Handling Compromised Passwords

If a database account password is suspected of being compromised, then you should change the password immediately.

You can perform this change without going through a password rollover period by using the ALTER USER statement in one execution to both change and expire the old password, instead of

executing two commands sequentially. This option is preferred over changing the PASSWORD_ROLLOVER_TIME in the associated user profile, because other accounts will then be affected.

Use the following syntax to change and expire the old password:

ALTER USER user_name IDENTIFIED BY new_password EXPIRE PASSWORD ROLLOVER PERIOD;

3.2.5.10 How Gradual Database Password Rollover Works During Oracle Data Pump Exports

When a user is exported while they are in the password rollover period, only the verifier corresponding to their new password is exported.

The verifier that corresponds to their old password is not included in the Oracle Data Pump dump file. After the user is imported, only the new password can be used to authenticate.

3.2.5.11 Using Gradual Database Password Rollover in an Oracle Data Guard Environment

In an Oracle Data Guard environment, you must set the <code>ADG_ACCOUNT_INFO_TRACKING</code> environment variable to <code>GLOBAL</code> to use gradual database password rollover.

```
ADG ACCOUNT INFO TRACKING=GLOBAL
```

Otherwise, any initial logons that are performed on the Oracle Data Guard standby by a user who is using the rolled over password after the PASSWORD_ROLLOVER_TIME expiration will result in an ORA-16000: database or pluggable database open for read-only access error.

3.2.5.12 Finding Users Who Still Use Their Old Passwords

You can perform a query that makes use of the AUTHENTICATION_TYPE field for a LOGIN audit record to find users who still use their old passwords.

The unified audit trail can identify which users are still connecting to the database using an old password. The AUTHENTICATION_TYPE field for a LOGON audit record can show if the old verifier was used. This information enables you to find applications that have not been updated with gradual database password rollover to use the new password. The LOGON audit record indicates which application server must be updated.

- 1. Connect to the database as a user who has the AUDIT VIEWER OF AUDIT MGMT role.
- 2. Execute the following query:

```
SELECT DBUSERNAME, AUTHENTICATION_TYPE, OS_USERNAME, USERHOST, EVENT_TIMESTAMP
FROM UNIFIED_AUDIT_TRAIL
WHERE ACTION_NAME='LOGON' AND EVENT_TIMESTAMP > SYSDATE-1
AND REGEXP LIKE(AUTHENTICATION TYPE, '\((VERIFIER=.*?\-OLD\))');
```



If there are users who are still using their old password, then output similar to the following appears:

```
DBUSERNAME
AUTHENTICATION_TYPE

OS_USERNAME USERHOST EVENT_TIMESTAMP
```

```
APP USER (TYPE=(DATABASE)); (CLIENT ADDRESS=((PROTOCOL=tcp)
(HOST=192.0.2.225) (PORT=24938))); (LOGON INFO=((VERIFIER=12C-OLD)
(CLIENT CAPABILITIES=05L NP,07L MR,08L LI))); oracle
db211 14-JAN-21 08.56.34.724172000 PM
APP USER (TYPE=(DATABASE)); (CLIENT ADDRESS=((PROTOCOL=tcp)
(HOST=192.0.2.225) (PORT=24983))); (LOGON INFO=((VERIFIER=12C-OLD)
(CLIENT CAPABILITIES=05L NP,07L MR,08L LI)));
db211 14-JAN-21 09.01.18.938008000 PM
           (TYPE=(DATABASE)); (CLIENT ADDRESS=((PROTOCOL=tcp)
APP USER
(HOST=192.0.2.226) (PORT=48727))); (LOGON INFO=((VERIFIER=12C-OLD)
(CLIENT CAPABILITIES=05L NP,07L MR,08L LI))); oracle
db212 14-JAN-21 10.10.48.042817000 PM
APP USER (TYPE=(DATABASE)); (CLIENT ADDRESS=((PROTOCOL=tcp)
(HOST=192.0.2.226) (PORT=48745))); (LOGON INFO=((VERIFIER=12C-OLD)
(CLIENT CAPABILITIES=05L NP,07L MR,08L LI))); oracle
db212 14-JAN-21 10.12.53.609965000 PM
APP USER (TYPE=(DATABASE)); (CLIENT ADDRESS=((PROTOCOL=tcp)
(HOST=192.0.2.226) (PORT=48751))); (LOGON INFO=((VERIFIER=12C-OLD)
(CLIENT CAPABILITIES=05L NP,07L MR,08L LI))); oracle
db212 14-JAN-21 10.13.41.112194000 PM
```

3.2.6 Managing the Complexity of Passwords

Oracle Database provides a set of functions that you can use to manage the complexity of passwords.

- About Password Complexity Verification
 Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.
- How Oracle Database Checks the Complexity of Passwords
 Oracle Database provides four password verification functions to check password
 complexity.
- Who Can Use the Password Complexity Functions?
 The password complexity functions enable you to customize how users access your data.
- verify_function_11G Function Password Requirements
 The verify_function_11G function originated in Oracle Database Release 11g.
- ora12c_verify_function Password Requirements
 The ora12c_verify_function function fulfills the Department of Defense Database
 Security Technical Implementation Guide requirements.
- ora12c_strong_verify_function Function Password Requirements
 The ora12c strong verify function function is a stringent password verify function.

- ora12c_stig_verify_function Password Requirements
 The ora12c_stig_verify_function function fulfills the Security Technical Implementation
 Guides (STIG) requirements.
- About Customizing Password Complexity Verification
 Oracle Database enables you to customize password complexity for your site.
- Enabling Password Complexity Verification
 The catpvf.sql script can be customized to enable password complexity verification.

3.2.6.1 About Password Complexity Verification

Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.

Using a complexity verification function forces users to create strong, secure passwords for database user accounts. You must ensure that the passwords for your users are complex enough to provide reasonable protection against intruders who try to break into the system by quessing passwords.

3.2.6.2 How Oracle Database Checks the Complexity of Passwords

Oracle Database provides four password verification functions to check password complexity.

These functions are in the <code>catpvf.sql PL/SQL</code> script (located in <code>\$ORACLE_HOME/rdbms/admin</code>). When these functions are enabled, they can check whether users are correctly creating or modifying their passwords. When enabled, password complexity checking is not enforced for user <code>SYS</code>; it only applies to non-<code>SYS</code> users. For better security of passwords, Oracle recommends that you associate the password verification function with the default profile. About Customizing Password Complexity Verification provides an example of how to accomplish this.

3.2.6.3 Who Can Use the Password Complexity Functions?

The password complexity functions enable you to customize how users access your data.

Before you can use the password complexity verification functions in the CREATE PROFILE or ALTER PROFILE statement, you must be granted the EXECUTE privilege on them.

The password verification functions are located in the SYS schema.

3.2.6.4 verify_function_11G Function Password Requirements

The verify_function_11G function originated in Oracle Database Release 11g.



The <code>verify_function_11G</code> function has been deprecated because it enforces the weaker password restrictions from earlier releases of Oracle Database. Instead, you should use the <code>ORA12C_VERIFY_FUNCTION</code>, <code>ORA12C_STRONG_VERIFY_FUNCTION</code>, <code>ORA12C_STIG_VERIFY_FUNCTION</code> functions, which enforce stronger, more up-to-date password verification restrictions.

This function checks for the following requirements when users create or modify passwords:

- The password contains no fewer than 8 characters and includes at least one numeric and one alphabetic character.
- The password is not the same as the user name, nor is it the user name reversed or with the numbers 1–100 appended.
- The password is not the same as the server name or the server name with the numbers 1– 100 appended.
- The password does not contain oracle (for example, oracle with the numbers 1–100 appended).
- The password is not too simple (for example, welcome1, database1, account1, user1234, password1, oracle123, computer1, abcdefg1, or change on install).
- The password differs from the previous password by at least 3 characters.

The following internal check is also applied:

 The password does not contain the double-quotation character ("). However, it can be surrounded by double-quotation marks.

3.2.6.5 ora12c verify function Password Requirements

The orallc_verify_function function fulfills the Department of Defense Database Security Technical Implementation Guide requirements.

This function checks for the following requirements when users create or modify passwords:

- The password contains no fewer than 8 characters and includes at least one numeric and one alphabetic character.
- The password is not the same as the user name or the user name reversed.
- The password is not the same as the database name.
- The password does not contain the word oracle (such as oracle123).
- The password differs from the previous password by at least 3 characters.
- The password contains at least 1 special character.

The following internal check is also applied:

 The password does not contain the double-quotation character ("). However, it can be surrounded by double-quotation marks.

3.2.6.6 ora12c_strong_verify_function Function Password Requirements

The <code>oral2c_strong_verify_function</code> function is a stringent password verify function.

This function checks for the following requirements when users create or modify passwords:

- The password contains no fewer than 9 characters.
- The password contains at least 2 upper case letters.
- The password contains at least 2 lower case letters.
- The password contains at least 2 numeric characters.
- The password contains at least 2 special characters. These special characters are as follows:

```
'~!@#$%^&*()_-+={}[]\/<>,.;?':|(space)
```

The password differs from the previous password by at least 4 characters.

The following internal check is also applied:

• The password does not contain the double-quotation character ("). It can be surrounded by double-quotation marks, however.

3.2.6.7 ora12c stig verify function Password Requirements

The oral2c_stig_verify_function function fulfills the Security Technical Implementation Guides (STIG) requirements.

This function checks for the following requirements when users create or modify passwords:

- The password has at least 15 characters.
- The password has at least 1 lower case character and at least 1 upper case character.
- The password has at least 1 digit.
- The password has at least 1 special character.
- The password differs from the previous password by at least 8 characters.

The following internal check is also applied:

 The password does not contain the double-quotation character ("). However, it can be surrounded by double-quotation marks.

The oral2c_stig_verify_function function is the default handler for the ORA_STIG_PROFILE profile, which is available in a newly-created or upgraded Oracle database.

3.2.6.8 About Customizing Password Complexity Verification

Oracle Database enables you to customize password complexity for your site.

You can create your own password complexity verification function in the SYS schema, similar to the functions that are defined in admin/catpvf.sql. In fact, Oracle recommends that you do so to further secure your site's passwords.

Note the following:

- Do not include Data Definition Language (DDL) statements in the custom password complexity verification function. DDLs are not allowed during the execution of password complexity verification functions.
- Do not modify the admin/catpvf.sql script or the Oracle-supplied password complexity functions. You can create your own functions based on the contents of these files.
- If you make no modifications to the utlpwdmg.sql script, then it uses the orallo verify function function as the default function.



Guideline 1 in Guidelines for Securing Passwords for general advice on creating passwords



3.2.6.9 Enabling Password Complexity Verification

The catpvf.sql script can be customized to enable password complexity verification.

To enable password complexity verification, you must edit the <code>catpvf.sql</code> script to use the password verification function that you want, and then run the script to enable it.

1. Log in to SQL*Plus with administrative privileges.

For example:

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

2. Run the catpvf.sql script (or your modified version of this script) to create the password complexity functions in the SYS schema.

```
@$ORACLE HOME/rdbms/admin/catpvf.sql
```

3. Grant any users who must use this function the EXECUTE privilege on it.

For example:

```
GRANT pmsith EXECUTE ON oral2c_strong_verify_function;
```

- 4. In the default profile or the user profile, set the PASSWORD_VERIFY_FUNCTION setting to either the sample password complexity function in the catpvf.sql script, or to your customized function. Use one of the following methods:
 - Log in to SQL*Plus with administrator privileges and use the CREATE PROFILE or ALTER
 PROFILE statement to enable the function. Ensure that you have the EXECUTE privilege
 on the function.

For example, to update the default profile to use the <code>oral2c_strong_verify_function</code> function:

```
ALTER PROFILE default LIMIT PASSWORD_VERIFY_FUNCTION oral2c_strong_verify_function;
```

 In Oracle Enterprise Manager Cloud Control, from the Administration menu, select Security, and then Profiles. Select the Password tab. Under Complexity, from the Complexity function list, select the name of the complexity function that you want. Click Apply.

After you have enabled password complexity verification, it takes effect immediately. If you must disable it, then run the following statement:

```
ALTER PROFILE DEFAULT LIMIT PASSWORD VERIFY FUNCTION NULL;
```



Note:

The ALTER USER statement has a REPLACE clause. With this clause, users can change their own unexpired passwords by supplying the previous password to authenticate themselves.

If the password has expired, then the user cannot log in to SQL to issue the ALTER USER command. Instead, the OCIPasswordChange() function must be used, which also requires the previous password.

A database administrator with ALTER ANY USER privilege can change any user password (force a new password) without supplying the old one.

3.2.7 Managing Password Case Sensitivity

You can manage the password case sensitivity for passwords from user accounts from previous releases.

- SEC_CASE_SENSITIVE_LOGON Parameter and Password Case Sensitivity
 The SEC_CASE_SENSITIVE_LOGON initialization parameter controls the use of case sensitivity in passwords.
- Using the ALTER SYSTEM Statement to Enable Password Case Sensitivity
 If password case sensitivity has been disabled, then you can enable it by setting the
 SEC CASE SENSITIVE LOGON parameter to TRUE.
- Management of Case Sensitivity for Secure Role Passwords
 For better security, you should ensure that the passwords for secure roles are case sensitive.
- Management of Password Versions of Users
 By default, Oracle Database uses Exclusive Mode, which does not permit case-insensitive passwords, to manage password versions.
- Finding and Resetting User Passwords That Use the 10G Password Version
 For better security, find and reset passwords for user accounts that use the 10G password version so that they use later, more secure password versions.
- How Case Sensitivity Affects Password Files
 By default, password files are case sensitive. The IGNORECASE argument in the ORAPWD command line utility controls the case sensitivity of password files.
- How Case Sensitivity Affects Passwords Used in Database Link Connections
 When you create a database link connection, you must define a user name and password
 for the connection.

3.2.7.1 SEC_CASE_SENSITIVE_LOGON Parameter and Password Case Sensitivity

The SEC_CASE_SENSITIVE_LOGON initialization parameter controls the use of case sensitivity in passwords.

Only users who have the ALTER SYSTEM privilege can set the SEC_CASE_SENSITIVE_LOGON parameter. You should ensure that this parameter is set to TRUE so that case sensitivity is enforced when a user enters a password. However, you should be aware that the SEC_CASE_SENSITIVE_LOGON parameter is deprecated, but is currently retained for backward compatibility.

When you create or modify user accounts, by default, passwords are case sensitive. Case sensitivity affects not only passwords that users enter manually, but it affects password files as well.

Ensure that the SEC_CASE_SENSITIVE_LOGON parameter is not set to FALSE if the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter is set to 12 or 12a. This is because the more secure password versions used for this mode only support case-sensitive password checking. For compatibility reasons, Oracle Database does not prevent the use of FALSE for SEC_CASE_SENSITIVE_LOGON when SQLNET.ALLOWED_LOGON_VERSION_SERVER is set to 12 or 12a. Setting SEC_CASE_SENSITIVE_LOGON to FALSE when SQLNET.ALLOWED_LOGON_VERSION_SERVER is set to 12 or 12a causes all accounts to become inaccessible. If SQLNET.ALLOWED_LOGON_VERSION_SERVER is set to 11 or a lower value, then Oracle recommends that you set SEC_CASE_SENSITIVE_LOGON to TRUE, because the more secure password versions used in Exclusive Mode (when SQLNET.ALLOWED_LOGON_VERSION_SERVER is 12 or 12a) in Oracle Database 12c do not support case insensitive password matching.

In addition to the server-side settings, you should ensure that the client software with which the users are connecting has the $O5L_NP$ capability flag. All Oracle Database release 11.2.0.3 and later clients have the $O5L_NP$ capability. If you have an earlier client, then you must install the CPUOct2012 patch.

3.2.7.2 Using the ALTER SYSTEM Statement to Enable Password Case Sensitivity

If password case sensitivity has been disabled, then you can enable it by setting the SEC CASE SENSITIVE LOGON parameter to TRUE.

- 1. If you are using a password file, then ensure that it was created with the ORAPWD utility IGNORECASE parameter set to N and the FORMAT parameter set to 12.
 - The IGNORECASE parameter overrides the SEC_CASE_SENSITIVE_LOGON parameter. By default, IGNORECASE is set to N, which means that passwords are treated as case sensitive.
 - Note that the <code>ignorecase</code> parameter and the <code>sec_case_sensitive_logon</code> system parameter are deprecated. Oracle strongly recommends that you set <code>ignorecase</code> to <code>n</code> or omit the <code>ignorecase</code> setting entirely.
- Enter the following ALTER SYSTEM statement:

```
ALTER SYSTEM SET SEC CASE SENSITIVE LOGON = TRUE;
```

Related Topics

Oracle Database Administrator's Guide

3.2.7.3 Management of Case Sensitivity for Secure Role Passwords

For better security, you should ensure that the passwords for secure roles are case sensitive.

If before upgrading to Oracle Database 12c release 2 (12.2), you created secure roles by using the IDENTIFIED BY clause of the CREATE ROLE statement, and if upon upgrading to Oracle Database 12c release 12.2, you set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter to one of the Exclusive Modes 12 or 12a, then you must change the password for these secure roles in order for them to remain usable. Because Exclusive Mode is now the default, secure roles that were created in earlier releases (such as Oracle Database 10g, in which the 10g password version was the default) will need to have their passwords changed.



You can query the PASSWORD_REQUIRED and AUTHENTICATION_TYPE columns of the DBA_ROLES data dictionary view to find any secure roles that must have their password changed after upgrade to Oracle Database 12c, in order to become usable again.

Otherwise, the password version for these secure roles cannot be used, unless you set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter to 8. If this parameter is set to 12 or 12a, then you must run the following SQL statement to ensure that case sensitivity is enabled. If not, then secure roles will remain unusable even after their passwords have been changed.

ALTER SYSTEM SET SEC CASE SENSITIVE LOGON = "TRUE";

3.2.7.4 Management of Password Versions of Users

By default, Oracle Database uses Exclusive Mode, which does not permit case-insensitive passwords, to manage password versions.

In a default installation, the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter is set to 12 to enable Exclusive Mode. Exclusive Mode requires that the password-based authentication protocol use one of the case-sensitive password versions (11g or 12c) for the account that is being authenticated. Exclusive Mode excludes the use of the 10g password version that was used in earlier releases. After you upgrade to Oracle Database 12c release 2 (12.2), accounts that use the 10g password version become inaccessible. This occurs because the server runs in Exclusive Mode by default, and Exclusive Mode cannot use the old 10g password version to authenticate the client. The server is left with no password version with which to authenticate the client.

The user accounts from Release 10g use the 10g password version. Therefore, you should find the user accounts that use the 10g password version, and then reset the passwords for these accounts. This generates the appropriate password version based on the setting of the SQLNET.ALLOWED LOGON VERSION SERVER parameter, as follows:

- SQLNET.ALLOWED_LOGON_VERSION_SERVER=8 generates all three password versions 10G, 11G, and 12C.
- SQLNET.ALLOWED_LOGON_VERSION_SERVER=12 generates both 11G and 12C password versions, and removes the 10G password version.
- SQLNET.ALLOWED LOGON VERSION SERVER=12a generates only the 12C password version.

If you first relax the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting to a more permissive value (such as SQLNET.ALLOWED_LOGON_VERSION_SERVER=8) and then import the user accounts from an Oracle Database release 10g (or earlier) release into the current database release, then because the 10g password version (used in the older release) is not case sensitive, these users will still be able to log into the database using any case for their password. But when such a user changes their password, the new 11g and 12c password versions are generated automatically, and their password will automatically become case sensitive, because the default value for the instance initialization parameter SEC_CASE_SENSITIVE_LOGON is TRUE. (Be aware that SEC_CASE_SENSITIVE_LOGON is deprecated, but is currently retained for backward compatibility.)

The following example demonstrates the effect of setting the <code>SEC_CASE_SENSITIVE_LOGON</code> parameter to <code>TRUE</code>. In this scenario, user <code>rtaylor</code> has been imported from Oracle Database release <code>10g</code>, and therefore this account only has the <code>10g</code> password version. On the server, the <code>SQLNET.ALLOWED_LOGON_VERSION_SERVER</code> is set to 8 because otherwise <code>rtaylor</code> would not be able to log in. In addition, the <code>SEC_CASE_SENSITIVE_LOGON</code> parameter is set to <code>TRUE</code> to enable case sensitivity for the <code>11g</code> and <code>12c</code> password versions.

1. Check the password versions for user rtaylor:



2. Connect as user rtaylor.

```
CONNECT rtaylor
Enter password: "MaresEatOats"
Connected.
```

User rtaylor can connect to the database because his password still uses the 10g password version, which is case insensitive. Here, he enters his password in mixed case, though his actual password is all lower case: mareseatoats.

3. Check the password versions for one of the default users, SCOTT.

4. Try connecting as user SCOTT using a mixed case for the password, even though his actual password is all lowercase: luv2walkmyk9.

```
CONNECT SCOTT
Enter password: "LuvToWalkMyK9"

ERROR: ORA-01017: invalid username/password; logon denied Warning: You are no longer connected to ORACLE.
```

Because user SCOTT's password versions are 11G and 12G, the password is case sensitive. The password entered in this example is correct, but the case is incorrect.

5. Alter rtaylor's password to grumble mumble2work.

```
ALTER USER rtaylor IDENTIFIED BY grumble_mumble2work;
User altered.
```

6. Connect with the SYSDBA administrative privilege.

```
CONNECT / AS SYSDBA
```

7. Find the password versions for user rtaylor.

The authentication protocol that was configured with the

SQLNET.ALLOWED_LOGON_VERSION_SERVER and SEC_CASE_SENSITIVE_LOGON settings will enforce the case sensitivity of rtaylor's password, now that he has changed this password.

Try connecting as rtaylor using a mixed case for the password.

```
CONNECT rtaylor
Enter password: "Grumble Mumble2Work"
```

ERROR: ORA-01017: invalid username/password; logon denied Warning: You are no longer connected to ORACLE.

The password entered fails because it was not entered using the case in which the password was created.

9. Try connecting as rtaylor again but with the password using the correct case

```
CONNECT rtaylor
Enter password: "grumble_mumble2work"
Connected.
```

User rtaylor can connect.

The case sensitivity of the <code>rtaylor</code> account is a result of the server's default setting for <code>SEC_CASE_SENSITIVE_LOGON</code>, which is <code>TRUE</code>. If this setting is <code>FALSE</code>, then case-insensitive matching can be restored because the <code>rtaylor</code> account still has the <code>10G</code> password version. However, Oracle does not recommend this setting. The <code>SEC_CASE_SENSITIVE_LOGON</code> parameter is deprecated for this reason. For greater security, Oracle strongly recommends that you keep case-sensitive password authentication enabled.

3.2.7.5 Finding and Resetting User Passwords That Use the 10G Password Version

For better security, find and reset passwords for user accounts that use the 10g password version so that they use later, more secure password versions.

Finding All Password Versions of Current Users

You can query the DBA_USERS data dictionary view to find a list of all the password versions configured for user accounts.

For example:

SELECT USERNAME, PASSWORD VERSIONS FROM DBA USERS;

USERNAME	PASSWORD_VERSIONS
TONES	10G 11G 12C
JONES	
ADAMS	10G 11G
CLARK	10G 11G
PRESTON	11G
BLAKE	10G

The PASSWORD_VERSIONS column shows the list of password versions that exist for the account. 10G refers to the earlier case-insensitive Oracle password version, 11G refers to the SHA-1-based password version, and 12C refers to the SHA-2-based SHA-512 password version.

- User jones: The password for this user was reset in Oracle Database 12c Release 12.1 when the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter setting was 8. This enabled all three password versions to be created.
- Users adams and clark: The passwords for these accounts were originally created in Oracle Database 10g and then reset in Oracle Database 11g. The Oracle Database 11g software was using the default SQLNET.ALLOWED_LOGON_VERSION setting of 8 at that time. Because case insensitivity is enabled by default, their passwords are now case sensitive, as is the password for preston.

- User preston: This account was imported from an Oracle Database 11g database that was running in Exclusive Mode (SQLNET.ALLOWED LOGON VERSION = 12).
- User blake: This account still uses the Oracle Database 10g password version. At this stage, user blake is prevented from logging in.

Resetting User Passwords That Use the 10G Password Version

For better security, remove the 10g password version from the accounts of all users. In the following procedure, to reset the passwords of users who have the 10g password version, you must temporarily relax the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting, which controls the ability level required of clients before login can be allowed. Relaxing the setting enables these users to log in and change their passwords, and hence generate the newer password versions in addition to the 10g password version. Afterward, you can set the database to use Exclusive Mode and ensure that the clients have the $O5L_NP$ capability. Then the users can reset their passwords again, so that their password versions no longer include 10g, but only have the more secure 11g and 12g password versions.

1. Query the DBA USERS view to find users who only use the 10g password version.

```
SELECT USERNAME FROM DBA_USERS
WHERE ( PASSWORD_VERSIONS = '10G '
OR PASSWORD_VERSIONS = '10G HTTP ')
AND USERNAME <> 'ANONYMOUS';
```

- 2. Configure the database so that it does not run in Exclusive Mode, as follows:
 - a. Edit the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting in the sqlnet.ora file so that it is more permissive than the default. For example:

```
SQLNET.ALLOWED LOGON VERSION SERVER=11
```

- b. Restart the database.
- 3. Expire the users that you found when you queried the DBA_USERS view to find users who only use the 10g password version.

You must expire the users who have only the 10G password version, and do not have one or both of the 11G or 12C password versions.

For example:

```
ALTER USER username PASSWORD EXPIRE;
```

4. Ask the users whose passwords you expired to log in.

When the users log in, they are prompted to change their passwords. The database generates the missing 11G and 12C password versions for their account, in addition to the 10G password version. The 10G password version continues to be present, because the database is running in the permissive mode.

- 5. Ensure that the client software with which the users are connecting has the O5L NP ability.
 - All Oracle Database release 11.2.0.3 and later clients have the O5L_NP ability. If you have an earlier Oracle Database client, then you must install the CPUOct2012 patch.
- 6. After all clients have the O5L_NP capability, set the security for the server back to Exclusive Mode, as follows:
 - a. Remove the SEC_CASE_SENSITIVE_LOGON parameter setting from the instance initialization file, or set SEC CASE SENSITIVE LOGON to TRUE.

```
SEC_CASE_SENSITIVE_LOGON = TRUE
```



b. Remove the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter from the server sqlnet.ora file, or set the value of SQLNET.ALLOWED_LOGON_VERSION_SERVER in the server sqlnet.ora file back to 12, to set it to an Exclusive Mode.

```
SQLNET.ALLOWED_LOGON_VERSION_SERVER = 12
```

- Restart the database.
- 7. Find the accounts that still have the 10g password version.

```
SELECT USERNAME FROM DBA_USERS
WHERE PASSWORD_VERSIONS LIKE '%10G%'
AND USERNAME <> 'ANONYMOUS';
```

8. Expire the accounts that still have the 10g password version.

```
ALTER USER username PASSWORD EXPIRE;
```

9. Ask these users to log in to their accounts.

When the users log in, they are prompted to reset their passwords. The database then generates only the 11G and 12C password versions for their accounts. Because the database is running in Exclusive Mode, the 10G password version is no longer generated.

10. Rerun the following query:

```
SELECT USERNAME FROM DBA_USERS
WHERE PASSWORD_VERSIONS LIKE '%10G%'
AND USERNAME <> 'ANONYMOUS';
```

If this query does not return any results, then it means that no user accounts have the 10G password version. Hence, the database is running in a more secure mode than in previous releases.

3.2.7.6 How Case Sensitivity Affects Password Files

By default, password files are case sensitive. The IGNORECASE argument in the ORAPWD command line utility controls the case sensitivity of password files.

The default value for IGNORECASE is N (no), which enforces case sensitivity. For better security, set IGNORECASE to N or omit the ignorecase argument entirely. Note that IGNORECASE is deprecated.

The following example shows how to enable case sensitivity in password files.

```
orapwd file=orapw entries=100
Enter password for SYS: password
```

This command creates a case sensitive password file called <code>orapw</code>. By default, passwords are case sensitive. Afterward, if you connect using this password, it succeeds—as long as you enter it using the *exact* case in which it was created. If you enter the same password but with a different case, then the authentication attempt that uses the password fails.

Alternatively, you can turn the IGNORECASE parameter off by using password file migration from one format to another format. For example:

```
orapwd input file=input password file file=output password file
```

If you imported user accounts from a previous release and these accounts were created with SYSDBA or SYSOPER administrative privilege, then they will be included in the password file. The passwords for these accounts are case insensitive. The next time these users change their passwords, and assuming case sensitivity is enabled, the passwords become case sensitive. For greater security, have these users change their passwords.

Related Topics

Oracle Database Administrator's Guide

3.2.7.7 How Case Sensitivity Affects Passwords Used in Database Link Connections

When you create a database link connection, you must define a user name and password for the connection.

When you create the database link connection, the password is case sensitive. How a user enters his or her password for connections depends on the release in which the database link was created:

- Users can connect from a pre-Oracle Database 12c database to a Oracle Database 12c database. Because case sensitivity is enabled, then the user must enter the password using the case that was used when the account was created.
- If the user connects from a Oracle Database 12c database to a pre-Oracle Database 12c database, and if the SEC_CASE_SENSITIVE_LOGON parameter in the pre-Release 12c database had been set to FALSE, then the password for this database link can be specified using any case.

You can find the user accounts for existing database links by querying the V\$DBLINK view. For example:

```
SELECT DB_LINK, OWNER_ID FROM V$DBLINK;
```

See Oracle Database Reference for more information about the V\$DBLINK view.

3.2.8 Ensuring Against Password Security Threats by Using the 12C Password Version

The 12C password version enables users to create complex passwords that meet compliance standards.

- About the 12C Version of the Password Hash
 The 12C password hash protects against password-based security threats by including support for mixed case passwords.
- Oracle Database 12C Password Version Configuration Guidelines
 By default, Oracle Database generates two versions of the password hash: 11G and 12C.
- Configuring Oracle Database to Use the 12C Password Version Exclusively
 You should set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter to 12a so that only
 the 12C password hash version is used.
- How Server and Client Logon Versions Affect Database Links
 The SQLNET.ALLOWED_LOGON_VERSION_SERVER and
 SQLNET.ALLOWED_LOGON_VERSION_CLIENT parameters can accommodate connections
 between databases and clients of different releases.
- Configuring Oracle Database Clients to Use the 12C Password Version Exclusively
 An intruder may try to provision a fake server to downgrade authentication and trick the
 client into using a weaker password hash version.

3.2.8.1 About the 12C Version of the Password Hash

The 12C password hash protects against password-based security threats by including support for mixed case passwords.



The cryptographic hash function used for generating the 12c version of the password hash is based on a de-optimized algorithm involving Password-Based Key Derivation Function 2 (PBKDF2) and the SHA-512 cryptographic hash functions. The PBKDF2 algorithm introduces computational asymmetry in the challenge that faces an intruder who is trying to recover the original password when in possession of the 12c version of the password hash. The 12c password generation performs a SHA-512 hash of the PBKDF2 output as its last step. This two-step approach used in the 12c password version generation allows server CPU resources to be conserved when the client has the O7L_MR capability. This is because during the password verification phase of the O5LOGON authentication, the server only needs to perform a single SHA-512 hash of a value transmitted by the O7L_MR capable client, rather than having to repeat the entire PBKDF2 calculation on the password itself.

In addition, the 12C password version adds a salt to the password when it is hashed, which provides additional protection. The 12C password version enables your users to create far more complex passwords. The 12C password version's use of salt, its use of PBKDF2 deoptimization, and its support for mixed-case passwords makes it more expensive for an intruder to perform dictionary or brute force attacks on the 12C password version in an attempt to recover the user's password. Oracle recommends that you use the 12C version of the password hash.

The password hash values are considered to be extremely sensitive, because they are used as a "shared secret" between the server and person who is logging in. If an intruder learns this secret, then the protection of the authentication is immediately and severely compromised. Remember that administrative users who have account management privileges, administrative users who have the SYSDBA administrative privilege, or even users who have the EXP_FULL_DATABASE role can immediately access the password hash values. Therefore, this type of administrative user must be trustworthy if the integrity of the database password-based authentication is to be preserved. If you cannot trust these administrators, then it is better to deploy a directory server (such as Oracle Database Enterprise User Security) so that the password hash values remain within the Enterprise User Security directory and are never accessible to anyone except the Enterprise User Security administrator.

Related Topics

Oracle Database Net Services Reference

3.2.8.2 Oracle Database 12C Password Version Configuration Guidelines

By default, Oracle Database generates two versions of the password hash: 11G and 12C.

The version of the password hash that Oracle Database uses to authenticate a given client depends on the client's ability, and the settings for the <code>SQLNET.ALLOWED_LOGON_VERSION_CLIENT</code> and <code>SQLNET.ALLOWED_LOGON_VERSION_SERVER</code> parameters. See the column "Ability Required of the Client" in the "SQLNET.ALLOWED_LOGON_VERSION_SERVER Settings" table in the <code>SQLNET.ALLOWED_LOGON_VERSION_SERVER</code> parameter description in *Oracle Database Net Services Reference* for detailed information about how the client authentication works with password versions.

The 10G password version, which was generated in Oracle Database 10g, is not case sensitive. Both the 11G and 12C password versions are case sensitive.

In Oracle Database 12g release 2 (12.2), the sqlnet.ora parameter SQLNET.ALLOWED_LOGON_VERSION_SERVER defaults to 12, which is Exclusive Mode and prevents the use of the 10g password version, and the SQLNET.ALLOWED_LOGON_VERSION_CLIENT parameter defaults to 11. For new accounts, when the client is Oracle Database 12c, then Oracle Database uses the 12c password version exclusively with clients that are running the Oracle Database 12c release software. For accounts that were created before Oracle



Database release 12c, logins will succeed as long as the client has the O5L_NP ability, because an <code>11G</code> password version normally exists for accounts created in earlier releases such as Oracle Database release 11g. For a very old account (for example, from Oracle Database release 10g), the user's password may need to be reset, in order to create a SHA-1 password version for the account. To configure this server to generate only the <code>12c</code> password version whenever a new account is created or an existing account password is changed, then set the <code>SQLNET.ALLOWED_LOGON_VERSION_SERVER</code> parameter to <code>12a</code>. However, if you want your applications to be compatible with older clients, then ensure that <code>SQLNET.ALLOWED_LOGON_VERSION_SERVER</code> is set to <code>12</code>, which is the default.

How you set the <code>SQLNET.ALLOWED_LOGON_VERSION_SERVER</code> parameter depends on the balance of security and interoperability with older clients that your system requires. You can control the levels of security as follows:

- Greatest level of compatibility: To configure the server to generate all three versions of the password hash (the 12C password version, the 11G password version, and the DES-based 10G password version), whenever a new account is created or an existing account password is changed, set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter to the value 11 or lower. (Be aware that earlier releases used the value 8 as the default.)
- Recommended level of security: To configure the server to generate both the 12C password version and the 11G password version (but not the 10G password version), whenever a new account is created or an existing account password is changed, set the SQLNET.ALLOWED LOGON VERSION SERVER parameter to the value 12.
- **Highest level of security:** To configure the server to generate *only* the 12C password version whenever a new account is created or an existing account password is changed, set the SQLNET.ALLOWED LOGON VERSION SERVER parameter to the value 12a.

During authentication, the following scenarios are possible, based on the kinds of password versions that exist for the account, and on the version of the client software being used:

• Accounts with only the 10G version of the password hash: If you want to force the server to generate the newer versions of the password hash for older accounts, an administrator must expire the password for any account that has only the 10G password version (and none of the more secure password versions, 11G or 12C). You must generate these password versions because the database depends on using these password versions to provide stronger security. You can find these users as follows.

```
SELECT USERNAME FROM DBA_USERS
WHERE PASSWORD_VERSIONS LIKE '%10G%
AND USERNAME <> 'ANONYMOUS';
```

And then expire each account as follows:

```
ALTER USER username PASSWORD EXPIRE;
```

After you have expired each account, notify these users to log in, in which case they will be prompted to change their password. The version of the client determines the password version that is used. The setting of the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter determines the password versions that are generated. If the client has the O7L_MR ability (Oracle Database release 12c), then the 12C password version is used to authenticate. If the client has the O5L_NP ability but not the O7L_MR ability (such as Oracle Database release 11g clients), then the 11g password version is used to authenticate. You should upgrade all clients to Oracle Database release 12c so that the 12c password version can be used exclusively to authenticate. (By default, Oracle Database release 11.2.0.3 and later clients have the O5L_NP ability, which enables the 11g password version to be used exclusively. If you have an earlier Oracle Database client, then you must install the CPUOct2012 patch.)



When an account password is expired and the ALLOWED_LOGON_VERSION_SERVER parameter is set to 12 or 12a, then the 10G password version is removed and only one or both of the new password versions are created, depending on how the parameter is set, as follows:

- If ALLOWED_LOGON_VERSION_SERVER is set to 12 (the default), then both the 11g and 12c versions of the password hash are generated.
- If ALLOWED_LOGON_VERSION_SERVER is set to 12a, then only the 12C version of the password hash is generated.

For more details, see the "Generated Password Version" column in the table in the "Usage Notes" section for the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter in *Oracle Database Net Services Reference*.

- Accounts with both 10G and 11G versions of the password hash: For users who are
 using a Release 10g or later client, the user logins will succeed because the 11G version of
 the password hash is used. However, to use the latest version, expire these passwords, as
 described in the previous bulleted item for accounts.
- Accounts with only the 11G version of the password hash: The authentication uses
 the 11G version of the password hash. To use the latest version, expire the passwords, as
 described in the first bulleted item.

The Oracle Database 12c default configuration for SQLNET.ALLOWED_LOGON_VERSION_SERVER is 12, which means that it is compatible with Oracle Database 12c release 2 (12.2) authentication protocols and later products that use OCI-based drivers, including SQL*Plus, ODBC, Oracle .NET, Oracle Forms, and various third-party Oracle Database adapters. It is also compatible with JDBC type-4 (thin) versions that have had the CPUOct2012 bundle patch applied or starting with Oracle Database 11g, and Oracle Database Client interface (OCI)-based drivers starting in Oracle Database 10g release 10.2. Be aware that earlier releases of the OCI client drivers cannot authenticate to an Oracle database using password-based authentication.

3.2.8.3 Configuring Oracle Database to Use the 12C Password Version Exclusively

You should set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter to 12a so that only the 12C password hash version is used.

The 12C password version is the most restrictive and secure of the password hash versions, and for this reason, Oracle recommends that you use only this password version. By default, SQLNET.ALLOWED_LOGON_VERSION_SERVER is set to 12, which enables both the 11G and 12C password versions to be used. (Both the SQLNET.ALLOWED_LOGON_VERSION_SERVER values 12 and 12a are considered Exclusive Mode, which prevents the use of the earlier 10G password version.) If you have upgraded from a previous release, or if

SQLNET.ALLOWED_LOGON_VERSION_SERVER is set to 12 or another setting that was used in previous releases, then you should reconfigure this parameter, because intruders will attempt to downgrade the authentication to use weaker password versions. The table later in this topic shows the effect of the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting on password version generation.

Be aware that you can use the 12C password version exclusively only if you use Oracle Database 12c release 12.1.0.2 or later clients. Before you change the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter to 12a, check the versions of the database clients that are connected to the server.

- 1. Log in to SQL*Plus as an administrative user who has the ALTER USER system privilege.
- Perform the following SQL query to find the password versions of your users.

SELECT USERNAME, PASSWORD VERSIONS FROM DBA USERS;



3. Expire the account of each user who does not have the 12c password version.

For example, assuming user blake is still using a 10G password version:

ALTER USER blake PASSWORD EXPIRE;

The next time that these users log in, they will be forced to change their passwords, which enables the server to generate the password versions required for Exclusive Mode.

4. Remind users to log in within a reasonable period of time (such as 30 days).

When they log in, they will be prompted to change their password, ensuring that the password versions required for authentication in Exclusive Mode are generated by the server. (For more information about how Exclusive Mode works, see the usage notes for the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter in *Oracle Database Net Services Reference*.)

- 5. Manually change the passwords for accounts that are used in test scripts or batch jobs so that they exactly match the passwords used by these test scripts or batch jobs, including the password's case.
- **6.** Enable the Exclusive Mode configuration as follows:
 - a. Create a back up copy of the sqlnet.ora parameter file.

By default, this file is located in the <code>\$ORACLE_HOME/network/admin</code> directory on UNIX operating systems and the <code>%ORACLE_HOME%\network\admin</code> directory on Microsoft Windows operating systems.

Be aware that in a Multitenant environment, the settings in the sqlnet.ora file apply to all PDBs.

- b. Set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter, using Table 3-4 for guidance.
- c. Save the sqlnet.ora file.

The following table shows the effect of the SQLNET.ALLOWED_LOGON_VERSION_SERVER setting on password version generation.

Table 3-4 Effect of SQLNET.ALLOWED_LOGON_VERSION_SERVER on Password Version Generation

SQLNET.ALLOWED_LOGON_VERSION_SE RVER Setting	8	11	12	12a
Server runs in Exclusive Mode?	No	No	Yes	Yes
Generate the 10G password version?	Yes	Yes	No	No
Generate the 11G password version?	Yes	Yes	Yes	No
Generate the 12C password version?	Yes	Yes	Yes	Yes

If you only use Oracle Database 12c release 12.1.0.2 or later clients, then set SQLNET.ALLOWED_LOGON_VERSION_SERVER to 12a.

The higher the setting, the more restrictive the use of password versions, as follows:

- A setting of 12a, the most restrictive and secure setting, only permits the 12c password version
- A setting of 12 permits both the 11G and 12C password versions to be used for authentication.

A setting of 8 permits the most password versions: 10G, 11G, and 12C.

For detailed information about the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter, see *Oracle Database Net Services Reference*.

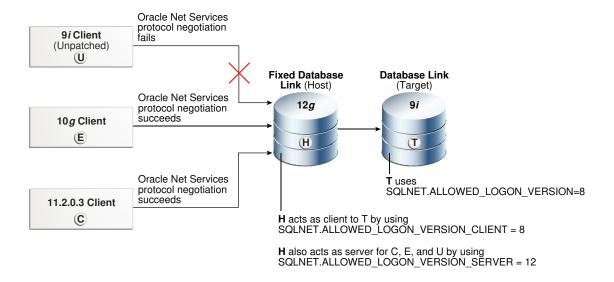


If your system hosts a fixed database link to a target database that runs an earlier release, then you can set the <code>SQLNET.ALLOWED_LOGON_VERSION_CLIENT</code> parameter, as described in How Server and Client Logon Versions Affect Database Links.

3.2.8.4 How Server and Client Logon Versions Affect Database Links

The SQLNET.ALLOWED_LOGON_VERSION_SERVER and SQLNET.ALLOWED_LOGON_VERSION_CLIENT parameters can accommodate connections between databases and clients of different releases.

The following diagram illustrates how connections between databases and clients of different releases work. The SQLNET.ALLOWED_LOGON_VERSION_CLIENT parameter affects the "client allowed logon version" aspect of a server that hosts the database link **H**. This setting enables **H** to connect through database links to older servers, such as those running Oracle 9*i* (**T**), yet still refuse connections from older unpatched clients (**U**). When this happens, the Oracle Net Services protocol negotiation fails, which raises an ORA-28040: No matching authentication protocol error message in this client, which is attempting to authenticate using the Oracle 9*l* software. The Oracle Net Services protocol negotiation for Oracle Database 10*g* release 10.2 client **E** succeeds because this release incorporates the critical patch update CPUOct2012. The Oracle Net Services protocol negotiation for Release 11.2.0.3 client **C** succeeds because it uses a secure password version.



This scenario uses the following settings for the system that hosts the database link H:

SQLNET.ALLOWED_LOGON_VERSION_CLIENT=8
SQLNET.ALLOWED LOGON VERSION SERVER=12

Note that the remote Oracle Database **T** has the following setting:



SQLNET.ALLOWED LOGON VERSION=8

If the release of the remote Oracle Database T does not meet or exceed the value defined by the SQLNET.ALLOWED_LOGON_VERSION_CLIENT parameter set for the host H, then queries over the fixed database link would fail during authentication of the database link user, resulting in an ORA-28040: No matching authentication protocol error when an end-user attempts to access a table over the database link.

Note:

If you are using an older Oracle Database client (such as Oracle Database 11g release 11.1.0.7), then Oracle strongly recommends that you upgrade to use the critical patch update CPUOct2012.

See Also:

- Oracle Database Net Services Reference for more information about the SQLNET.ALLOWED LOGON VERSION CLIENT parameter
- http://www.oracle.com/technetwork/topics/security/ cpuoct2012-1515893.html for more information about CPUOct2012

3.2.8.5 Configuring Oracle Database Clients to Use the 12C Password Version Exclusively

An intruder may try to provision a fake server to downgrade authentication and trick the client into using a weaker password hash version.

- To prevent the use of the 10g password version, or both the 10g and 11g password versions, after you configure the server, configure the clients to run in Exclusive Mode, as follows:
 - To use the client Exclusive Mode setting to permit both the 11G and 12C password versions:

```
SQLNET.ALLOWED_LOGON_VERSION_CLIENT = 12
```

To use the more restrictive client Exclusive Mode setting to permit only the 12c password version (this setting permits the client to connect only to Oracle Database 12c release 1 (12.1.0.2) and later servers):

```
SQLNET.ALLOWED_LOGON_VERSION_CLIENT = 12a
```

If the server and the client are both installed on the same computer, then ensure that the <code>TNS_ADMIN</code> environment variable for each points to the correct directory for its respective Oracle Net Services configuration files. If the variable is the same for both, then the server could use the client's <code>SQLNET.ALLOWED LOGON VERSION CLIENT</code> setting instead.

If you are using older Oracle Database clients (such as Oracle Database 11g release 11.1.0.7), then you should apply CPU Oct2012 or later to these clients. This patch provides the <code>05L_NP</code> ability. Unless you apply this patch, users will be unable to log in.

See Also:

- Oracle Database Net Services Reference for more information about the SQLNET.ALLOWED_LOGON_VERSION_CLIENT parameter
- The following Oracle Technology Network site for more information about CPUOct2012:

http://www.oracle.com/technetwork/topics/security/cpuoct2012-1515893.html

3.2.9 Managing the Secure External Password Store for Password Credentials

The secure external password store is a client-side wallet that is used to store password credentials.

- About the Secure External Password Store
 You can store password credentials database connections by using a client-side Oracle
 wallet.
- How Does the Secure External Password Store Work?
 Users (and applications, batch jobs, and scripts) connect to databases by using a standard CONNECT statement that specifies a database connection string.
- About Configuring Clients to Use the Secure External Password Store
 If your client is configured to use external authentication, such as Windows native
 authentication or SSL, then Oracle Database uses that authentication method.
- Configuring a Client to Use the Secure External Password Store
 You can configure a client to use the secure external password store feature by using the
 mkstore command-line utility.
- Example: Sample sqlnet.ora File with Wallet Parameters Set
 You can set special parameters in the sqlnet.ora file to control how wallets are
 managed.
- Managing External Password Store Credentials
 The mkstore command-line utility manages credentials from an external password store.

3.2.9.1 About the Secure External Password Store

You can store password credentials database connections by using a client-side Oracle wallet.

An Oracle wallet is a secure software container that stores authentication and signing credentials. This wallet usage can simplify large-scale deployments that rely on password credentials for connecting to databases. When this feature is configured, application code, scripts no longer need embedded user names and passwords. This reduces risk because the passwords are no longer exposed, and password management policies are more easily enforced without changing application code whenever user names or passwords change.





The external password store of the wallet is separate from the area where public key infrastructure (PKI) credentials are stored. Consequently, you cannot use Oracle Wallet Manager to manage credentials in the external password store of the wallet. Instead, use the command-line utility mkstore to manage these credentials.

Related Topics

- Using Proxy Authentication with the Secure External Password Store
 Use a secure external password store if you are concerned about the password used in
 proxy authentication being obtained by a malicious user.
- Oracle Database Enterprise User Security Administrator's Guide

3.2.9.2 How Does the Secure External Password Store Work?

Users (and applications, batch jobs, and scripts) connect to databases by using a standard CONNECT statement that specifies a database connection string.

This string can include a user name and password, and an Oracle Net service name identifying the database on an Oracle Database network. If the password is omitted, the connection prompts the user for the password.

For example, the service name could be the URL that identifies that database, or a TNS alias you entered in the tnsnames.ora file in the database. Another possibility is a host:port:sid string.

The following examples are standard CONNECT statements that could be used for a client that is not configured to use the external password store:

```
CONNECT salesapp@sales_db.us.example.com
Enter password: password

CONNECT salesapp@orasales
Enter password: password

CONNECT salesapp@ourhost37:1527:DB17
Enter password: password
```

In these examples, salesapp is the user name, with the unique connection string for the database shown as specified in three different ways. You could use its URL sales_db.us.example.com, or its TNS alias orasales from the tnsnames.ora file, or its host:port:sid String.

However, when clients are configured to use the secure external password store, applications can connect to a database with the following CONNECT statement syntax, without specifying database login credentials:

```
CONNECT /@db_connect_string AS SYSDBA

CONNECT /@db connect string AS SYSOPER
```

In this specification, <code>db_connect_string</code> is a valid connection string to access the intended database, such as the service name, URL, or alias as shown in the earlier examples. Each

user account must have its own unique connection string; you cannot create one connection string for multiple users.

In this case, the database credentials, user name and password, are securely stored in an Oracle wallet created for this purpose. The autologin feature of this wallet is turned on, so the system does not need a password to open the wallet. From the wallet, it gets the credentials to access the database for the user they represent.

Related Topics

Oracle Database Enterprise User Security Administrator's Guide

3.2.9.3 About Configuring Clients to Use the Secure External Password Store

If your client is configured to use external authentication, such as Windows native authentication or SSL, then Oracle Database uses that authentication method.

The same credentials used for this type of authentication are typically also used to log in to the database. For clients not using such authentication methods or wanting to override them for database authentication, you can set the <code>SQLNET.WALLET_OVERRIDE</code> parameter in <code>sqlnet.ora</code> to <code>TRUE</code>. The default value for <code>SQLNET.WALLET_OVERRIDE</code> is <code>FALSE</code>, allowing standard use of authentication credentials as before.

3.2.9.4 Configuring a Client to Use the Secure External Password Store

You can configure a client to use the secure external password store feature by using the mkstore command-line utility.

1. Create a wallet on the client by using the following syntax at the command line:

```
mkstore -wrl wallet location -create
```

For example:

```
mkstore -wrl c:\oracle\product\19.1.0\db_1\wallets -create
Enter password: password
```

wallet_location is the path to the directory where you want to create and store the wallet. This command creates an Oracle wallet with the autologin feature enabled at the location you specify. The autologin feature enables the client to access the wallet contents without supplying a password.

The mkstore utility -create option uses password complexity verification. See About Password Complexity Verification for more information.

2. Create database connection credentials in the wallet by using the following syntax at the command line:

```
\label{location-createCredential} $$ mkstore -wrl wallet_location - createCredential $$ db\_connect\_string username $$ Enter password: $password$$
```

For example:

```
\label{lem:mkstore -wrl c:\oracle\product\19.1.0\db_1\wallets -createCredential orcl system Enter password: password
```

In this specification:

- wallet location is the path to the directory where you created the wallet in Step 1.
- db_connect_string is the TNS alias you use to specify the database in the tnsnames.ora file or any service name you use to identify the database on an Oracle

network. By default, tnsnames.ora is located in the <code>\$ORACLE_HOME/network/admin</code> directory on UNIX systems and in <code>ORACLE_HOME/network/admin</code> on Windows.

username is the database login credential. When prompted, enter the password for this
user.

Repeat this step for each database you want accessible using the CONNECT / @db_connect_string syntax. The db_connect_string used in the CONNECT / @db_connect_string statement must be identical to the db_connect_string specified in the -createCredential command.

3. In the client sqlnet.ora file, enter the WALLET_LOCATION parameter and set it to the directory location of the wallet you created in Step 1.

For example, if you created the wallet in <code>\$ORACLE_HOME/network/admin</code> and your Oracle home is set to <code>/private/orall</code>, then you need to enter the following into your client <code>sglnet.ora</code> file:

```
WALLET_LOCATION =
  (SOURCE =
      (METHOD = FILE)
      (METHOD_DATA =
      (DIRECTORY = /private/orall/network/admin)
  )
)
```

4. In the client sqlnet.ora file, enter the SQLNET.WALLET_OVERRIDE parameter and set it to TRUE as follows:

```
SQLNET.WALLET OVERRIDE = TRUE
```

This setting causes all CONNECT $/@db_connect_string$ statements to use the information in the wallet at the specified location to authenticate to databases.

When external authentication is in use, an authenticated user with such a wallet can use the CONNECT /@db_connect_string syntax to access the previously specified databases without providing a user name and password. However, if a user fails that external authentication, then these connect statements also fail.

Note:

If an application uses SSL for encryption, then the sqlnet.ora parameter, SQLNET.AUTHENTICATION_SERVICES, specifies SSL and an SSL wallet is created. If this application wants to use secret store credentials to authenticate to databases (instead of the SSL certificate), then those credentials must be stored in the SSL wallet. After SSL authentication, if SQLNET.WALLET_OVERRIDE = TRUE, then the user names and passwords from the wallet are used to authenticate to databases. If SQLNET.WALLET_OVERRIDE = FALSE, then the SSL certificate is used.

3.2.9.5 Example: Sample sqlnet.ora File with Wallet Parameters Set

You can set special parameters in the sqlnet.ora file to control how wallets are managed.

Example 3-2 shows a sample sqlnet.ora file with the WALLET_LOCATION and the SQLNET.WALLET_OVERRIDE parameters set as described in Steps 3 and 4 of Configuring a Client to Use the Secure External Password Store.

Example 3-2 Sample sqlnet.ora File with Wallet Parameters Set

3.2.9.6 Managing External Password Store Credentials

The mkstore command-line utility manages credentials from an external password store.

Listing External Password Store Contents

You can view the contents, including specific credentials, of a client wallet external password store.

- Adding Credentials to an External Password Store You can store multiple credentials in one client wallet.
- Tod but store maniple diedentials in one cheft wanet
- Modifying Credentials in an External Password Store
 You can modify the database login credentials that are stored in the wallet if the database connection strings change.
- Deleting Credentials from an External Password Store
 You can delete login credentials for a database from a wallet if the database no longer exists or to disable connections to a specific database.

3.2.9.6.1 Listing External Password Store Contents

You can view the contents, including specific credentials, of a client wallet external password store.

Listing the external password store contents provides information you can use to decide whether to add or delete credentials from the store.

 To list the contents of the external password store, enter the following command at the command line:

```
mkstore -wrl wallet location -listCredential
```

For example:

```
mkstore -wrl c:\oracle\product\19.1.0\db 1\wallets -listCredential
```

wallet_location specifies the path to the directory where the wallet, whose external password store contents you want to view, is located. This command lists all of the credential database service names (aliases) and the corresponding user name (schema) for that database. Passwords are not listed.

3.2.9.6.2 Adding Credentials to an External Password Store

You can store multiple credentials in one client wallet.

For example, if a client batch job connects to hr_database and a script connects to sales_database, then you can store the login credentials in the same client wallet. You cannot,

however, store multiple credentials (for logging in to multiple schemas) for the same database in the same wallet. If you have multiple login credentials for the same database, then they must be stored in separate wallets.

 To add database login credentials to an existing client wallet, enter the following command at the command line:

```
mkstore -wrl wallet location -createCredential db alias username
```

For example:

 $\label{lem:mkstore -wrl c:\oracle\product\19.1.0\db_1\wallets -createCredential orcl system} \\ Enter password: password$

In this specification:

- wallet_location is the path to the directory where the client wallet to which you want to add credentials is stored.
- db_alias can be the TNS alias you use to specify the database in the tnsnames.ora file or any service name you use to identify the database on an Oracle network.
- username is the database login credential for the schema to which your application connects. When prompted, enter the password for this user.

3.2.9.6.3 Modifying Credentials in an External Password Store

You can modify the database login credentials that are stored in the wallet if the database connection strings change.

 To modify database login credentials in a wallet, enter the following command at the command line:

```
mkstore -wrl wallet location -modifyCredential db alias username
```

For example:

 $\label{lem:mkstore -wrl c:\archeven} $$ mkstore -wrl c:\archeven'19.1.0\db_1\wallets -modifyCredential sales_db Enter password: $$ password : password $$$

In this specification:

- wallet location is the path to the directory where the wallet is located.
- db_alias is a new or different alias you want to use to identify the database. It can be a TNS alias you use to specify the database in the tnsnames.ora file or any service name you use to identify the database on an Oracle network.
- *username* is the new or different database login credential. When prompted, enter the password for this user.

3.2.9.6.4 Deleting Credentials from an External Password Store

You can delete login credentials for a database from a wallet if the database no longer exists or to disable connections to a specific database.

 To delete database login credentials from a wallet, enter the following command at the command line:

```
mkstore -wrl wallet location -deleteCredential db alias
```

For example:

mkstore -wrl c:\oracle\product\19.1.0\db 1\wallets -deleteCredential orcl



In this specification:

- wallet location is the path to the directory where the wallet is located.
- *db_alias* is the TNS alias you use to specify the database in the tnsnames.ora file, or any service name you use to identify the database on an Oracle Database network.

3.2.10 Managing Passwords for Administrative Users

The passwords of administrative users have special protections, such as password files and password complexity functions.

- About Managing Passwords for Administrative Users
 The passwords of administrative users are stored outside of the database so that the users can be authenticated even when the database is not open.
- Setting the LOCK and EXPIRED Status of Administrative Users
 Administrative users whose accounts have been locked cannot connect to the database.
- Password Profile Settings for Administrative Users
 There are several user profile password settings that are enforced for administrative users.
- Last Successful Login Time for Administrative Users
 The last successful login time of administrative user connections that use password file-based authentication is captured.
- Management of the Password File of Administrative Users
 Setting the ORAPWD utility FORMAT parameter to 12.2 enables you to manage the password profile parameters for administrative users.
- Migration of the Password File of Administrative Users
 The ORAPWD utility input_file parameter or DBUA can be used to migrate from earlier password file formats to the 12 or 12.2 format.
- How the Multitenant Option Affects Password Files for Administrative Users
 In a multitenant environment, the password information for the local and common administrative users is stored in different locations.
- Password Complexity Verification Functions for Administrative Users
 For better security, use password complexity verification functions for the passwords of administrative users.

3.2.10.1 About Managing Passwords for Administrative Users

The passwords of administrative users are stored outside of the database so that the users can be authenticated even when the database is not open.

There is no special protection with the password file. The password verifiers must be stored outside of the database so that authentication can be performed even when the database is not open. In previous releases, password complexity functions were available for non-administrative users only. Starting with Oracle Database release 12c (12.2), password complexity functions can be used for both non-administrative users and administrative users.

3.2.10.2 Setting the LOCK and EXPIRED Status of Administrative Users

Administrative users whose accounts have been locked cannot connect to the database.

To unlock locked or expired administrative accounts, use the ALTER USER statement.



For example:

```
ALTER USER hr admin ACCOUNT UNLOCK;
```

If the administrative user's password has expired, then the next time the user attempts to log in, the user will be prompted to create a new password.

3.2.10.3 Password Profile Settings for Administrative Users

There are several user profile password settings that are enforced for administrative users.

These password profile parameters are as follows:

- FAILED LOGIN ATTEMPT
- INACTIVE_ACCOUNT_TIME
- PASSWORD LOCK TIME
- PASSWORD LIFE TIME
- PASSWORD GRACE TIME

Related Topics

Managing Resources with Profiles

A profile is a named set of resource limits and password parameters that restrict database usage and instance resources for a user.

3.2.10.4 Last Successful Login Time for Administrative Users

The last successful login time of administrative user connections that use password file-based authentication is captured.

To find this login time, query the LAST_LOGIN column of the V\$PWFILE_USERS dynamic performance view.

3.2.10.5 Management of the Password File of Administrative Users

Setting the ORAPWD utility FORMAT parameter to 12.2 enables you to manage the password profile parameters for administrative users.

The password file is particularly important for administrative users because it stores the administrative user's credentials in an external file, not in the database itself. This enables the administrative user to log in to a database that is not open and perform tasks such as querying the data dictionary views. To create the password file, you must use the ORAPWD utility.

The FORMAT parameter setting of 12.2, which is the default setting, enables the password file to accommodate the password profile information for the administrative user.

For example:

```
orapwd file=orapworcl input_file=orapwold format=12.2
```

Setting FORMAT to 12.2 enforces the following rules:

 The password contains no fewer than 8 characters and includes at least one numeric and one alphabetic character.

- The password does not contain the user name or the user name reversed.
- The password does not contain the word oracle (such as oracle123).
- The password contains at least 1 special character.

FORMAT=12.2 also applies the following internal checks:

- The password does not exceed 30 bytes.
- The password does not contain the double-quotation character ("). However, it can be surrounded by double-quotation marks.

The following user profile password settings are enforced for administrative users:

- FAILED LOGIN ATTEMPT
- INACTIVE ACCOUNT TIME
- PASSWORD GRACE TIME
- PASSWORD LIFE TIME
- PASSWORD_LOCK_TIME

You can find the administrative users who have been included in the password file and their administrative privileges by querying the V\$PWFILE USERS dynamic view.

3.2.10.6 Migration of the Password File of Administrative Users

The ORAPWD utility input_file parameter or DBUA can be used to migrate from earlier password file formats to the 12 or 12.2 format.

You can migrate from earlier password file formats to the 12 or 12.2 format by using either the ORAPWD utility file and input_file parameters, or by using Oracle Database Upgrade Assistant (DBUA).

• The ORAPWD FILE and INPUT_FILE parameters: To migrate using the ORAPWD utility, set the FILE parameter to a name for the new password file and the INPUT_FILE parameter to the name of the earlier password file.

For example:

```
orapwd file=orapworcl input_file=orapwold format=12.2
```

DBUA: To migrate from the earlier formats of password files (FORMAT = LEGACY and FORMAT = 12), you can use the DBUA when you upgrade an earlier database to the current release. However, ensure that the database is open in read-only mode. You can check the database read-only status by querying the OPEN_MODE column of the V\$DATABASE dynamic view.

Related Topics

Oracle Database Administrator's Guide

3.2.10.7 How the Multitenant Option Affects Password Files for Administrative Users

In a multitenant environment, the password information for the local and common administrative users is stored in different locations.

 For CDB administrative users: The password information (hashes of the password) for the CDB common administrative users to whom administrative privileges were granted in the CDB root is stored in the password file. • For all users in a CDB to whom administrative privileges were granted outside the CDB root: To view information about the password hash information of these users, query the \$PWFILE USERS dynamic view.

3.2.10.8 Password Complexity Verification Functions for Administrative Users

For better security, use password complexity verification functions for the passwords of administrative users.

Note the following:

- Profiles: You can specify a password complexity verification function for the SYS user by
 using the PASSWORD_VERIFY_FUNCTION clause of the CREATE PROFILE or ALTER PROFILE
 statement. Oracle recommends that you use password verification functions to better
 protect the passwords of administrative users.
- ORAPWD password files: If you created a password file using the ORAPWD utility, then
 Oracle Database enforces password complexity checking for the SYS user and for
 administrative users who have logged in using the SYSDBA, SYSBACKUP, SYSDG, and SYSKM
 administrative privileges.

The password checks for the following requirements:

- The password contains no fewer than 8 characters and includes at least one numeric character, one alphabetic character, and one special character.
- The password is not the same as the user name or the user name reversed.
- The password does not contain the word oracle (such as oracle123).
- The password differs from the previous password by at least three characters.

The following internal checks are also applied:

- The password does not exceed 30 bytes.
- The password does not contain the double-quotation character ("). However, it can be surrounded by double-quotation marks.

Related Topics

Managing the Complexity of Passwords
 Oracle Database provides a set of functions that you can use to manage the complexity of passwords.

3.3 Authentication of Database Administrators

You can authenticate database administrators by using strong authentication, from the operating system, or from the database using passwords.

- About Authentication of Database Administrators
 Database administrators perform special administrative operations, such as shutting down or starting databases.
- Strong Authentication, Centralized Management for Administrators
 Strong authentication methods for centrally managed databases include directory authentication, Kerberos authentication, and SSL authentication.
- Authentication of Database Administrators by Using the Operating System
 For both Windows and UNIX systems, you use DBA-privileged groups to authenticate for
 the operating system.



- Authentication of Database Administrators by Using Their Passwords Password files are used to authenticate database administrators.
- Risks of Using Password Files for Database Administrator Authentication Be aware that using password files may pose security risks.

3.3.1 About Authentication of Database Administrators

Database administrators perform special administrative operations, such as shutting down or starting databases.

Oracle Database provides methods to secure the authentication of database administrators who have the SYSDBA, SYSOPER, SYSBACKUP, SYSDG, or SYSKM administrative privilege.

3.3.2 Strong Authentication, Centralized Management for Administrators

Strong authentication methods for centrally managed databases include directory authentication, Kerberos authentication, and SSL authentication.

- About Strong Authentication for Database Administrators
 Strong authentication lets you centrally control SYSDBA and SYSOPER access to multiple databases.
- Configuring Directory Authentication for Administrative Users
 Oracle Internet Directory configures directory authentication for administrative users.
- Configuring Kerberos Authentication for Administrative Users
 Oracle Internet Directory can be used to configure Kerberos authentication for administrative users.
- Configuring User Authentication with Transport Layer Security
 Both the client and server side can authenticate administrative users with Transport Layer Security (TLS).

3.3.2.1 About Strong Authentication for Database Administrators

Strong authentication lets you centrally control SYSDBA and SYSOPER access to multiple databases.

Consider using this type of authentication for database administration for the following situations:

- You have concerns about password file vulnerability.
- Your site has very strict security requirements.
- You want to separate the identity management from your database. By using a directory server such as Oracle Internet Directory (OID), for example, you can maintain, secure, and administer that server separately.

To enable the Oracle Internet Directory server to authorize SYSDBA and SYSOPER connections, use one of the following methods described in this section, depending on your environment.

3.3.2.2 Configuring Directory Authentication for Administrative Users

Oracle Internet Directory configures directory authentication for administrative users.

1. Configure the administrative user by using the same procedures you would use to configure a typical user.

2. In Oracle Internet Directory, grant the SYSDBA or SYSOPER administrative privilege to the user for the database that this user will administer.

Grant SYSDBA or SYSOPER only to trusted users.

3. Set the LDAP DIRECTORY SYSAUTH initialization parameter to YES:

```
ALTER SYSTEM SET LDAP_DIRECTORY_SYSAUTH = YES;
```

When set to YES, the LDAP_DIRECTORY_SYSAUTH parameter enables SYSDBA and SYSOPER users to authenticate to the database by using a strong authentication method.

4. Set the LDAP DIRECTORY ACCESS parameter to either PASSWORD or SSL. For example:

```
ALTER SYSTEM SET LDAP DIRECTORY ACCESS = PASSWORD;
```

Ensure that the LDAP_DIRECTORY_ACCESS initialization parameter is not set to NONE. Setting this parameter to PASSWORD or SSL ensures that users can be authenticated using the SYSDBA or SYSOPER administrative privileges through Oracle Internet Directory.

In an Oracle Real Application Clusters (Oracle RAC) environment, ensure that all instances have the same LDAP_DIRECTORY_ACCESS setting, either through the ALTER SYSTEM statement or through the init.ora file.

In an Oracle Data Guard or Active Data Guard environment, ensure that the standby database has the same <code>LDAP_DIRECTORY_ACCESS</code> setting as the primary database. In this environment, the <code>ALTER_SYSTEM</code> statement propagates its settings from the primary database to the standby database. If you choose to update the <code>init.ora</code> file, remember that the <code>init.ora</code> parameters are used by both the primary database and the standby database, so you do not need to manually propagate this setting from one database to the other.

Afterward, this user can log in by including the net service name in the CONNECT statement in SQL*Plus. For example, to log on as SYSDBA if the net service name is orcl:

```
CONNECT someuser@orcl AS SYSDBA Enter password: password
```

If the database is configured to use a password file for remote authentication, Oracle Database checks the password file first.

Related Topics

- Guidelines for Securing User Accounts and Privileges
 Oracle provides guidelines to secure user accounts and privileges.
- Oracle Database Reference
- Oracle Database Reference

3.3.2.3 Configuring Kerberos Authentication for Administrative Users

Oracle Internet Directory can be used to configure Kerberos authentication for administrative users.

 Configure the administrative user by using the same procedures you would use to configure a typical user.

See Configuring Kerberos Authentication, for more information.

Configure Oracle Internet Directory for Kerberos authentication.

See Oracle Database Enterprise User Security Administrator's Guide for more information.

3. In Oracle Internet Directory, grant the SYSDBA or SYSOPER administrative privilege to the user for the database that this user will administer.

Grant SYSDBA or SYSOPER only to trusted users. See Guidelines for Securing User Accounts and Privileges for advice on this topic.

4. Set the LDAP DIRECTORY SYSAUTH **initialization parameter to** YES:

```
ALTER SYSTEM SET LDAP_DIRECTORY_SYSAUTH = YES;
```

When set to YES, the LDAP_DIRECTORY_SYSAUTH parameter enables SYSDBA and SYSOPER users to authenticate to the database by using strong authentication methods. See *Oracle Database Reference* for more information about LDAP DIRECTORY SYSAUTH.

5. Set the LDAP DIRECTORY ACCESS parameter to either PASSWORD or SSL. For example:

```
ALTER SYSTEM SET LDAP DIRECTORY ACCESS = SSL;
```

Ensure that the LDAP_DIRECTORY_ACCESS initialization parameter is not set to NONE. Setting this parameter to PASSWORD or SSL ensures that users can be authenticated using SYSDBA or SYSOPER through Oracle Internet Directory. See *Oracle Database Reference* for more information about LDAP DIRECTORY ACCESS.

In an Oracle Real Application Clusters (Oracle RAC) environment, ensure that all instances have the same LDAP_DIRECTORY_ACCESS setting, either through the ALTER SYSTEM statement or through the init.ora file.

In an Oracle Data Guard or Active Data Guard environment, ensure that the standby database has the same LDAP_DIRECTORY_ACCESS setting as the primary database. In this environment, the ALTER SYSTEM statement propagates its settings from the primary database to the standby database. If you choose to update the init.ora file, remember that the init.ora parameters are used by both the primary database and the standby database, so you do not need to manually propagate this setting from one database to the other.

Afterward, this user can log in by including the net service name in the CONNECT statement in SQL*Plus. For example, to log on as SYSDBA if the net service name is orcl:

```
CONNECT /@orcl AS SYSDBA
```

3.3.2.4 Configuring User Authentication with Transport Layer Security

Both the client and server side can authenticate administrative users with Transport Layer Security (TLS).

- 1. For both the client and the server, get user certificates signed by the same root Certificate Authority (CA) certificate, either public or self-signed.
- 2. Configure the client to use TLS:
 - a. Add the signed user certificate to the client wallet. The CA root trust certificate should already be in the client wallet. Ensure that any intermediate certificates that are required for the user certificate are added to the wallet before you add the user certificate.

You can use orapki to configure the client wallet and user certificate.

b. Set TLS as an authentication service in the sqlnet.ora file.

```
SSL CLIENT AUTHENTICATION=TRUE
```



c. Optionally, for better security, set the client to use full or partial DN matching.

When DN matching is enabled, the client will check the server certificate to ensure that host names will match what the client is configured to match. You perform this step when you enable Oracle Internet Directory to use TLS.



The database client and server will use the strongest TLS protocol and cipher suite to establish a connection. Therefore, you do not need to specify the TLS version and cipher suites unless you have specific security requirements that require it. Be aware that if you set specific TLS versions and cipher suites, you will need to update the configuration when the older versions are no longer used.

- 3. Configure the listener for TLS on the client, the listener, and server.
 - a. Create a separate listener entry for TLS connections using the secure database port 1522.

For example:

- **b.** Comment out the non-TLS listener entry (for example, the line with PROTOCOL = TCP) or leave it in for non-TLS required connections.
- **c.** Add SSL_CLIENT_AUTHENTICATION = FALSE to the sqlnet.ora file so the database server authenticates the client, not the listener.

The same wallet that the server uses can be used by the listener, along with the same server certificate. The listener will look for the wallet using the standard Oracle Database wallet search order. Alternatively, you can specify the wallet location in the listener by setting the WALLET_LOCATION parameter. (You cannot use the WALLET_ROOT parameter for this purpose, because the listener cannot use it.)

- 4. Configure the server to use TLS:
 - a. In the sqlnet.ora file, set SSL_CLIENT_AUTHENTICATION to FALSE (or OFF) to enable one-way TLS.
 - **b.** For the TLS server wallet, do the following:
 - Set the WALLET ROOT parameter to a location for the TLS server.
 - Create the tls directory under WALLET ROOT/pdb guid.
 - Move the TLS server wallet to the WALLET ROOT/pdb guid/tls directory.
 - c. In the sqlnet.ora file, add the following parameter:

```
SSL CLIENT AUTHENTICATION=TRUE
```



If you want to restrict authentication to only TCPS, then set AUTHENTICATION_SERVICES to TCPS.

5. Create a new schema or alter an existing schema to map to the user.

```
CREATE USER user name IDENTIFIED EXTERNALLY AS 'user DN on certificate';
```

6. Grant the database schema to appropriate administrative privileges, such as SYSDBA, SYSOPER, and so on.

Administrative users with TLS authentication can authenticate with TLS. To enable these users, grant the appropriate administrative privilege to the user schema. The administrative user must log in using this administrative privilege. For example, for a user who was granted the SYSOPER administrative privilege:

```
CONNNECT /@pdb name AS SYSOPER
```

Afterward, this user can log in by including the net service name in the CONNECT statement in SQL*Plus. For example, to log on as SYSDBA if the net service name is orcl:

CONNECT /@orcl AS SYSDBA

3.3.3 Authentication of Database Administrators by Using the Operating System

For both Windows and UNIX systems, you use DBA-privileged groups to authenticate for the operating system.

Operating system authentication for a database administrator typically involves establishing a group on the operating system, granting DBA privileges to that group, and then adding the names of persons who should have those privileges to that group. (On UNIX systems, the group is the **dba** group.)



In a multitenant environment, you can use operating system authentication for a database administrator only for the CDB root. You cannot use it for for PDBs, the application root, or application PDBs.

On Microsoft Windows systems:

- Users who connect with the SYSDBA administrative privilege can take advantage of the Windows native authentication. If these users work with Oracle Database using their domain accounts, then you must explicitly grant them local administrative privileges and ORA DBA membership.
- Oracle recommends that you run Oracle Database services using a low privileged Microsoft Windows user account rather than a Microsoft Windows built-in account.

See Also:

Your Oracle Database operating system-specific documentation for information about configuring operating system authentication of database administrators

3.3.4 Authentication of Database Administrators by Using Their Passwords

Password files are used to authenticate database administrators.

That is, Oracle Database users who have been granted the SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM administrative privileges are first authenticated using database-specific password files.

These privileges enable the following activities:

- The SYSOPER system privilege lets database administrators perform STARTUP, SHUTDOWN, ALTER DATABASE OPEN/MOUNT, ALTER DATABASE BACKUP, ARCHIVE LOG, and RECOVER operations. SYSOPER also includes the RESTRICTED SESSION privilege.
- The SYSDBA administrative privilege has all system privileges with ADMIN OPTION, including
 the SYSOPER administrative privilege, and permits CREATE DATABASE and time-based
 recovery.
- A password file containing users who have the SYSDBA, SYSDER, SYSASM, SYSBACKUP, SYSDG, and SYSKM administrative privileges can be shared between different databases. In addition, this type of password file authentication can be used in a Transport Layer Security (TLS) or Kerberos configuration, and for common administrative users in a multitenant environment. You can have a shared password file that contains users in addition to the SYS user. To share a password file among different databases, set the REMOTE LOGIN PASSWORDFILE parameter in the init.ora file to SHARED.

If you set the REMOTE_LOGIN_PASSWORDFILE initialization parameter to EXCLUSIVE or SHARED from NONE, then ensure that the password file is synchronized with the dictionary passwords. See *Oracle Database Administrator*'s *Guide* for more information.

- For Automatic Storage Management (ASM) environments, you can create shared ASM password files. Remember that you must have the SYSASM system privilege to create an ASM password file. See *Oracle Automatic Storage Management Administrator's Guide* for more information.
- The SYSDG administrative privilege must be included in a password file for sharding administrators to perform tasks that involve file transfer and Oracle Recovery Manager (RMAN) activities.
- Password file-based authentication is enabled by default. This means that the database is
 ready to use a password file for authenticating users that have SYSDBA, SYSOPER, SYSASM,
 SYSBACKUP, SYSDG, and SYSKM administrative privileges. Password file-based authentication
 is activated as soon as you create a password file by using the ORAPWD utility.
 - Anyone who has EXECUTE privileges and write privileges to the <code>\$ORACLE_HOME/dbs</code> directory can run the <code>ORAPWD</code> utility.
- Password limits such as FAILED_LOGIN_ATTEMPTS and PASSWORD_LIFE_TIME are enforced
 for administrative logins, if the password file is created in the Oracle Database 12c release
 2 (12.2) format.



Note:

- To find a list of users who are included in the password file, you can query the V\$PWFILE USERS data dictionary view.
- Connections requested AS SYSDBA or AS SYSOPER must use these phrases. Without them, the connection fails.

3.3.5 Risks of Using Password Files for Database Administrator Authentication

Be aware that using password files may pose security risks.

For this reason, consider using the authentication methods described in Strong Authentication, Centralized Management for Administrators.

Examples of password security risks are as follows:

- An intruder could steal or attack the password file.
- Many users do not change the default password.
- The password could be easily guessed.
- The password is vulnerable if it can be found in a dictionary.
- Passwords that are too short, chosen perhaps for ease of typing, are vulnerable if an intruder obtains the cryptographic hash of the password.



Oracle Database Administrator's Guide for information about creating and maintaining password files

3.4 Database Authentication of Users

Database authentication of users entails using information within the database itself to perform the authentication.

- About Database Authentication of Users
 Oracle Database can authenticate users attempting to connect to a database by using information stored in that database itself.
- Advantages of Database Authentication
 There are three advantages of using the database to authenticate users.
- Creating Users Who Are Authenticated by the Database
 When you create a user who is authenticated by the database, you assign this user a
 password.



3.4.1 About Database Authentication of Users

Oracle Database can authenticate users attempting to connect to a database by using information stored in that database itself.

To configure Oracle Database to use database authentication, you must create each user with an associated password. User names can use the National Language Support (NLS) character format, but you cannot include double quotation mark characters in the password. The user must provide this user name and password when attempting to establish a connection.

Oracle Database generates a one-way hash of the user's password and stores it for use when verifying the provided login password. In order to support older clients, Oracle Database can be configured to generate the one-way hash of the user's password using a variety of different hashing algorithms. The resulting password hashes are known as password versions, which have the short names 10G, 11G, and 12C. The short names 10G, 11G, and 12C serve as abbreviations for the details of the one-way password hashing algorithms, which are described in more detail in the documentation for the PASSWORD_VERSIONS column of the DBA_USERS view. To find the list of password versions for any given user, query the PASSWORD_VERSIONS column of the DBA_USERS view.

By default, there are currently two versions of the one-way hashing algorithm in use in Oracle Database: the salted SHA-1 hashing algorithm, and the salted PKBDF2 SHA-2 SHA-512 hashing algorithm. The salted SHA-1 hashing algorithm generates the hash that is used for the 11G password version. The salted PKBDF2 SHA-2 SHA-512 hashing algorithm generates the hash that is used for the 12C password version. This hash generation takes place for the same password; that is, both algorithms run for the same password. Oracle Database records these password versions in the DBA_USERS data dictionary view. When you query this view, you will see two password versions. For example:

SELECT USERNAME, PASSWORD_VERSIONS FROM DBA_USERS;

```
USERNAME PASSWORD_VERSIONS
------
ADAMS 11G, 12C
SYS 11G, 12C
```

To specify which authentication protocol to allow during authentication of a client or of a database server acting as a client, you can explicitly set the

SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter in the server sqlnet.ora file. (The client version of this parameter is SQLNET.ALLOWED_LOGON_VERSION_CLIENT.) Each connection attempt is tested, and if the client or server does not meet the client ability requirements specified by its partner, authentication fails with an ORA-28040 No matching authentication protocol error in the "Ability Required of the Client" in the

"SQLNET.ALLOWED_LOGON_VERSION_SERVER Settings" table under the description of the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter in *Oracle Database Net Services**Reference.* The parameter can take the values 12a, 12, 11, 10, 9, or 8. The default value is 12, which is Exclusive Mode. These values represent the version of the authentication protocol. Oracle recommends the value 12. However, be aware that if you set

SQLNET.ALLOWED_LOGON_VERSION_SERVER and SQLNET.ALLOWED_LOGON_VERSION_CLIENT to 11, then pre-Oracle Database Release 11.1 client applications including JDBC thin clients cannot authenticate to the Oracle database using password-based authentication.

To enhance security when using database authentication, Oracle recommends that you use password management, including account locking, password aging and expiration, password history, and password complexity verification.

If you are not using external authentication and only using local database password authentication, then set ${\tt AUTHENTICATION_SERVICES=(none)}$ in the client ${\tt sqlnet.ora}$ file. This setting improves performance because the default for this value is ${\tt ALL}$, which forces the client to check external authentication as well as database password authentication.

Related Topics

- SQLNET.ALLOWED_LOGON_VERSION_CLIENT
- SQLNET.ALLOWED_LOGON_VERSION_SERVER
- SQLNET.AUTHENTICATION SERVICES
- About Password Complexity Verification
 Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.
- Using a Password Management Policy
 A password management policy can create and enforce a set of restrictions that can better secure user passwords.
- Management of Password Versions of Users
 By default, Oracle Database uses Exclusive Mode, which does not permit case-insensitive passwords, to manage password versions.

3.4.2 Advantages of Database Authentication

There are three advantages of using the database to authenticate users.

These advantages are as follows:

- User accounts and all authentication are controlled by the database. There is no reliance on anything outside of the database.
- Oracle Database provides strong password management features to enhance security when using database authentication.
- It is easier to administer when there are small user communities.

3.4.3 Creating Users Who Are Authenticated by the Database

When you create a user who is authenticated by the database, you assign this user a password.

• To create a user who is authenticated by the database, include the IDENTIFIED BY clause when you create the user.

For example, the following SQL statement creates a user who is identified and authenticated by Oracle Database. User sebastian must specify the assigned password whenever he connects to Oracle Database.

CREATE USER sebastian IDENTIFIED BY password;

Related Topics

Creating User Accounts

A user account can have restrictions such as profiles, a default role, and tablespace restrictions.



3.5 Schema-Only Accounts

You can create schema-only accounts, that is, the schema user has no password.

- About Schema-Only Accounts
 A schema-only account cannot log in to the database but can proxy in a single session proxy.
- Creating a Schema-Only Account
 The CREATE USER SQL statement creates schema-only accounts.
- Altering a Schema-Only Account
 The ALTER USER SQL statement can be used to modify schema-only accounts.

3.5.1 About Schema-Only Accounts

A schema-only account cannot log in to the database but can proxy in a single session proxy.

This type of account, designed for some Oracle-provided schemas along with some user-created schemas, can be created without the specification of a password or an authentication type. It cannot be authenticated unless an authentication method is assigned by using the ALTER USER statement. A schema-only account does not contain an entry in the DBA USERS WITH DEFPWD data dictionary view.

By default, most of the predefined schema user accounts that are available with Oracle Database, such as the sample schema user accounts (for example, HR), are schema-only accounts. You can assign these accounts passwords if you want to, but for better security, Oracle recommends that you set them back to being schema-only afterwards. To check if a schema user account is schema only, query the AUTHENTICATION_TYPE column of the DBA USERS data dictionary view. NONE indicates that the account is schema only.

Note the following rules about using schema only accounts:

- Schema only accounts can be used for both administrator and non-administrator accounts.
- Schema only accounts must be created on the database instance only, not in Oracle Automatic Storage Management (ASM) environments.
- You can grant system privileges (such as CREATE ANY TABLE) and administrator roles (such
 as DBA) to schema only accounts. Schema only accounts can create objects such as tables
 or procedures, assuming they have had to correct privileges granted to them.
- You can configure schema only accounts to be used as client users in a proxy
 authentication in a single session proxy. This is because in a single session proxy, only the
 credentials of the proxy user are verified, not the credentials of the client user. Therefore, a
 schema only account can be a client user. However, you cannot configure schema only
 accounts for a two-proxy scenario, because the client credentials must be verified. Hence,
 the authentication for a schema only account will fail.
- Schema only accounts cannot connect through database links, either with connected user links, fixed user links, or current user links.

Related Topics

Predefined Sample Schema User Accounts
 Oracle Database creates a set of sample user accounts if you install the sample schemas.



3.5.2 Creating a Schema-Only Account

The CREATE USER SQL statement creates schema-only accounts.

You can run the CREATE USER statement with the NO AUTHENTICATION clause only on a database instance. You cannot run it on an Oracle Automatic Storage Management (ASM) instance.

Use the CREATE USER statement with the NO AUTHENTICATION clause.

For example:

CREATE USER psmith NO AUTHENTICATION;

3.5.3 Altering a Schema-Only Account

The ALTER USER SQL statement can be used to modify schema-only accounts.

Check if the schema user has administrative privileges.

You can query the V\$PWFILE USERS to find if the schema user has administrative privileges.

- 2. If the schema user has administrative privileges, then use the REVOKE statement to revoke these privileges.
- 3. Use the ALTER USER SQL statement with the NO AUTHENTICATION clause to modify the schema account to have no authentication.

For example:

ALTER USER psmith NO AUTHENTICATION;

You can use ALTER USER to enable authentication for a schema-only account.

3.6 Operating System Authentication of Users

Oracle Database can authenticate by using information that is maintained by the operating system.

Using the operating system to authenticate users has both advantages and disadvantages.

This functionality has the following benefits:

Once authenticated by the operating system, users can connect to Oracle Database more
conveniently, without specifying a user name or password. For example, an operating
system-authenticated user can invoke SQL*Plus and omit the user name and password
prompts by entering the following command at the command line:

SQLPLUS /

Within SQL*Plus, you enter:

CONNECT /

- With control over user authentication centralized in the operating system, Oracle Database does not need to store or manage the cryptographic hashes (also called verifiers) of the user passwords, although it still maintains user names in the database.
- The audit trail captures the operating system user name and the database user name, where the database user name is the value of the OS_AUTHENT_PREFIX instance initialization parameter prefixed to the operating system user name. For example, if

OS_AUTHENT_PREFIX is set to OPS\$ and the operating system user name is psmith, then the database user name will be OPS\$PSMITH.

- You can authenticate both operating system and non-operating system users in the same system. For example:
 - Authenticate users by the operating system. You create the user account using the
 IDENTIFIED EXTERNALLY clause of the CREATE USER statement, and then you set the
 OS_AUTHENT_PREFIX initialization parameter to specify a prefix that Oracle Database
 uses to authenticate users attempting to connect to the server.
 - Authenticate non-operating system users. These are users who are assigned passwords and authenticated by the database.
 - Authenticate Oracle Database Enterprise User Security users. These user
 accounts where created using the IDENTIFIED GLOBALLY clause of the CREATE USER
 statement, and then authenticated by Oracle Internet Directory (OID) currently in the
 same database.

However, you should be aware of the following drawbacks to using the operating system to authenticate users:

- A user must have an operating system account on the computer that must be accessed.
 Not all users have operating system accounts, particularly non-administrative users.
- If a user has logged in using this method and steps away from the terminal, another user could easily log in because this user does not need any passwords or credentials. This could pose a serious security problem.
- When an operating system is used to authenticate database users, managing distributed database environments and database links requires special care. Operating systemauthenticated database links can pose a security weakness. For this reason, Oracle recommends that you do not use them.
- In a multitenant environment, you can use operating system authentication for a database administrator only for the CDB root. You cannot use it for PDBs, the application root, or application PDBs.

See Also:

- Oracle Database Administrator's Guide for more information about authentication, operating systems, distributed database concepts, and distributed data management
- Operating system-specific documentation by Oracle Database for more information about authenticating by using your operating system

3.7 Network Authentication of Users

You can authenticate users over a network by using Transport Layer Security with third-party services.

Authentication with Transport Layer Security
 The Transport Layer Security (TLS) protocol is an application layer protocol.



Authentication with Third-Party Services

The third-party services Kerberos, RADIUS, directory-based services, and public key infrastructure can authenticate Oracle Database over a network.

3.7.1 Authentication with Transport Layer Security

The Transport Layer Security (TLS) protocol is an application layer protocol.

You can use TLS for user authentication to a database, and it is independent of global user management in Oracle Internet Directory. That is, users can use TLS to authenticate to the database without a directory server in place.

Related Topics

Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.

3.7.2 Authentication with Third-Party Services

The third-party services Kerberos, RADIUS, directory-based services, and public key infrastructure can authenticate Oracle Database over a network.

- About Authentication Using Third-Party Services
 You must use third-party network authentication services if you want to authenticate Oracle
 Database users over a network.
- Authentication with Kerberos
 Kerberos is a trusted third-party authentication system that relies on shared secrets.
- Authentication with RADIUS
 Remote Authentication Dial-In User Service (RADIUS) is a standard lightweight protocol used for user authentication, authorization, and accounting.
- Authentication with Directory-Based Services
 Using a central directory can make authentication and its administration efficient.
- Authentication with Public Key Infrastructure
 Authentication systems based on public key infrastructure (PKI) issue digital certificates to user clients.

3.7.2.1 About Authentication Using Third-Party Services

You must use third-party network authentication services if you want to authenticate Oracle Database users over a network.

Prominent examples include Kerberos, PKI (public key infrastructure), the RADIUS (Remote Authentication Dial-In User Service), and directory-based services.

If network authentication services are available to you, then Oracle Database can accept authentication from the network service. If you use a network authentication service, then some special considerations arise for network roles and database links.

3.7.2.2 Authentication with Kerberos

Kerberos is a trusted third-party authentication system that relies on shared secrets.

Kerberos presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does

this through a Kerberos authentication server, or through Cybersafe Active Trust, a commercial Kerberos-based authentication server.

Related Topics

Configuring Kerberos Authentication
 Kerberos is a trusted third-party authentication system that relies on shared secrets and presumes that the third party is secure.

3.7.2.3 Authentication with RADIUS

Remote Authentication Dial-In User Service (RADIUS) is a standard lightweight protocol used for user authentication, authorization, and accounting.

RADIUS also enables users to use the RSA One-Time Password Specifications (OTPS) to authenticate to the Oracle database.

See Also:

- · Configuring RADIUS Authentication for information about configuring RADIUS
- RSA documentation about OTPS

3.7.2.4 Authentication with Directory-Based Services

Using a central directory can make authentication and its administration efficient.

Directory-based services include the following:

- Oracle Internet Directory, which uses the Lightweight Directory Access Protocol (LDAP), uses a central repository to store and manage information about users (called enterprise users) whose accounts were created in a distributed environment. Although database users must be created (with passwords) in each database that they need to access, enterprise user information is accessible centrally in the Oracle Internet Directory. You can also integrate this directory with Microsoft Active Directory and SunOne.
- Oracle Enterprise Security Manager lets you store and retrieve roles from Oracle Internet Directory, which provides centralized privilege management to make administration easier and increase security levels.

3.7.2.5 Authentication with Public Key Infrastructure

Authentication systems based on public key infrastructure (PKI) issue digital certificates to user clients.

These clients can use these certificates to authenticate directly to servers in the enterprise without directly involving an authentication. Oracle Database provides a PKI for using public keys and certificates, consisting of the following components:

- Authentication and secure session key management using SSL. See Authentication with Transport Layer Security for more information.
- Trusted certificates. These are used to identify third-party entities that are trusted as
 signers of user certificates when an identity is being validated. When the user certificate is
 being validated, the signer is checked by using trust points or a trusted certificate chain of
 certificate authorities stored in the validating system. If there are several levels of trusted

certificates in this chain, then a trusted certificate at a lower level is simply trusted without needing to have all its higher-level certificates reverified.

Oracle Wallet Manager. An Oracle wallet is a data structure that contains the private key
of a user, a user certificate, and the set of trust points of a user (trusted certificate
authorities). See Oracle Database Enterprise User Security Administrator's Guide for
information about managing Oracle wallets.

You can use Oracle Wallet Manager to manage Oracle wallets. This is a standalone Java application used to manage and edit the security credentials in Oracle wallets. It performs the following operations:

- Generates a public-private key pair and creates a certificate request for submission to a certificate authority, and creates wallets
- Installs a certificate for the entity
- Manages X.509 version 3 certificates on Oracle Database clients and servers
- Configures trusted certificates for the entity
- Opens a wallet to enable access to PKI-based services
- X.509 version 3 certificates obtained from (and signed by) a trusted entity, a
 certificate authority. Because the certificate authority is trusted, these certificates verify
 that the requesting entity's information is correct and that the public key on the certificate
 belongs to the identified entity. The certificate is loaded into an Oracle wallet to enable
 future authentication.

3.8 Configuring Operating System Users for a PDB

The DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure configures user accounts to be operating system users for a PDB.

- About Configuring Operating System Users for a PDB
 Instead the oracle operating system user, you can set a specific user account to be the operating system user for that PDB.
- Configuring an Operating System User for a PDB
 The DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure can set an operating system user for a PDB.

3.8.1 About Configuring Operating System Users for a PDB

Instead the oracle operating system user, you can set a specific user account to be the operating system user for that PDB.

If you do not set a specific user to be the operating system user for the PDB, then by default the PDB uses the <code>oracle</code> operating system user. For the root, you can use the <code>oracle</code> operating system user when you must interact with the operating system.

For better security, Oracle recommends that you set a unique operating system user for each PDB in a multitenant environment. Doing so helps to ensure that operating system interactions are performed as a less powerful user than the oracle operating system user, and helps to protect data that belongs to one PDB from being accessed by users who are connected to other PDBs.



3.8.2 Configuring an Operating System User for a PDB

The DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure can set an operating system user for a PDB.

1. Log in to the database instance root as a user who has the EXECUTE privilege for the DBMS CREDENTIAL PL/SQL package and the ALTER SYSTEM system privilege.

For example:

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

2. Run the DBMS_CREDENTIAL.CREATE_CREDENTIAL procedure to create an Oracle credential for the operating system user.

For example, to set the credential for a user named os admin:

```
BEGIN

DBMS_CREDENTIAL.CREATE_CREDENTIAL (
    credential_name => 'PDB1_OS_USER',
    username => 'os_admin',
    password => 'password');
END;
//
```

Connect to the PDB for which the operating system user will be used.

For example:

```
CONNECT cc##sec_admin@hrpdb
Enter password: password
```

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

 Set the PDB_OS_CREDENTIAL initialization parameter for the user whose credential was set in Step 2.

For example:

```
ALTER SYSTEM SET PDB OS CREDENTIAL = PDB1 OS USER SCOPE = SPFILE;
```

The PDB_OS_CREDENTIAL parameter is a static parameter, so you must set it using the SCOPE = SPFILE clause.

5. Restart the database instance.

```
SHUTDOWN IMMEDIATE STARTUP
```

Related Topics

Minimum Requirements for Passwords

Oracle provides a set of minimum requirements for passwords.

3.9 Global User Authentication and Authorization

Global user authentication and authorization enables you to centralize the management of user-related information.

- About Configuring Global User Authentication and Authorization
 An LDAP-based directory service centralizes the management of user-related information, including authorizations.
- Configuration of Users Who Are Authorized by a Directory Service
 You can configure either a global user or multiple enterprise users to be authorized by a directory service.
- Advantages of Global Authentication and Global Authorization
 There are several advantages of global user authentication and authorization.

3.9.1 About Configuring Global User Authentication and Authorization

An LDAP-based directory service centralizes the management of user-related information, including authorizations.

This enables users and administrators to be identified in the database as global users, meaning that they are authenticated by TLS and that the management of these users is handled outside of the database by the centralized directory service. Global roles are defined in a database and are known only to that database, but the directory service handles authorizations for global roles.



You can also have users authenticated by Transport Layer Security (TLS), whose authorizations are not managed in a directory, that is, they have local database roles only.

This centralized management enables the creation of **enterprise users** and **enterprise roles**. Enterprise users are defined and managed in the directory. They have unique identities across the enterprise and can be assigned enterprise roles that determine their access privileges across multiple databases. An enterprise role consists of one or more global roles, and might be thought of as a container for global roles.

You also can use centrally managed users to authenticate and authorize users through a directory service such as Microsoft Active Directory.

Related Topics

- Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.
- Strong Authentication, Centralized Management for Administrators
 Strong authentication methods for centrally managed databases include directory authentication, Kerberos authentication, and SSL authentication.
- Configuring Centrally Managed Users with Microsoft Active Directory
 Oracle Database can authenticate and authorize Microsoft Active Directory users with the
 database directly without intermediate directories or Oracle Enterprise User Security.

3.9.2 Configuration of Users Who Are Authorized by a Directory Service

You can configure either a global user or multiple enterprise users to be authorized by a directory service.

- Creating a Global User Who Has a Private Schema
 You can create a user account who has a private schema by providing an identifier (distinguished name, or DN) meaningful to the enterprise directory.
- Creating Multiple Enterprise Users Who Share Schemas
 Multiple enterprise users can share a single schema in the database.

3.9.2.1 Creating a Global User Who Has a Private Schema

You can create a user account who has a private schema by providing an identifier (distinguished name, or DN) meaningful to the enterprise directory.

However, be aware that you must create this user in every database that the user must access, plus the directory.

• To create a global user who has a private schema, use the CREATE USER ... IDENTIFIED GLOBALLY SQL statement.

You can include standard LDAP Data Interchange Format (LDIF) fields. For example, to create a global user (psmith_gl with a private schema, authenticated by SSL, and authorized by the enterprise directory service:

CREATE USER psmith gl IDENTIFIED GLOBALLY AS 'CN=psmith,OU=division1,O=example,C=US';

In this specification:

- CN refers to the common name of this user, psmith gl.
- OU refers to the user's organizational unit, division1.
- o refers to the user's organization, Example.
- C refers to the country in which the organization Example is located, the US.

3.9.2.2 Creating Multiple Enterprise Users Who Share Schemas

Multiple enterprise users can share a single schema in the database.

These users are authorized by the enterprise directory service but do not own individual private schemas in the database. These users are not individually created in the database. They connect to a shared schema in the database.

1. Create a shared schema in the database using the following example:

```
CREATE USER appschema IDENTIFIED GLOBALLY AS '';
```

2. In the directory, create multiple enterprise users and a mapping object.

The mapping object tells the database how you want to map the DNs for the users to the shared schema. You can either create a full distinguished name (DN) mapping (one directory entry for each unique DN), or you can map, for each user, multiple DN components to one schema. For example:

OU=division1,O=Example,C=US



See Also:

Oracle Database Enterprise User Security Administrator's Guide for an explanation of these mappings

Most users do not need their own schemas, and implementing schema-independent users separates users from databases. You create multiple users who share the same schema in a database, and as enterprise users, they can also access shared schemas in other databases.

3.9.3 Advantages of Global Authentication and Global Authorization

There are several advantages of global user authentication and authorization.

- Provides strong authentication using SSL, Kerberos, or Windows native authentication.
- Enables centralized management of users and privileges across the enterprise.
- Is easy to administer: You do not have to create a schema for every user in every database in the enterprise.
- Facilitates single sign-on: Users need to sign on once to only access multiple databases and services. Further, users using passwords can have a single password to access multiple databases accepting password-authenticated enterprise users.
- Because global user authentication and authorization provide password-based access, you
 can migrate previously defined password-authenticated database users to the directory
 (using the User Migration Utility) to be centrally administered. This makes global
 authentication and authorization available for earlier Oracle Database release clients that
 are still supported.
- CURRENT_USER database links connect as a global user. A local user can connect as a
 global user in the context of a stored procedure, that is, without storing the global user
 password in a link definition.

Related Topics

Oracle Database Enterprise User Security Administrator's Guide

3.10 Configuring an External Service to Authenticate Users and Passwords

An external service (the operating system or the network) can administer passwords and authenticate users.

- About External Authentication
 With external authentication, Oracle Database maintains the user account, but an external service performs the password administration and user authentication.
- Advantages of External Authentication
 External authentication provides several advantages.
- Enabling External Authentication
 To enable external authentication, you can set the initialization parameter
 OS AUTHENT PREFIX, and use this prefix in Oracle Database user names.



- Creating a User Who Is Authenticated Externally
 Externally authenticated users are authenticated by the operating system or network service.
- Authentication of User Logins By Using the Operating System
 Oracle Database allows operating system-authenticated logins only over secure connections, which precludes using Oracle Net and a shared server configuration.
- Authentication of User Logins Using Network Authentication
 Oracle strong authentication performs network authentication, which you can configure to use a third-party service such as Kerberos.

3.10.1 About External Authentication

With external authentication, Oracle Database maintains the user account, but an external service performs the password administration and user authentication.

This external service can be the operating system or a network service, such as Oracle Net. If you are authenticating users through a password file, then you can configure external authentication for users who have been granted the SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM administrative privileges.

With external authentication, your database relies on the underlying operating system or network authentication service to restrict access to database accounts. A database password is not used for this type of login. If your operating system or network service permits, then it can authenticate users before they can log in to the database.

You also can use centrally managed users to authenticate and authorize users through a directory service such as Microsoft Active Directory.

Related Topics

- Management of the Password File of Administrative Users
 Setting the ORAPWD utility FORMAT parameter to 12.2 enables you to manage the password profile parameters for administrative users.
- Configuring Centrally Managed Users with Microsoft Active Directory
 Oracle Database can authenticate and authorize Microsoft Active Directory users with the
 database directly without intermediate directories or Oracle Enterprise User Security.

3.10.2 Advantages of External Authentication

External authentication provides several advantages.

These advantages are as follows:

- More choices of authentication mechanisms are available, such as smart cards, fingerprints, Kerberos, or the operating system.
- Many network authentication services, such as Kerberos support single sign-on, enabling users to have fewer passwords to remember.
- If you are already using an external mechanism for authentication, such as one of those
 listed earlier, then there may be less administrative overhead to use that mechanism with
 the database.



3.10.3 Enabling External Authentication

To enable external authentication, you can set the initialization parameter <code>OS_AUTHENT_PREFIX</code>, and use this prefix in Oracle Database user names.

The OS_AUTHENT_PREFIX parameter defines a prefix that Oracle Database adds to the beginning of the operating system account name of every user. Oracle Database compares the prefixed user name with the Oracle Database user names in the database when a user attempts to connect.

1. Set OS_AUTHENT_PREFIX to a null string (an empty set of double quotation marks: ""). Using a null string eliminates the addition of any prefix to operating system account names, so that Oracle Database user names exactly match operating system user names.

For example:

```
OS AUTHENT PREFIX=""
```

2. Ensure that the OS_AUTHENT_PREFIX remains the same for the life of a database. If you change the prefix, then any database user name that includes the old prefix cannot be used to establish a connection, unless you alter the user name to have it use password authentication.

The default value of the OS_AUTHENT_PREFIX parameter is OPS\$ for backward compatibility with previous versions of Oracle Database. For example, assume that you set OS_AUTHENT_PREFIX as follows:

```
OS AUTHENT PREFIX=OPS$
```

If a user with an operating system account named tsmith is to connect to an Oracle database installation and be authenticated by the operating system, then Oracle Database checks that there is a corresponding database user OPS\$tsmith and, if so, lets the user connect. All references to a user authenticated by the operating system must include the prefix, OPS\$, as seen in OPS\$tsmith.



The text of the <code>OS_AUTHENT_PREFIX</code> initialization parameter is case-sensitive on some operating systems. See your operating system-specific Oracle Database documentation for more information about this initialization parameter.

3.10.4 Creating a User Who Is Authenticated Externally

Externally authenticated users are authenticated by the operating system or network service.

You can create users who are authenticated externally. Oracle Database then relies on this external login authentication when it provides that specific operating system user with access to the database resources of a specific user.

• Use the IDENTIFIED EXTERNALLY clause of the CREATE USER statement to create users who are authenticated externally.

The following example creates a user who is identified by Oracle Database and authenticated by the operating system or a network service. This example assumes that the OS AUTHENT PREFIX parameter has been set to a blank space (" ").



CREATE USER psmith IDENTIFIED EXTERNALLY;

3.10.5 Authentication of User Logins By Using the Operating System

Oracle Database allows operating system-authenticated logins only over secure connections, which precludes using Oracle Net and a shared server configuration.

This type of operating system authentication is the default. This restriction prevents a remote user from impersonating another operating system user over a network connection.

Setting the REMOTE_OS_AUTHENT parameter to TRUE in the database initialization parameter file forces the database to accept the client operating system user name received over an unsecure connection and use it for account access. Because clients, in general, such as PCs, are not trusted to perform operating system authentication properly, it is very poor security practice to turn on this feature.

The default setting, REMOTE_OS_AUTHENT = FALSE, creates a more secure configuration that enforces proper, server-based authentication of clients connecting to an Oracle database.

Be aware that the REMOTE_OS_AUTHENT parameter was deprecated in Oracle Database 11*g* Release 1 (11.1), and is retained only for backward compatibility.

Any change to this parameter takes effect the next time you start the instance and mount the database. Generally, user authentication through the host operating system offers faster and more convenient connection to Oracle Database without specifying a separate database user name or password. Also, user entries correspond in the database and operating system audit trails.

3.10.6 Authentication of User Logins Using Network Authentication

Oracle strong authentication performs network authentication, which you can configure to use a third-party service such as Kerberos.

If you are using Oracle strong authentication as your only external authentication service, then the REMOTE_OS_AUTHENT parameter setting is irrelevant, because Oracle strong authentication permits only secure connections.

3.11 Multitier Authentication and Authorization

Oracle Database secures middle-tier applications by limiting privileges, preserving client identities through all tiers, and auditing actions by clients.

In applications that use a very busy middle tier, such as a transaction processing monitor, the identity of the clients connecting to the middle tier must be preserved. One advantage of using a middle tier is **connection pooling**, which allows multiple users to access a data server without each of them needing a separate connection. In such environments, you need to be able to set up and break down connections very quickly.

For these environments, you can use the Oracle Call Interface to create **lightweight sessions**, which enable database password authentication for each user. This method preserves the identity of the real user through the middle tier without the overhead of a separate database connection for each user.

You can create lightweight sessions with or without passwords. However, if a middle tier is outside of or on a firewall, then security is better when each lightweight session has its own password. For an internal application server, lightweight sessions without passwords might be appropriate.



3.12 Administration and Security in Clients, Application Servers, and Database Servers

In a multitier environment, an application server provides data for clients and serves as an interface to one or more database servers.

The application server can validate the credentials of a client, such as a Web browser, and the database server can audit operations performed by the application server. These auditable operations include actions performed by the application server on behalf of clients, such as requests that information be displayed on the client. A request to connect to the database server is an example of an application server operation not related to a specific client.

Authentication in a multitier environment is based on trust regions. Client authentication is the domain of the application server. The application server itself is authenticated by the database server. The following operations take place:

- The end user provides proof of authenticity to the application server, typically, by using a
 password or an X.509 certificate.
- The application server authenticates the end user and then authenticates itself to the database server.
- The database server authenticates the application server, verifies that the end user exists, and verifies that the application server has the privilege to connect for the end user.

Application servers can also enable roles for an end user on whose behalf they connect. The application server can obtain these roles from a directory, which serves as an authorization repository. The application server can only request that these roles be enabled. The database verifies the following requirements:

- That the client has these roles by checking its internal role repository
- That the application server has the privilege to connect on behalf of the user and thus to use these roles as the user could

The following diagram shows an example of multitier authentication.



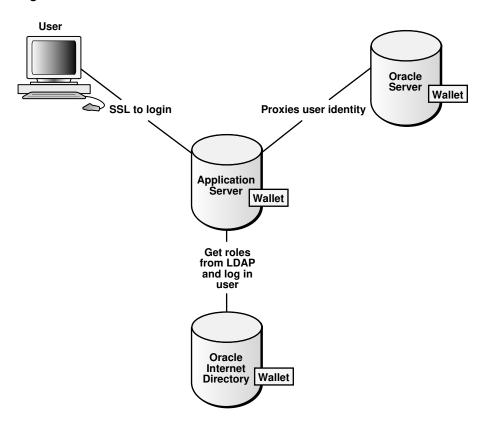


Figure 3-3 Multitier Authentication

The following actions take place:

- The user logs on using a password or Transport Layer Security. The authentication information is passed through Oracle Application Server.
- 2. Oracle Internet Directory authenticates the user, gets the roles associated with that user from the wallet, and then passes this information back to Oracle Application Server.
- 3. Oracle Application Server checks the identity of the user in Oracle Database, which contains a wallet that stores this information, and then sets the role for that user.

Security for middle-tier applications must address the following key issues:

- Accountability. The database server must be able to distinguish between the actions of the application and the actions an application takes on behalf of a client. It must be possible to audit both kinds of actions.
- Least privilege. Users and middle tiers should be given the fewest privileges necessary to perform their actions, to reduce the danger of inadvertent or malicious unauthorized activities.

3.13 Preserving User Identity in Multitiered Environments

You can use middle tier servers for proxy authentication and client identifiers to identify application users who are not known to the database.

Middle Tier Server Use for Proxy Authentication
 Oracle Call Interface (OCI), JDBC/OCI, or JDBC Thin Driver supports the middle tier for proxy authentication for database users or enterprise users.

Using Client Identifiers to Identify Application Users Unknown to the Database
 Client identifiers preserve user identity in middle tier systems; they also can be used
 independently of the global application context.

3.13.1 Middle Tier Server Use for Proxy Authentication

Oracle Call Interface (OCI), JDBC/OCI, or JDBC Thin Driver supports the middle tier for proxy authentication for database users or enterprise users.

- About Proxy Authentication
 Oracle Database provides proxy authentication in Oracle Call Interface (OCI), JDBC/OCI,
 or JDBC Thin Driver for database users or enterprise users.
- Advantages of Proxy Authentication
 In multitier environments, proxy authentication preserves client identities and privileges through all tiers in middle-tier applications and by auditing client actions.
- Who Can Create Proxy User Accounts?
 To create proxy user accounts, users must have special privileges.
- Guidelines for Creating Proxy User Accounts
 Oracle provides special guidelines for when you create proxy user accounts.
- Creating Proxy User Accounts and Authorizing Users to Connect Through Them
 The CREATE USER and ALTER USER statements can be used to create a proxy user and
 authorize users to connect through it.
- Proxy User Accounts and the Authorization of Users to Connect Through Them
 The CREATE USER statement enables you to create the several types of user accounts, all
 of which can be used as proxy accounts.
- Using Proxy Authentication with the Secure External Password Store
 Use a secure external password store if you are concerned about the password used in proxy authentication being obtained by a malicious user.
- How the Identity of the Real User Is Passed with Proxy Authentication
 You can use Oracle Call Interface, JDBC/OCI, or Thin drivers for enterprise users or
 database users.
- Limits to the Privileges of the Middle Tier
 Least privilege is the principle that users should have the fewest privileges necessary to perform their duties and no more.
- Authorizing a Middle Tier to Proxy and Authenticate a User You can authorize a middle-tier server to connect as a user.
- Authorizing a Middle Tier to Proxy a User Authenticated by Other Means
 You can authorize a middle tier to proxy a user that has been authenticated by other
 means.
- Reauthenticating a User Through the Middle Tier to the Database
 You can specify that authentication is required by using the AUTHENTICATION REQUIRED proxy clause with the ALTER USER SQL statement.
- Using Password-Based Proxy Authentication
 When you use password-based proxy authentication, Oracle Database passes the password of the client to the middle-tier server.
- Using Proxy Authentication with Enterprise Users
 How the middle-tier responds for proxy authentication depends on how the user is authenticated, either as an enterprise user or a password-authenticated user.

3.13.1.1 About Proxy Authentication

Oracle Database provides proxy authentication in Oracle Call Interface (OCI), JDBC/OCI, or JDBC Thin Driver for database users or enterprise users.

Enterprise users are those who are managed in Oracle Internet Directory and who access a shared schema in the database.

You can design a middle-tier server to authenticate clients in a secure fashion by using the following three forms of proxy authentication:

- The middle-tier server authenticates itself with the database server and a client, in this
 case an application user or another application, authenticates itself with the middle-tier
 server. Client identities can be maintained all the way through to the database.
- The client, in this case a database user, is not authenticated by the middle-tier server. The
 clients identity and database password are passed through the middle-tier server to the
 database server for authentication.
- The client, in this case a global user, is authenticated by the middle-tier server, and passes one of the following through the middle tier for retrieving the client's user name.
 - Distinguished name (DN)
 - Certificate

In all cases, an administrator must authorize the middle-tier server to act on behalf of the client.

Related Topics

- Auditing SQL Statements and Privileges in a Multitier Environment
 You can create a unified audit policy to audit the activities of a client in a multitier environment.
- Oracle Database JDBC Developer's Guide

3.13.1.2 Advantages of Proxy Authentication

In multitier environments, proxy authentication preserves client identities and privileges through all tiers in middle-tier applications and by auditing client actions.

For example, this feature allows the identity of a user using a Web application (which acts as a proxy) to be passed through the application to the database server.

Three-tier systems provide the following benefits to organizations:

- Organizations can separate application logic from data storage, partitioning the former in application servers and the latter in databases.
- Application servers and Web servers enable users to access data stored in databases.
- Users like using a familiar, easy-to-use browser interface.
- Organizations can also lower their cost of computing by replacing many thick clients with numerous thin clients and an application server.

In addition, Oracle Database proxy authentication provides the following security benefits:

- A limited trust model, by controlling the users on whose behalf middle tiers can connect and the roles that the middle tiers can assume for the user
- Scalability, by supporting user sessions through OCI, JDBC/OCI, or JDBC Thin driver and eliminating the overhead of reauthenticating clients



- Accountability, by preserving the identity of the real user through to the database, and enabling auditing of actions taken on behalf of the real user
- Flexibility, by supporting environments in which users are known to the database, and in which users are merely application users of which the database has no awareness



Oracle Database supports this proxy authentication functionality in three tiers only. It does not support it across multiple middle tiers.

3.13.1.3 Who Can Create Proxy User Accounts?

To create proxy user accounts, users must have special privileges.

These privileges are as follows:

- The CREATE USER system privilege to create a database user account that will be used as a proxy user account
- The DV ACCTMGR role if Oracle Database Vault is enabled, to create the proxy user account
- The ability to grant the CREATE SESSION system privilege to the proxy user account
- The ALTER USER system privilege to enable existing user accounts to connect to the database through the proxy account



In an Oracle Database Vault environment, when operations control is enabled, common users cannot proxy as local users in a PDB.

3.13.1.4 Guidelines for Creating Proxy User Accounts

Oracle provides special guidelines for when you create proxy user accounts.

- For better security and to adhere to the principle of least privilege, only grant the proxy user account the CREATE SESSION privilege. Do not grant this user any other privileges. The proxy user account is designed to only enable another user to connect using the proxy account. Any privileges that must be exercised during the connection should belong to the connecting user, not to the proxy account.
- As with all passwords, ensure that the password you create for the proxy user is strong and not easily guessed. Remember that multiple users will be connecting as the proxy user, so it is especially important that this password be strong.
- Consider using the Oracle strong authentication network connection features, to prevent network eavesdropping.
- For further fine-tuning of the amount of control that the connecting user has, consider restricting the roles used by the connecting user when he or she is connected through the proxy account. The ALTER USER statement WITH ROLE clause enables you to configure the user to connect using specified roles, any role except a specified role, or with no roles at all. Be aware that the proxy user can only activate those roles that are included in the WITH



ROLE clause. The proxy user session will have all the privileges that were directly granted to the client (that is, current) user.

• A proxy user in a proxy session can enable a password-protected role or secure application role only if the role has been allowed to be enabled with the WITH ROLE OF WITH ROLE ALL clause. (If this clause is not specified, then WITH ROLE ALL is the default.) If WITH ROLE does not specify the secure roles, then those roles cannot be enabled, even with the correct password.

Related Topics

Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.

3.13.1.5 Creating Proxy User Accounts and Authorizing Users to Connect Through Them

The CREATE USER and ALTER USER statements can be used to create a proxy user and authorize users to connect through it.

A proxy user in a proxy session can enable a password-protected role or secure application role only if the role has been allowed to be enabled with the WITH ROLE OF WITH ROLE ALL clause. (If this clause is not specified, then WITH ROLE ALL is the default.) If WITH ROLE does not specify the secure roles, then those roles cannot be enabled, even with the correct password.

1. Use the CREATE USER statement to create the proxy user account.

For example:

```
CREATE USER appuser IDENTIFIED BY password;
```

2. Use the GRANT CONNECT THROUGH clause of the ALTER USER statement to enable an existing user to connect through the proxy user account.

For example:

```
ALTER USER preston GRANT CONNECT THROUGH appuser;
```

Be aware that the user name and proxy combination must not exceed 250 characters.

Suppose user preston has a large number of roles, but you only want this user to use one role (for example, the appuser_role) when this user is connected to the database through the appuser proxy account. You can use the following ALTER USER statement:

```
ALTER USER preston GRANT CONNECT THROUGH appuser WITH ROLE appuser role;
```

Any other roles that user preston has will not be available to her as long as this user is connecting as the appuser proxy.

After you complete these steps, user preston can connect using the appuser proxy user as follows:

```
CONNECT appuser[preston]
Enter password: appuser_password
```

Related Topics

- Oracle Database SQL Language Reference
- Oracle Database SQL Language Reference



3.13.1.6 Proxy User Accounts and the Authorization of Users to Connect Through Them

The CREATE USER statement enables you to create the several types of user accounts, all of which can be used as proxy accounts.

These accounts are as follows:

- Database user accounts, which are authenticated by passwords
- External user accounts, which are authenticated by external sources, such as Secure Socket Layer (SSL) or Kerberos
- Global user accounts, which are authenticated by an enterprise directory service (Oracle Internet Directory).

Note the following:

- The proxy user can only perform activities that the user preston has privileges to perform. Remember that the proxy user itself, appuser, only has the minimum privileges (CREATE SESSION).
- **Using roles with middle-tier clients.** You can also specify roles that the middle tier is permitted to activate when connecting as the client. Operations performed on behalf of a client by a middle-tier server can be audited.
- **Finding proxy users.** To find the users who are currently authorized to connect through a middle tier, query the PROXY USERS data dictionary view, for example:

```
SELECT * FROM PROXY USERS;
```

• Removing proxy connections. Use the REVOKE CONNECT THROUGH clause of ALTER USER to disallow a proxy connection. For example, to revoke user preston from connecting through the proxy user appuser, enter the following statement:

```
ALTER USER preston REVOKE CONNECT THROUGH appuser;
```

Password expiration and proxy connections. Middle-tier use of password expiration
does not apply to accounts that are authenticated through a proxy. Instead, lock the
account rather than expire the password.

Related Topics

- Auditing SQL Statements and Privileges in a Multitier Environment
 You can create a unified audit policy to audit the activities of a client in a multitier environment.
- Oracle Database Enterprise User Security Administrator's Guide

3.13.1.7 Using Proxy Authentication with the Secure External Password Store

Use a secure external password store if you are concerned about the password used in proxy authentication being obtained by a malicious user.

To accomplish this, you use the secure external password store with the proxy authentication to store the password credentials in a wallet.

Connecting to Oracle Database using proxy authentication and the secure external password store is ideal for situations such as running batch files. When a proxy user connects to the database and authenticates using a secure external password, the password is not exposed in the event that a malicious user tries to obtain the password.



To use proxy authentication with the secure external password store:

- 1. Configure the proxy authentication account, as shown in the procedure in Proxy User Accounts and the Authorization of Users to Connect Through Them.
- 2. Configure the secure external password store, as described in About Configuring Clients to Use the Secure External Password Store.

Afterward, the user can connect using the proxy but without having to specify a password. For example:

```
sqlplus [preston]/@db alias
```

When you use the secure external password store, the user logging in does not need to supply the user name and password. Only the <code>SERVICE_NAME</code> value (that is, <code>db_alias</code>) from the <code>tnsnames.ora</code> file must be specified.

3.13.1.8 How the Identity of the Real User Is Passed with Proxy Authentication

You can use Oracle Call Interface, JDBC/OCI, or Thin drivers for enterprise users or database users.

These tools enable a middle tier to set up several user sessions within a single database connection, each of which uniquely identifies a connected user (connection pooling)

These sessions reduce the network overhead of creating separate network connections from the middle tier to the database.

If you want to authenticate from clients through a middle tier to the database, then the full authentication sequence from the client to the middle tier to the database occurs as follows:

- 1. The client authenticates to the middle tier, using whatever form of authentication the middle tier will accept. For example, the client could authenticate to the middle tier by using a user name and password or an X.509 certificate by means of SSL.
- 2. The middle tier authenticates itself to the database by using whatever form of authentication the database accepts. This could be a password or an authentication mechanism supported by Oracle Database, such as a Kerberos ticket or an X.509 certificate (SSL).
- 3. The middle tier then creates one or more sessions for users using OCI, JDBC/OCI, or Thin driver.
 - If the user is a database user, then the session must, as a minimum, include the database user name. If the database requires it, then the session can include a password (which the database verifies against the password store in the database). The session can also include a list of database roles for the user.
 - If the user is an enterprise user, then the session may provide different information depending on how the user is authenticated.

Example 1: If the user authenticates to the middle tier using SSL, then the middle tier can provide the DN from the X.509 certificate of the user, or the certificate itself in the session. The database uses the DN to look up the user in Oracle Internet Directory.

Example 2: If the user is a password-authenticated enterprise user, then the middle tier must provide, as a minimum, a globally unique name for the user. The database uses this name to look up the user in Oracle Internet Directory. If the session also provides a password for the user, then the database will verify the password against Oracle Internet Directory. User roles are automatically retrieved from Oracle Internet Directory after the session is established.



- The middle tier may optionally provide a list of database roles for the client. These roles are enabled if the proxy is authorized to use the roles on behalf of the client.
- The database verifies that the middle tier has the privilege to create sessions on behalf of the user.

The <code>OCISessionBegin</code> call fails if the application server cannot perform a proxy authentication on behalf of the client by the administrator, or if the application server is not allowed to activate the specified roles.

3.13.1.9 Limits to the Privileges of the Middle Tier

Least privilege is the principle that users should have the fewest privileges necessary to perform their duties and no more.

As applied to middle tier applications, this means that the middle tier should not have more privileges than it needs.

Oracle Database enables you to limit the middle tier such that it can connect only on behalf of certain database users, using only specific database roles. You can limit the privilege of the middle tier to connect on behalf of an enterprise user, stored in an LDAP directory, by granting to the middle tier the privilege to connect as the mapped database user. For instance, if the enterprise user is mapped to the APPUSER schema, then you must at least grant to the middle tier the ability to connect on behalf of APPUSER. Otherwise, attempts to create a session for the enterprise user will fail.

However, you cannot limit the ability of the middle tier to connect on behalf of enterprise users. For example, suppose that user Sarah wants to connect to the database through a middle tier, appsrv (which is also a database user). Sarah has multiple roles, but it is desirable to restrict the middle tier to use only the clerk role on her behalf.

An administrator can grant permission for appsrv to initiate connections on behalf of Sarah using her clerk role only by using the following SQL statement:

```
ALTER USER sarah GRANT CONNECT THROUGH appsrv WITH ROLE clerk;
```

By default, the middle tier cannot create connections for any client. The permission must be granted for each user.

To enable <code>appsrv</code> to use all of the roles granted to the client Sarah, you can use the following statement:

```
ALTER USER sarah GRANT CONNECT THROUGH appsrv;
```

Each time a middle tier initiates an OCI, JDBC/OCI, or Thin driver session for another database user, the database verifies that the middle tier is authorized to connect for that user by using the role specified.



Note:

Instead of using default roles, create your own roles and assign only necessary privileges to them. Creating your own roles enables you to control the privileges granted by them and protects you if Oracle Database changes or removes default roles. For example, the CONNECT role now has only the CREATE SESSION privilege, the one most directly needed when connecting to a database. However, CONNECT formerly provided several additional privileges, often not needed or appropriate for most users. Extra privileges can endanger the security of your database and applications. These have now been removed from CONNECT.

A proxy user in a proxy session can enable a password-protected role or secure application role only if the role has been allowed to be enabled with the WITH ROLE OR WITH ROLE ALL clause. (If this clause is not specified, then WITH ROLE ALL is the default.) If WITH ROLE does not specify the secure roles, then those roles cannot be enabled, even with the correct password.

Related Topics

Configuring Privilege and Role Authorization
 Privilege and role authorization controls the permissions that users have to perform day-to-day tasks.

3.13.1.10 Authorizing a Middle Tier to Proxy and Authenticate a User

You can authorize a middle-tier server to connect as a user.

A proxy user in a proxy session can enable a password-protected role or secure application role only if the role has been allowed to be enabled with the WITH ROLE OF WITH ROLE ALL clause. (If this clause is not specified, then WITH ROLE ALL is the default.) If WITH ROLE does not specify the secure roles, then those roles cannot be enabled, even with the correct password.

To authorize a middle-tier server to connect as a user, use the ALTER USER statement.

The following statement authorizes the middle-tier server appserve to connect as user bill. It uses the WITH ROLE clause to specify that appserve activate all roles associated with bill, except payroll.

```
ALTER USER bill
GRANT CONNECT THROUGH appserve
WITH ROLE ALL EXCEPT payroll;
```

To revoke the middle-tier server (appserve) authorization to connect as user bill, you can use the REVOKE CONNECT THROUGH clause. For example:

ALTER USER bill REVOKE CONNECT THROUGH appserve;

3.13.1.11 Authorizing a Middle Tier to Proxy a User Authenticated by Other Means

You can authorize a middle tier to proxy a user that has been authenticated by other means.

Currently, PASSWORD is the only means supported.

• Use the AUTHENTICATION REQURED clause of the ALTER USER ... GRANT CONNECT THROUGH statement to authorize a user to be proxied, but not authenticated, by a middle tier.

For example:

```
ALTER USER mary

GRANT CONNECT THROUGH midtier

AUTHENTICATION REQUIRED;
```

In the preceding statement, middle-tier server midtier is authorized to connect as user mary, and midtier must also pass the user password to the database server for authorization.

3.13.1.12 Reauthenticating a User Through the Middle Tier to the Database

You can specify that authentication is required by using the AUTHENTICATION REQUIRED proxy clause with the ALTER USER SQL statement.

In this case, the middle tier must provide user authentication credentials.

For example, suppose that user Sarah wants to connect to the database through a middle tier, appsrv.

 To require that appsrv provides authentication credentials for the user Sarah, use the following syntax:

```
ALTER USER sarah GRANT CONNECT THROUGH appsrv AUTHENTICATION REQUIRED;
```

The AUTHENTICATION REQUIRED clause ensures that authentication credentials for the user must be presented when the user is authenticated through the specified proxy.



For backward compatibility, if you use the AUTHENTICATED USING PASSWORD proxy clause, then Oracle Database transforms it to AUTHENTICATION REQUIRED.

3.13.1.13 Using Password-Based Proxy Authentication

When you use password-based proxy authentication, Oracle Database passes the password of the client to the middle-tier server.

The middle-tier server then passes the password as an attribute to the data server for verification.

The main advantage to this type of authentication is that the client computer does not have to have Oracle software installed on it to perform database operations.

• To pass the password of the client, configure the the middle-tier server to call the OCIAttrSet() function as follows, passing OCI_ATTR_PASSWORD as the type of the attribute being set.

3.13.1.14 Using Proxy Authentication with Enterprise Users

How the middle-tier responds for proxy authentication depends on how the user is authenticated, either as an enterprise user or a password-authenticated user.

If the middle tier connects to the database as a client who is an enterprise user, then either the distinguished name, or the X.509 certificate containing the distinguished name is passed over instead of the database user name. If the user is a password-authenticated enterprise user, then the middle tier must provide, as a minimum, a globally unique name for the user. The database uses this name to look up the user in Oracle Internet Directory.

- To configure proxy authentication with enterprise users, configure the application server and the middle tier to use the appropriate Oracle Call Interface settings:
 - To pass over the distinguished name of the client, configure the application server to call the Oracle Call Interface method OCIAttrSet() with

OCI ATTR DISTINGUISHED NAME as the attribute type, as follows:

To pass over the entire certificate, configure the middle tier to call OCIAttrSet() with
 OCI ATTR CERTIFICATE as the attribute type, as follows:

If the type is not specified, then the database uses its default certificate type of X.509.

Note:

- OCI_ATTR_CERTIFICATE is Distinguished Encoding Rules (DER) encoded.
- Certificate based proxy authentication using OCI_ATTR_CERTIFICATE will not be supported in future Oracle Database releases. Use the OCI ATTR DISTINGUISHED NAME or OCI ATTR USERNAME attribute instead

If you are using proxy authentication for password-authenticated enterprise users, then use the same OCI attributes as for database users authenticated by password (OCI_ATTR_USERNAME). Oracle Database first checks the user name against the database. If it finds no user, then the database checks the user name in the directory. This user name must be globally unique.

3.13.2 Using Client Identifiers to Identify Application Users Unknown to the Database

Client identifiers preserve user identity in middle tier systems; they also can be used independently of the global application context.

- About Client Identifiers
 - Oracle Database provides the CLIENT_IDENTIFIER attribute of the built-in USERENV application context namespace for application users.
- How Client Identifiers Work in Middle Tier Systems
 Many applications use session pooling to set up several sessions to be reused by multiple application users.
- Use of the CLIENT_IDENTIFIER Attribute to Preserve User Identity
 The CLIENT_IDENTIFIER predefined attribute of the built-in application context namespace,
 USERENV, captures the application user name for use with a global application context.
- Use of the CLIENT_IDENTIFIER Independent of Global Application Context
 Using the CLIENT_IDENTIFIER attribute is especially useful for those applications in which
 the users are unknown to the database.
- Setting the CLIENT_IDENTIFIER Independent of Global Application Context You can set the CLIENT_IDENTIFIER setting with Oracle Call Interface to be independent of the global application context.
- Use of the DBMS_SESSION PL/SQL Package to Set and Clear the Client Identifier
 The DBMS_SESSION PL/SQL package manages client identifiers on both the middle tier and
 the database itself.
- Enabling the CLIENTID_OVERWRITE Event System-Wide

 The ALTER SYSTEM statement can enable the CLIENTID_OVERWRITE event system-wide.
- Enabling the CLIENTID_OVERWRITE Event for the Current Session

 The ALTER SESSION statement can enable the CLIENTID_OVERWRITE event for the current session only.
- Disabling the CLIENTID_OVERWRITE Event
 The ALTER SYSTEM statement can disable the CLIENTID_OVERWRITE event.

3.13.2.1 About Client Identifiers

Oracle Database provides the CLIENT_IDENTIFIER attribute of the built-in USERENV application context namespace for application users.

These application users are known to an application but unknown to the database. The <code>CLIENT_IDENTIFIER</code> attribute can capture any value that the application uses for identification or access control, and passes it to the database. The <code>CLIENT_IDENTIFIER</code> attribute is supported in OCI, JDBC/OCI, or Thin driver.

3.13.2.2 How Client Identifiers Work in Middle Tier Systems

Many applications use session pooling to set up several sessions to be reused by multiple application users.

Users authenticate themselves to a middle-tier application, which uses a single identity to log in to the database and maintains all the user connections. In this model, application users are users who are authenticated to the middle tier of an application, but who are not known to the

database. You can use a CLIENT_IDENTIFIER attribute, which acts like an application user proxy for these types of applications.

In this model, the middle tier passes a client identifier to the database upon the session establishment. The client identifier could actually be anything that represents a client connecting to the middle tier, for example, a cookie or an IP address. The client identifier, representing the application user, is available in user session information and can also be accessed with an application context (by using the USERENV naming context). In this way, applications can set up and reuse sessions, while still being able to keep track of the application user in the session. Applications can reset the client identifier and thus reuse the session for a different user, enabling high performance.

3.13.2.3 Use of the CLIENT_IDENTIFIER Attribute to Preserve User Identity

The CLIENT_IDENTIFIER predefined attribute of the built-in application context namespace, USERENV, captures the application user name for use with a global application context.

You also can use the CLIENT IDENTIFIER attribute independently.

When you use the <code>CLIENT_IDENTIFIER</code> attribute independently from a global application context, you can set <code>CLIENT_IDENTIFIER</code> with the <code>DBMS_SESSION</code> interface. The ability to pass a <code>CLIENT_IDENTIFIER</code> to the database is supported in Oracle Call Interface (OCI), <code>JDBC/OCI</code>, or Thin driver.

When you use the <code>CLIENT_IDENTIFIER</code> attribute with global application context, it provides flexibility and high performance for building applications. For example, suppose a Web-based application that provides information to business partners has three types of users: gold partner, silver partner, and bronze partner, representing different levels of information available. Instead of each user having his or her own session set up with individual application contexts, the application could set up global application contexts for gold partners, silver partners, and bronze partners. Then, use the <code>CLIENT_IDENTIFIER</code> to point the session at the correct context to retrieve the appropriate type of data. The application need only initialize the three global contexts once and use the <code>CLIENT_IDENTIFIER</code> to access the correct application context to limit data access. This provides performance benefits through session reuse and through accessing global application contexts set up once, instead of having to initialize application contexts for each session individually.

Related Topics

- Global Application Contexts
 You can use a global application context to access application values across database sessions, including an Oracle Real Application Clusters environment.
- Tutorial: Creating a Global Application Context That Uses a Client Session ID
 This tutorial demonstrates how you can create a global application context that uses a client session ID.

3.13.2.4 Use of the CLIENT_IDENTIFIER Independent of Global Application Context

Using the <code>CLIENT_IDENTIFIER</code> attribute is especially useful for those applications in which the users are unknown to the database.

In these situations, the application typically connects as a single database user and all actions are taken as that user.

Because all user sessions are created as the same user, this security model makes it difficult to achieve data separation for each user. These applications can use the <code>CLIENT_IDENTIFIER</code> attribute to preserve the real application user identity through to the database.



With this approach, sessions can be reused by multiple users by changing the value of the <code>CLIENT_IDENTIFIER</code> attribute, which captures the name of the real application user. This avoids the overhead of setting up a separate session and separate attributes for each user, and enables reuse of sessions by the application. When the <code>CLIENT_IDENTIFIER</code> attribute value changes, the change is added to the next OCI, <code>JDBC/OCI</code>, or Thin driver call for additional performance benefits.

For example, the user Daniel connects to a Web Expense application. Daniel is not a database user; he is a typical Web Expense application user. The application accesses the built-in application context namespace and sets <code>DANIEL</code> as the <code>CLIENT_IDENTIFIER</code> attribute value. Daniel completes his Web Expense form and exits the application. Then, Ajit connects to the Web Expense application. Instead of setting up a new session for Ajit, the application reuses the session that currently exists for Daniel, by changing the <code>CLIENT_IDENTIFIER</code> to <code>AJIT</code>. This avoids the overhead of setting up a new connection to the database and the overhead of setting up a global application context. The <code>CLIENT_IDENTIFIER</code> attribute can be set to any value on which the application bases access control. It does not have to be the application user name.

3.13.2.5 Setting the CLIENT_IDENTIFIER Independent of Global Application Context

You can set the CLIENT_IDENTIFIER setting with Oracle Call Interface to be independent of the global application context.

• To set the CLIENT_IDENTIFIER attribute with OCI, use the OCI_ATTR_CLIENT_IDENTIFIER attribute in the call to OCIAttrSet(). Then, on the next request to the server, the information is propagated and stored in the server sessions.

For example:

```
OCI_HTYPE_SESSION,
(dvoid *) "appuser1",
(ub4) strlen("appuser1"),
OCI_ATTR_CLIENT_IDENTIFIER,
*error handle);
```

For applications that use JDBC, be aware that JDBC does not set the client identifier. To set the client identifier in a connection pooling environment, use Dynamic Monitoring Service (DMS) metrics. If DMS is not available, then use the <code>connection.setClientInfo</code> method. For example:

```
connection.setClientInfo("E2E_CONTEXT.CLIENT_IDENTIFIER", "appuser");
```

See Also:

- Oracle Call Interface Programmer's Guide about how the OCI_ATTR_CLIENT_IDENTIFIER user session handle attribute is used in middle-tier applications
- Oracle Database JDBC Developer's Guide for more information about configuring client connections using JDBC and DMS metrics
- Oracle Database JDBC Developer's Guide for more information about the setClientInfo method

3.13.2.6 Use of the DBMS_SESSION PL/SQL Package to Set and Clear the Client Identifier

The DBMS_SESSION PL/SQL package manages client identifiers on both the middle tier and the database itself.

To use the <code>DBMS_SESSION</code> package to set and clear the <code>CLIENT_IDENTIFIER</code> value on the middle tier, you must use the <code>SET IDENTIFIER</code> and <code>CLEAR IDENTIFIER</code> procedures.

The middle tier uses <code>SET_IDENTIFIER</code> to associate the database session with a particular user or group. Then, the <code>CLIENT_IDENTIFIER</code> is an attribute of the session and can be viewed in session information.

If you plan to use the <code>DBMS_SESSION.SET_IDENTIFIER</code> procedure, then be aware of the following:

- The maximum number of bytes for the client_id parameter of
 DBMS_SESSION.SET_IDENTIFIER is 64 bytes. If it exceeds 64, then the additional bytes are
 truncated.
- The DBMS_APPLICATION_INFO.SET_CLIENT_INFO procedure can overwrite the value of the client identifier. Typically, these values should be the same, so if SET_CLIENT_INFO is set, then its value can be automatically propagated to the value set by SET_IDENTIFIER if the CLIENTID_OVERWRITE event is set to ON. You can check the status of the CLIENTID_OVERWRITE event by running the SHOW PARAMETER command for the EVENT parameter.

For example, assuming that CLIENTID OVERWRITE is enabled:

SHOW PARAMETER EVENT

NAME	TYPE	VALUE
event	string	clientid overwrite

3.13.2.7 Enabling the CLIENTID OVERWRITE Event System-Wide

The ALTER SYSTEM statement can enable the CLIENTID OVERWRITE event system-wide.

1. Enter the following ALTER SYSTEM statement:

```
ALTER SYSTEM SET EVENTS 'CLIENTID OVERWRITE';
```

Or, enter the following line in your init.ora file:

```
event="clientid overwrite"
```

2. Restart the database.

For example:

SHUTDOWN IMMEDIATE STARTUP



See Also:

- Global Application Contexts for information about using client identifiers in a global application context
- Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS SESSION package

3.13.2.8 Enabling the CLIENTID OVERWRITE Event for the Current Session

The ALTER SESSION statement can enable the CLIENTID_OVERWRITE event for the current session only.

1. Use the ALTER SESSION statement to set the CLIENTID_OVERWRITE value for the session only.

For example:

```
ALTER SESSION SET EVENTS 'CLIENTID_OVERWRITE OFF';
```

2. If you set the client identifier by using the DBMS_APPLICATION_INFO.SET_CLIENT_INFO procedure, then run DBMS_SESSION.SET_IDENTIFIER so that the client identifier settings are the same.

For example:

```
DBMS SESSION.SET IDENTIFIER(session id p);
```

3.13.2.9 Disabling the CLIENTID OVERWRITE Event

The ALTER SYSTEM statement can disable the CLIENTID OVERWRITE event.

1. Enter the following ALTER SYSTEM statement:

```
ALTER SYSTEM SET EVENTS 'CLIENTID_OVERWRITE OFF';
```

2. Restart the database.

For example:

```
SHUTDOWN IMMEDIATE STARTUP
```

3.14 User Authentication Data Dictionary Views

Oracle Database provides data dictionary views that list information about user authentication, such as roles that users have or profiles they use.

Table 3-5 lists the data dictionary views.

Table 3-5 Data Dictionary Views That Describe User Authentication

View	Description	
DBA_PROFILES	Displays information about profiles, including their settings and limits	



Table 3-5 (Cont.) Data Dictionary Views That Describe User Authentication

View	Description	
DBA_ROLES	Displays the kind of authentication used for a database role to log in to the database, such as NONE or GLOBAL (query the AUTHENTICATION_TYPE column)	
DBA_USERS	Among other user information, displays the following:	
	 The kind of authentication the user used to log in to the database, such as PASSWORD or EXTERNAL (AUTHENTICATION_TYPE column) The list of versions of password versions (also known as hashes) 	
	that exist for the user account (PASSWORD_VERSIONS column)	
DBA_USERS_WITH_DEFPWD	Displays whether the user account password is a default password	
PROXY_USERS	Displays users who are currently authorized to connect through a middle tier	
V\$DBLINK	Displays user accounts for existing database links (DB_LINK, OWNER_ID columns); applies to the current pluggable database (PDB)	
V\$PWFILE	Lists the names and granted administrative privileges of the administrative users who are included in the password file	
V\$SESSION	Querying the ${\tt USERNAME}$ column displays concurrently logged in users to the current PDB	

Related Topics

Oracle Database Reference



Configuring Privilege and Role Authorization

Privilege and role authorization controls the permissions that users have to perform day-to-day tasks.

About Privileges and Roles

Authorization permits users to access, process, or alter data; it also creates limitations on user access or actions.

Who Should Be Granted Privileges?

You grant privileges to users so they can accomplish tasks required for their jobs.

How the Oracle Multitenant Option Affects Privileges

In a multitenant environment, all users, including common users, can exercise their privileges only within the current container.

Managing Administrative Privileges

Administrative privileges can be used for both general and specific database operations.

Managing System Privileges

To perform actions on schema objects, you must be granted the appropriate system privileges.

Managing Commonly and Locally Granted Privileges

In a multitenant environment, privileges can be granted commonly for an entire CDB or application container, or granted locally to a specific PDB.

Managing Common Roles and Local Roles

A common role is a role that is created in the root; a local role is created in a PDB.

Managing User Roles

A user role is a named collection of privileges that you can create and assign to other users.

Restricting Operations on PDBs Using PDB Lockdown Profiles

You can use PDB lockdown profiles in a multitenant environment to restrict sets of user operations in pluggable databases (PDBs).

Managing Object Privileges

Object privileges enable you to perform actions on schema objects, such as tables or indexes.

Table Privileges

Object privileges for tables enable table security at the DML or DDL level of operation.

View Privileges

You can apply DML object privileges to views, similar to tables.

Procedure Privileges

The EXECUTE privilege enables users to run procedures and functions, either standalone or in packages.

Type Privileges

You can control system and object privileges for types, methods, and objects.

Grants of User Privileges and Roles

The GRANT statement provides privileges for a user to perform specific actions, such as executing a procedure.

- Revokes of Privileges and Roles from a User
 - When you revoke system or object privileges, be aware of the cascading effects of revoking a privilege.
- Grants and Revokes of Privileges to and from the PUBLIC Role
 You can grant and revoke privileges and roles from the role PUBLIC.
- Grants of Roles Using the Operating System or Network
 Using the operating system or network to manage roles can help centralize the role
 management in a large enterprise.
- How Grants and Revokes Work with SET ROLE and Default Role Settings
 Privilege grants and the SET ROLE statement affect when and how grants and revokes take place.
- User Privilege and Role Data Dictionary Views
 You can use special queries to find information about various types of privilege and role
 grants.

4.1 About Privileges and Roles

Authorization permits users to access, process, or alter data; it also creates limitations on user access or actions.

The limitations placed on (or removed from) users can apply to objects such as schemas, entire tables, or table rows.

A user **privilege** is the right to run a particular type of SQL statement, or the right to access an object that belongs to another user, run a PL/SQL package, and so on. The types of privileges are defined by Oracle Database.

Roles are created by users (usually administrators) to group together privileges or other roles. They are a way to facilitate the granting of multiple privileges or roles to users.

Privileges can fall into the following general categories:

- Administrative privileges. Administrative privileges are designed for commonly
 performed administrative tasks, such as performing backup and recovery operations.
 Oracle Database provides administrative privileges tailored to specific administrative tasks,
 such as the SYSKM administrative privilege for performing Transparent Data Encryption
 tasks.
- **System privileges.** System privileges enable users to perform actions on schema objects. Examples of a system privilege are the ability to create and update tables or tablespaces.
- Roles. A role groups several privileges and roles, so that they can be granted to and
 revoked from users simultaneously. You must enable the role for a user before the user
 can use it.
- Object privileges. Each type of object has privileges associated with it. Objects are schema objects, such as tables or indexes
- Table privileges. These privileges enable security at the DML (data manipulation language) or DDL (data definition language) level. DML operations are DELETE, INSERT, SELECT, and UPDATE operations on tables. DDL operations are ALTER, INDEX, and REFERENCES operations on tables and views.
- View privileges. You can apply DML object privileges to views, similar to tables.
- **Procedure privileges.** Procedures, including standalone procedures and functions, can be granted the EXECUTE privilege.



Type privileges. You can grant system privileges to named types (object types, VARRAYS, and nested tables).

Related Topics

- Managing Administrative Privileges
 Administrative privileges can be used for both general and specific database operations.
- Managing System Privileges
 To perform actions on schema objects, you must be granted the appropriate system privileges.
- Managing Commonly and Locally Granted Privileges
 In a multitenant environment, privileges can be granted commonly for an entire CDB or application container, or granted locally to a specific PDB.

4.2 Who Should Be Granted Privileges?

You grant privileges to users so they can accomplish tasks required for their jobs.

You should grant a privilege only to a user who requires that privilege to accomplish the necessary work. Excessive granting of unnecessary privileges can compromise security. For example, you never should grant SYSDBA or SYSOPER administrative privilege to users who do not perform administrative tasks.

You can grant privileges to a user in two ways:

- You can grant privileges to users explicitly. For example, you can explicitly grant to user psmith the privilege to insert records into the employees table.
- You can grant privileges to a role (a named group of privileges), and then grant the
 role to one or more users. For example, you can grant the privileges to select, insert,
 update, and delete records from the employees table to the role named clerk, which in
 turn you can grant to users psmith and robert.

Because roles allow for easier and better management of privileges, you should usually grant privileges to roles and not to specific users.

See Also:

- Guidelines for Securing User Accounts and Privileges for best practices to follow when granting privileges
- Oracle Database Vault Administrator's Guide if you are concerned about excessive privilege grants
- Oracle Database SQL Language Reference for the complete list of system privileges and their descriptions

4.3 How the Oracle Multitenant Option Affects Privileges

In a multitenant environment, all users, including common users, can exercise their privileges only within the current container.

However, a user connected to the root can perform certain operations that affect other pluggable databases (PDBs). These operations include ALTER PLUGGABLE DATABASE, CREATE

USER, CREATE ROLE, and ALTER USER. The common user must possess the commonly granted privileges that enable these operations. A common user connected to the root can see metadata pertaining to PDBs by way of the container data objects (for example, multitenant container database (CDB) views and V\$ views) in the root, provided that the common user has been granted privileges required to access these views and his CONTAINER_DATA attribute has been set to allow seeing data about various PDBs. The common user cannot query tables or views in a PDB.

Common users cannot exercise their privileges across other PDBs. They must first switch to the PDB that they want, and then exercise their privileges from there. To switch to a different container, the common user must have the SET CONTAINER privilege. The SET CONTAINER privilege must be granted either commonly or in the container to which the user is attempting to switch. Alternatively, the common user can start a new database session whose initial current container is the container this user wants, relying on the CREATE SESSION privilege in that PDB.

Be aware that commonly granted privileges may interfere with the security configured for individual PDBs. For example, suppose an application PDB database administrator wants to prevent any user in the PDB from modifying a particular application common object. A privilege (such as UPDATE) granted commonly to PUBLIC or to a common user or common role on the object would circumvent the PDB database administrator's intent.

Related Topics

Enabling Common Users to View CONTAINER_DATA Object Information
 Common users can view information about CONTAINER_DATA objects in the root or for data in specific PDBs.

4.4 Managing Administrative Privileges

Administrative privileges can be used for both general and specific database operations.

- About Administrative Privileges
 For better separation of duty, Oracle Database provides administrative privileges that are tailored for commonly performed specific administrative tasks.
- Grants of Administrative Privileges to Users
 As with all powerful privileges, only grant administrative privileges to trusted users.
- SYSDBA and SYSOPER Privileges for Standard Database Operations
 The SYSDBA and SYSOPER administrative privileges enable you to perform standard database operations.
- Forcing oracle Users to Enter a Password When Logging in as SYSDBA
 You can force an oracle user to enter a password when the user logs in to an Oracle
 database using the SYSDBA administrative privilege.
- SYSBACKUP Administrative Privilege for Backup and Recovery Operations
 The SYSBACKUP administrative privilege is used to perform backup and recovery operations from either Oracle Recovery Manager (RMAN) and or through SQL*Plus.
- SYSDG Administrative Privilege for Oracle Data Guard Operations
 You can log in as user SYSDG with the SYSDG administrative privilege to perform Data Guard operations.
- SYSKM Administrative Privilege for Transparent Data Encryption
 The SYSKM administrative privilege enables the SYSKM user to manage Transparent Data Encryption (TDE) wallet operations.



SYSRAC Administrative Privilege for Oracle Real Application Clusters
 The SYSRAC administrative privilege is used by the Oracle Real Application Clusters (Oracle RAC) Clusterware agent.

4.4.1 About Administrative Privileges

For better separation of duty, Oracle Database provides administrative privileges that are tailored for commonly performed specific administrative tasks.

These tasks include operations for backup and recovery, Oracle Data Guard, and encryption key management for Transparent Data Encryption (TDE).

You can find the administrative privileges that a user has by querying the V\$PWFILE_USERS dynamic view, which lists users in the password file.

In previous releases, you needed to have the SYSDBA administrative privilege to perform these tasks. To support backward compatibility, you still can use the SYSDBA privilege for these tasks, but Oracle recommends that you use the administrative privileges described in this section.

Users who have been granted administrative privileges can be altered to be schema-only accounts.

The use of administrative privileges is mandatorily audited.

Related Topics

Auditing Administrative Users
 You can create unified audit policies to capture the actions of administrative user accounts, such as SYS.

4.4.2 Grants of Administrative Privileges to Users

As with all powerful privileges, only grant administrative privileges to trusted users.

However, be aware that there is a restriction for users whose names have non-ASCII characters (for example, the umlaut in the name HÜBER). You can grant administrative privileges to these users, but if the Oracle database instance is down, the authentication using the granted privilege is not supported if the user name has non-ASCII characters. If the database instance is up, then the authentication is supported.

4.4.3 SYSDBA and SYSOPER Privileges for Standard Database Operations

The SYSDBA and SYSOPER administrative privileges enable you to perform standard database operations.

These database operations can include tasks such as database startups and shutdowns, creating the server parameter file (SPFILE), or altering the database archive log. In a multitenant environment, you can grant the SYSDBA and SYSOPER administrative privileges to application common users (but not to CDB common users).

You can find if a user has been granted an administrative privilege on a local (PDB) level, for a CDB root, or for an application root by querying the SCOPE column of the V\$PWFILE_USERS dynamic view.

You can grant the SYSDBA or SYSOPER administrative privilege to users who have been created with no authentication.



4.4.4 Forcing oracle Users to Enter a Password When Logging in as SYSDBA

You can force an oracle user to enter a password when the user logs in to an Oracle database using the SYSDBA administrative privilege.

- 1. Edit the \$ORACLE HOME/network/admin/sqlnet.ora file.
- 2. Set the SQLNET.AUTHENTICATION SERVICES parameter as follows:

```
sqlnet.authentication services=none
```

If SQLNET.AUTHENTICATION SERVICES is not set, then it defaults to ALL.

4.4.5 SYSBACKUP Administrative Privilege for Backup and Recovery Operations

The SYSBACKUP administrative privilege is used to perform backup and recovery operations from either Oracle Recovery Manager (RMAN) and or through SQL*Plus.

To connect to the database as SYSBACKUP using a password, you must create a password file for it. See *Oracle Database Administrator's Guide* for more information about creating password files.

You cannot grant the SYSBACKUP administrative privilege to users who have been created with no authentication.

This privilege enables you to perform the following operations:

- STARTUP
- SHUTDOWN
- ALTER DATABASE
- ALTER SYSTEM
- ALTER SESSION
- ALTER TABLESPACE
- CREATE CONTROLFILE
- CREATE ANY DIRECTORY
- CREATE ANY TABLE
- CREATE ANY CLUSTER
- CREATE PFILE
- CREATE RESTORE POINT (including GUARANTEED restore points)
- CREATE SESSION
- CREATE SPFILE
- DROP DATABASE
- DROP TABLESPACE



- DROP RESTORE POINT (including GUARANTEED restore points)
- FLASHBACK DATABASE
- RESUMABLE
- UNLIMITED TABLESPACE
- SELECT ANY DICTIONARY
- SELECT ANY TRANSACTION
- SELECT
 - X\$ tables (that is, the fixed tables)
 - V\$ and GV\$ views (that is, the dynamic performance views)
 - APPQOSSYS.WLM CLASSIFIER PLAN
 - SYSTEM.LOGSTDBY\$PARAMETERS
- DELETE/INSERT
 - SYS.APPLY\$ SOURCE SCHEMA
 - SYSTEM.LOGSTDBY\$PARAMETERS
- EXECUTE
 - SYS.DBMS BACKUP RESTORE
 - SYS.DBMS_RCVMAN
 - SYS.DBMS DATAPUMP
 - SYS.DBMS IR
 - SYS.DBMS_PIPE
 - SYS.SYS_ERROR
 - SYS.DBMS TTS
 - SYS.DBMS TDB
 - SYS.DBMS_PLUGTS
 - SYS.DBMS PLUGTSP
- SELECT CATALOG ROLE

In addition, the SYSBACKUP privilege enables you to connect to the database even if the database is not open.

See Also:

Oracle Database Backup and Recovery User's Guide for more information about backup and recovery operations

4.4.6 SYSDG Administrative Privilege for Oracle Data Guard Operations

You can log in as user SYSDG with the SYSDG administrative privilege to perform Data Guard operations.

You can use this privilege with either Data Guard Broker or the DGMGRL command-line interface. In order to connect to the database as SYSDG using a password, you must create a password file for it.

You cannot grant the SYSYSDG administrative privilege to users who have been created with no authentication.

The SYSDG privilege enables the following operations:

- STARTUP
- SHUTDOWN
- ALTER DATABASE
- ALTER SESSION
- ALTER SYSTEM
- CREATE RESTORE POINT (including GUARANTEED restore points)
- CREATE SESSION
- DROP RESTORE POINT (including GUARANTEED restore points)
- FLASHBACK DATABASE
- SELECT ANY DICTIONARY
- SELECT
 - X\$ tables (that is, the fixed tables)
 - V\$ and GV\$ views (that is, the dynamic performance views)
 - APPQOSSYS.WLM CLASSIFIER PLAN
- DELETE
 - APPQOSSYS.WLM CLASSIFIER PLAN
- EXECUTE
 - SYS.DBMS_DRS

In addition, the SYSDG privilege enables you to connect to the database even if it is not open.

See Also:

- Oracle Database Administrator's Guide for more information about creating password files
- Oracle Data Guard Concepts and Administration for more information about Oracle Data Guard

4.4.7 SYSKM Administrative Privilege for Transparent Data Encryption

The SYSKM administrative privilege enables the SYSKM user to manage Transparent Data Encryption (TDE) wallet operations.



In order to connect to the database as SYSKM using a password, you must create a password file for it.

You cannot grant the SYSKM administrative privilege to users who have been created with no authentication.

The SYSKM administrative privilege enables the following operations:

- ADMINISTER KEY MANAGEMENT
- CREATE SESSION
- SELECT (only when database is open)
 - SYS.V\$ENCRYPTED_TABLESPACES
 - SYS.V\$ENCRYPTION WALLET
 - SYS.V\$WALLET
 - SYS.V\$ENCRYPTION KEYS
 - SYS.V\$CLIENT_SECRETS
 - SYS.DBA ENCRYPTION KEY USAGE

In addition, the SYSKM privilege enables you to connect to the database even if it is not open.

See Also:

- Oracle Database Administrator's Guide for more information about creating password files
- Oracle Database Advanced Security Guide for more information about Transparent Data Encryption

4.4.8 SYSRAC Administrative Privilege for Oracle Real Application Clusters

The SYSRAC administrative privilege is used by the Oracle Real Application Clusters (Oracle RAC) Clusterware agent.

The SYSRAC administrative privilege provides only the minimal privileges necessary for performing day-to-day Oracle RAC operations. For example, this privilege is used for Oracle RAC utilities such as SRVCTL.

You cannot grant the SYSRAC administrative privilege to users who have been created with no authentication.

The SYSRAC administrative privilege enables the following operations:

- STARTUP
- SHUTDOWN
- ALTER DATABASE MOUNT
- ALTER DATABASE OPEN
- ALTER DATABASE OPEN READ ONLY



- ALTER DATABASE CLOSE NORMAL
- ALTER DATABASE DISMOUNT
- ALTER SESSION SET EVENTS
- ALTER SESSION SET _NOTIFY_CRS
- ALTER SESSION SET CONTAINER
- ALTER SYSTEM REGISTER
- ALTER SYSTEM SET local listener|remote listener|listener networks

In addition to these privileges, the SYSRAC user will have access to the following views:

- V\$PARAMETER
- V\$DATABASE
- V\$PDBS
- CDB SERVICE\$
- DBA SERVICES
- V\$ACTIVE SERVICES
- V\$SERVICES

The SYSRAC user is also granted the EXECUTE privilege for the following PL/SQL packages:

- DBMS DRS
- DBMS SERVICE
- DBMS SERVICE PRVT
- DBMS SESSION
- DBMS HA ALERTS PRVT
- Dequeue messaging SYS.SYS\$SERVICE METRICS

Related Topics

Oracle Real Application Clusters Administration and Deployment Guide

4.5 Managing System Privileges

To perform actions on schema objects, you must be granted the appropriate system privileges.

- About System Privileges
 - A system privilege is the right to perform an action or to perform actions on schema objects.
- Why Is It Important to Restrict System Privileges?
 - System privileges are very powerful, so only grant them to trusted users. You should also secure the data dictionary and SYS schema objects.
- · Grants and Revokes of System Privileges
 - You can grant or revoke system privileges to users and roles.
- Who Can Grant or Revoke System Privileges?
 - Only two types of users can grant system privileges to other users or revoke those privileges from them.



About ANY Privileges and the PUBLIC Role
 System privileges that use the ANY keyword enable you to set privileges for an entire category of objects in the database.

4.5.1 About System Privileges

A system privilege is the right to perform an action or to perform actions on schema objects.

For example, the privileges to create tablespaces and to delete the rows of any table in a database are system privileges.

There are over 100 distinct system privileges. Each system privilege allows a user to perform a particular database operation or class of database operations. *Remember that system privileges are very powerful.* Only grant them when necessary to roles and trusted users of the database. To find the system privileges that have been granted to a user, you can query the DBA SYS PRIVS data dictionary view.

System privileges such as <code>SELECT</code> ANY <code>TABLE</code> do not work on <code>SYS</code> objects or other objects that are protected by the <code>SELECT</code> ANY <code>DICTIONARY</code> privilege.

Related Topics

- How Commonly Granted System Privileges Work
 Users can exercise system privileges only within the PDB in which they were granted.
- Oracle Database SQL Language ReferenceGRANT

4.5.2 Why Is It Important to Restrict System Privileges?

System privileges are very powerful, so only grant them to trusted users. You should also secure the data dictionary and SYS schema objects.

- About the Importance of Restricting System Privileges
 System privileges are very powerful, so by default the database is configured to prevent typical (non-administrative) users from exercising the ANY system privileges.
- User Access to Objects in the SYS Schema
 Users with explicit object privileges or those who connect with administrative privileges
 (SYSDBA) can access objects in the SYS schema.

4.5.2.1 About the Importance of Restricting System Privileges

System privileges are very powerful, so by default the database is configured to prevent typical (non-administrative) users from exercising the ANY system privileges.

For example, users are prevented from exercising any system privileges such as $\tt UPDATE \ ANY \ TABLE \ on the data dictionary.$

Related Topics

Guidelines for Securing User Accounts and Privileges
 Oracle provides guidelines to secure user accounts and privileges.

4.5.2.2 User Access to Objects in the SYS Schema

Users with explicit object privileges or those who connect with administrative privileges (SYSDBA) can access objects in the SYS schema.



The following table lists roles that you can grant to users who need access to objects in the SYS schema.

Table 4-1 Roles to Allow Access to SYS Schema Objects

Role	Description
SELECT_CATALOG_ROLE	Grant this role to allow users SELECT privileges on data dictionary views.
EXECUTE_CATALOG_ROLE	Grant this role to allow users EXECUTE privileges for packages and procedures in the data dictionary.

Additionally, you can grant the SELECT ANY DICTIONARY system privilege to users who require access to tables created in the SYS schema. This system privilege allows query access to any object in the SYS schema, including tables created in that schema. It must be granted individually to each user requiring the privilege. It is not included in GRANT ALL PRIVILEGES, but it can be granted through a role.



You should grant these roles and the SELECT ANY DICTIONARY system privilege with extreme care, because the integrity of your system can be compromised by their misuse.

4.5.3 Grants and Revokes of System Privileges

You can grant or revoke system privileges to users and roles.

If you grant system privileges to roles, then you can use the roles to exercise system privileges. For example, roles permit privileges to be made selectively available. Ensure that you follow the separation of duty guidelines described in Guidelines for Securing Roles.

Use either of the following methods to grant or revoke system privileges to or from users and roles:

- GRANT and REVOKE SQL statements
- Oracle Enterprise Manager Cloud Control

Related Topics

User Privilege and Role Data Dictionary Views
 You can use special queries to find information about various types of privilege and role
 grants.

4.5.4 Who Can Grant or Revoke System Privileges?

Only two types of users can grant system privileges to other users or revoke those privileges from them.

These users are as follows:

- Users who were granted a specific system privilege with the ADMIN OPTION
- Users with the system privilege GRANT ANY PRIVILEGE

For this reason, only grant these privileges to trusted users.

4.5.5 About ANY Privileges and the PUBLIC Role

System privileges that use the ANY keyword enable you to set privileges for an entire category of objects in the database.

For example, the CREATE ANY PROCEDURE system privilege permits a user to create a procedure anywhere in the database. The behavior of an object created by users with the ANY privilege is not restricted to the schema in which it was created. For example, if user JSMITH has the CREATE ANY PROCEDURE privilege and creates a procedure in the schema JONES, then the procedure will run as JONES. However, JONES may not be aware that the procedure JSMITH created is running as him (JONES). If JONES has DBA privileges, letting JSMITH run a procedure as JONES could pose a security violation.

The PUBLIC role is a special role that every database user account automatically has when the account is created. By default, it has no privileges granted to it, but it does have numerous grants, mostly to Java objects. You cannot drop the PUBLIC role, and a manual grant or revoke of this role has no meaning, because the user account will always assume this role. Because all database user accounts assume the PUBLIC role, it does not appear in the DBA_ROLES and SESSION ROLES data dictionary views.

You can grant privileges to the PUBLIC role, but remember that this makes the privileges available to every user in the Oracle database. For this reason, be careful about granting privileges to the PUBLIC role, particularly powerful privileges such as the ANY privileges and system privileges. For example, if JSMITH has the CREATE PUBLIC SYNONYM system privilege, he could redefine an interface that he knows everyone else uses, and then point to it with the PUBLIC SYNONYM that he created. Instead of accessing the correct interface, users would access the interface of JSMITH, which could possibly perform illegal activities such as stealing the login credentials of users.

These types of privileges are very powerful and could pose a security risk if given to the wrong person. Be careful about granting privileges using ANY or PUBLIC. As with all privileges, you should follow the principles of "least privilege" when granting these privileges to users.

Related Topics

Guidelines for Securing a Database Installation and Configuration
 Oracle provides guidelines to secure the database installation and configuration.

4.6 Managing Commonly and Locally Granted Privileges

In a multitenant environment, privileges can be granted commonly for an entire CDB or application container, or granted locally to a specific PDB.

- About Commonly and Locally Granted Privileges
 In a multitenant environment, both common users and local users can grant privileges to one another.
- How Commonly Granted System Privileges Work
 Users can exercise system privileges only within the PDB in which they were granted.
- How Commonly Granted Object Privileges Work
 Object privileges on common objects applies to the object as well as all associated links on
 this common object.
- Granting or Revoking Privileges to Access a PDB
 You can grant and revoke privileges for PDB access in a multitenant environment.

- Example: Granting a Privilege in a Multitenant Environment
 You can use the GRANT statement to grant privileges in a multitenant environment.
- Enabling Common Users to View CONTAINER_DATA Object Information
 Common users can view information about CONTAINER_DATA objects in the root or for data
 in specific PDBs.

4.6.1 About Commonly and Locally Granted Privileges

In a multitenant environment, both common users and local users can grant privileges to one another.

Privileges by themselves are neither common nor local. How the privileges are applied depends on whether the privilege is granted commonly or granted locally.

For commonly granted privileges:

- A privilege that is granted commonly can be used in every existing and future container.
- Only common users can grant privileges commonly, and only if the grantee is common.
- A common user can grant privileges to another common user or to a common role.
- The grantor must be connected to the root and must specify CONTAINER=ALL in the GRANT statement.
- Both system and object privileges can be commonly granted. (Object privileges become actual only with regard to the specified object.)
- When a common user connects to or switches to a given container, this user's ability to
 perform various activities (such as creating a table) is controlled by privileges granted
 commonly as well as privileges granted locally in the given container.
- Do not grant privileges to PUBLIC commonly.

For locally granted privileges:

- A privilege granted locally can be used only in the container in which it was granted. When
 the privilege is granted in the root, it applies only to the root.
- Both common users and local users can grant privileges locally.
- A common user and a local user can grant privileges to other common or local roles.
- The grantor must be connected to the container and must specify CONTAINER=CURRENT in the GRANT statement.
- Any user can grant a privilege locally to any other user or role (both common and local) or to the PUBLIC role.

Related Topics

- Oracle Multitenant Administrator's Guide
- How the PUBLIC Role Works in a Multitenant Environment
 All privileges that Oracle grants to the PUBLIC role are granted locally.

4.6.2 How Commonly Granted System Privileges Work

Users can exercise system privileges only within the PDB in which they were granted.

For example, if a system privilege is locally granted to a common user A in a PDB B, user A can exercise that privilege only while connected to PDB B.



System privileges can apply in the root and in all existing and future PDBs if the following requirements are met:

- The system privilege grantor is a common user and the grantee is a common user, a common role, or the PUBLIC role. Do not commonly grant system privileges to the PUBLIC role, because this in effect makes the system privilege available to all users.
- The system privilege grantor possesses the ADMIN OPTION for the commonly granted privilege
- The GRANT statement must contain the CONTAINER=ALL clause.

The following example shows how to commonly grant a privilege to the common user c#hr admin.

```
CONNECT SYSTEM
Enter password: password
Connected.

GRANT CREATE ANY TABLE TO c##hr admin CONTAINER=ALL;
```

4.6.3 How Commonly Granted Object Privileges Work

Object privileges on common objects applies to the object as well as all associated links on this common object.

These links include all metadata links, data links (previously called object links), or extended data links that are associated with it in the root and in all PDBs belonging to the container (including future PDBs) if certain requirements are met.

These requirements are as follows:

- The object privilege grantor is a common user and the grantee is a common user, a common role, or the PUBLIC role.
- The object privilege grantor possesses the commonly granted GRANT OPTION for the privilege
- The GRANT statement contains the CONTAINER=ALL clause.

The following example shows how to grant an object privilege to the common user $c\#hr_admin$ so that he can select from the DBA_PDBS view in the CDB root or in any of the associated PDBs that he can access.

```
CONNECT SYSTEM
Enter password: password
Connected.

GRANT SELECT ON DBA_OBJECTS TO c##hr_admin
CONTAINER=ALL;
```

Related Topics

- Oracle Multitenant Administrator's Guide
- How the PUBLIC Role Works in a Multitenant Environment
 All privileges that Oracle grants to the PUBLIC role are granted locally.



4.6.4 Granting or Revoking Privileges to Access a PDB

You can grant and revoke privileges for PDB access in a multitenant environment.

To grant a privilege in a multitenant environment:

Include the CONTAINER clause in the GRANT or REVOKE statement.

Setting CONTAINER to ALL applies the privilege to all existing and future containers; setting it to CURRENT applies the privilege to the local container only. Omitting the CONTAINER clause applies the privilege to the local container. If you issue the GRANT statement from the root and omit the CONTAINER clause, then the privilege is applied locally.

Related Topics

Oracle Database SQL Language Reference

4.6.5 Example: Granting a Privilege in a Multitenant Environment

You can use the GRANT statement to grant privileges in a multitenant environment.

Example 4-1 shows how to commonly grant the CREATE TABLE privilege to common user c##hr admin so that this user can use this privilege in all existing and future containers.

Example 4-1 Granting a Privilege in a Multitenant Environment

CONNECT SYSTEM
Enter password: password
Connected.

GRANT CREATE TABLE TO c##hr admin CONTAINER=ALL;

4.6.6 Enabling Common Users to View CONTAINER_DATA Object Information

Common users can view information about CONTAINER_DATA objects in the root or for data in specific PDBs.

- Viewing Data About the Root, CDB, and PDBs While Connected to the Root
 You can restrict view information for the X\$ table and the V\$, GV\$ and CDB_* views when
 common users perform queries.
- Enabling Common Users to Query Data in Specific PDBs
 You can enable common users to access data pertaining to specific PDBs by adjusting the users' CONTAINER_DATA attribute.

4.6.6.1 Viewing Data About the Root, CDB, and PDBs While Connected to the Root

You can restrict view information for the X\$ table and the V\$, GV\$ and CDB_* views when common users perform queries.

The X\$ table and these views contain information about the application root and its associated application PDBs or, if you are connected to the CDB root, the entire CDB.

Restricting this information is useful when you do not want to expose sensitive information about other PDBs. To enable this functionality, Oracle Database provides these tables and views as container data objects. You can find if a specific table or view is a container data object by querying the TABLE_NAME, VIEW_NAME, and CONTAINER_DATA columns of the USER_| DBA_|ALL_VIEWS|TABLES dictionary views.

To find information about the default (user-level) and object-specific CONTAINER_DATA attributes:

- In SQL*Plus or SQL Developer, log in to the root.
- 2. Query the CDB CONTAINER DATA data dictionary view.

For example:

```
COLUMN USERNAME FORMAT A15
COLUMN DEFAULT ATTR FORMAT A7
COLUMN OWNER FORMAT A15
COLUMN OBJECT NAME FORMAT A15
COLUMN ALL CONTAINERS FORMAT A3
COLUMN CONTAINER NAME FORMAT A10
COLUMN CON ID FORMAT A6
SELECT USERNAME, DEFAULT ATTR, OWNER, OBJECT NAME,
     ALL CONTAINERS, CONTAINER NAME, CON ID
FROM CDB CONTAINER DATA
ORDER BY OBJECT NAME;
USERNAME DEFAULT OWNER OBJECT_NAME ALL CONTAINERS
C##HR_ADMIN N SYS V$SESSION
CDB$ROOT 1
C##HR_ADMIN N SYS V$SESSION
SALESPDB 1
                                 V$SESSION N
C##HR_ADMIN Y
                                                Ν
HRPDB
C##HR_ADMIN
CDB$ROOT
            1
DBSNMP
Y
                 1
SYSTEM
                 1
```

Related Topics

Oracle Database Reference

4.6.6.2 Enabling Common Users to Query Data in Specific PDBs

You can enable common users to access data pertaining to specific PDBs by adjusting the users' CONTAINER DATA attribute.

To enable common users to access data about specific PDBs:

Issue the ALTER USER statement in the root.

Example 4-2 Setting the CONTAINER_DATA Attribute

This example shows how to issue the ALTER USER statement to enable the common user c##hr_admin to view information pertaining to the CDB\$ROOT, SALES_PDB, and HRPDB containers in the V\$SESSION view (assuming this user can query that view).

```
CONNECT SYSTEM
Enter password: password
Connected.

ALTER USER c##hr_admin
SET CONTAINER_DATA = (CDB$ROOT, SALESPDB, HRPDB)
FOR V$SESSION CONTAINER=CURRENT;
```

In this specification:

- SET CONTAINER_DATA lists containers, data pertaining to which can be accessed by the user.
- FOR V\$SESSION specifies the CONTAINER_DATA dynamic view, which common user c##hr admin will query.
- CONTAINER = CURRENT must be specified because when you are connected to the root, CONTAINER=ALL is the default for the ALTER USER statement, but modification of the CONTAINER_DATA attribute must be restricted to the root.

If you want to enable user c##hr_admin to view information that pertains to the CDB\$ROOT, SALES_PDB, HRPDB containers in all CONTAINER_DATA objects that this user can access, then omit FOR V\$SESSION. For example:

```
ALTER USER c##hr_admin
SET CONTAINER_DATA = (CDB$ROOT, SALESPDB, HRPDB)
CONTAINER=CURRENT;
```

Related Topics

Oracle Database SQL Language Reference

4.7 Managing Common Roles and Local Roles

A common role is a role that is created in the root; a local role is created in a PDB.

- About Common Roles and Local Roles
 In a multitenant environment, database roles can be specific to a PDB or used throughout the entire system container or application container.
- How Common Roles Work

Common roles are visible in the root and in every PDB of a container within which they are defined in a multitenant environment.

- How the PUBLIC Role Works in a Multitenant Environment
 All privileges that Oracle grants to the PUBLIC role are granted locally.
- Privileges Required to Create, Modify, or Drop a Common Role
 Only common users who have the commonly granted CREATE ROLE, ALTER ROLE, and DROP
 ROLE privileges can create, alter, or drop common roles.

Rules for Creating Common Roles

When you create a common role, you must follow special rules.

Creating a Common Role

You can use the CREATE ROLE statement to create a common role.

Rules for Creating Local Roles

To create a local role, you must follow special rules.

Creating a Local Role

You can use the CREATE ROLE statement to create a role.

Role Grants and Revokes for Common Users and Local Users
 Role grants and revokes apply only to the scope of access of the common user or the local user.

4.7.1 About Common Roles and Local Roles

In a multitenant environment, database roles can be specific to a PDB or used throughout the entire system container or application container.

A common role is a role whose identity and (optional) password are created in the root of a container and will be known in the root and in all existing and future PDBs belonging to that container.

A local role exists in only one PDB and can only be used within this PDB. It does not have any commonly granted privileges.

Note the following:

- Common users can both create and grant common roles to other common and local users.
- You can grant a role (local or common) to a local user or role only locally.
- If you grant a common role locally, then the privileges of that common role apply only in the container where the role is granted.
- Local users cannot create common roles, but they can grant them to common and other local users.
- The CONTAINER = ALL clause is the default when you create a common role in the CDB root or an application root.

Related Topics

Predefined Roles in an Oracle Database Installation
 Oracle Database provides a set of predefined roles to help in database administration.

4.7.2 How Common Roles Work

Common roles are visible in the root and in every PDB of a container within which they are defined in a multitenant environment.

A privilege can be granted commonly to a common role if:

- The grantor is a common user.
- The grantor possesses the commonly granted ADMIN OPTION for the privilege that is being granted.
- The GRANT statement contains the CONTAINER=ALL clause.



If the common role contains locally granted privileges, then these privileges apply only within the PDB in which they were granted to the common role. A local role cannot be granted commonly.

For example, suppose the CDB common user $c\#hr_mgr$ has been commonly granted the DBA role. This means that user $c\#hr_mgr$ can use the privileges associated with the DBA role in the root and in every PDB in the multitenant environment. However, if the CDB common user $c\#hr_mgr$ has only been locally granted the DBA role for the hr_pdb PDB, then this user can only use the DBA role's privileges in the hr_pdb PDB.

4.7.3 How the PUBLIC Role Works in a Multitenant Environment

All privileges that Oracle grants to the PUBLIC role are granted locally.

This feature enables you to revoke privileges or roles that have been granted to the PUBLIC role individually in each PDB as needed. If you must grant any privileges to the PUBLIC role, then grant them locally. Never grant privileges to PUBLIC commonly.

Related Topics

About Commonly and Locally Granted Privileges
 In a multitenant environment, both common users and local users can grant privileges to one another.

4.7.4 Privileges Required to Create, Modify, or Drop a Common Role

Only common users who have the commonly granted CREATE ROLE, ALTER ROLE, and DROP ROLE privileges can create, alter, or drop common roles.

Common users can also create local roles, but these roles are available only in the PDB in which they were created.

4.7.5 Rules for Creating Common Roles

When you create a common role, you must follow special rules.

The rules are as follows:

- Ensure that you are in the correct root. For the creation of common roles, you must be in the correct root, either the CDB root or the application root. You cannot create common roles from a PDB. To check if you are in the correct root, run one of the following:
 - To confirm that you are in the CDB root, you can issue the show_con_name command.
 The output should be CDB\$ROOT.
 - To confirm that you are in an application root, verify that the following query returns YES:

```
SELECT APPLICATION_ROOT FROM V$PDBS WHERE CON_ID=SYS_CONTEXT('USERENV', 'CON_ID');
```

 Ensure that the name that you give the common role starts with the value of the COMMON_USER_PREFIX parameter (which defaults to C##). Note that this requirement does not apply to the names of existing Oracle-supplied roles, such as DBA or RESOURCE.



Optionally, set the CONTAINER clause to ALL. As long as you are in the root, if you omit
the CONTAINER = ALL clause, then by default the role is created as a common role for the
CDB root or the application root.

4.7.6 Creating a Common Role

You can use the CREATE ROLE statement to create a common role.

 Connect to the root of the CDB or the application container in which you want to create the common role.

For example:

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

2. Run the CREATE ROLE statement with the CONTAINER clause set to ALL.

For example:

```
CREATE ROLE c##sec admin IDENTIFIED BY password CONTAINER=ALL;
```

Related Topics

Creating a Role

You can create a role that is authenticated with or without a password. You also can create external or global roles.

Creating a Common Role in Enterprise Manager
 Common roles can be used to assign common privileges to common users.

4.7.7 Rules for Creating Local Roles

To create a local role, you must follow special rules.

These rules are as follows:

- You must be connected to the PDB in which you want to create the role, and have the CREATE ROLE privilege.
- The name that you give the local role must not start with the value of the COMMON USER PREFIX parameter (which defaults to C##).
- You can include CONTAINER=CURRENT in the CREATE ROLE statement to specify the role as a
 local role. If you are connected to a PDB and omit this clause, then the CONTAINER=CURRENT
 clause is implied.
- You cannot have common roles and local roles with the same name. However, you can
 use the same name for local roles in different PDBs. To find the names of existing roles,
 query the CDB ROLES and DBA ROLES data dictionary views.

4.7.8 Creating a Local Role

You can use the CREATE ROLE statement to create a role.

1. Connect to the PDB in which you want to create the local role.



For example:

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

2. Run the CREATE ROLE statement with the CONTAINER clause set to CURRENT.

For example:

```
CREATE ROLE sec admin CONTAINER=CURRENT;
```

4.7.9 Role Grants and Revokes for Common Users and Local Users

Role grants and revokes apply only to the scope of access of the common user or the local user.

Common users can grant and revoke common roles to and from other common users. A local user can grant a common role to any user in a PDB, including common users, but this grant applies only within the PDB.

The following example shows how to grant the common user c##sec_admin the AUDIT_ADMIN common role for use in all containers.

```
CONNECT SYSTEM
Enter password: password
Connected.

GRANT AUDIT_ADMIN TO c##sec_admin CONTAINER=ALL;
```

Similarly, the next example shows how local user <code>aud_admin</code> can grant the common user <code>c##sec admin</code> the <code>AUDIT ADMIN</code> common role for use within the <code>hrpdb PDB</code>.

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.

GRANT AUDIT_ADMIN TO c##sec_admin CONTAINER=CURRENT;
```

This example shows how a local user aud_admin can revoke a role from another user in a PDB. If you omit the CONTAINER clause, then CURRENT is implied.

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.

REVOKE sec admin FROM psmith CONTAINER=CURRENT;
```

Related Topics

 Revoking Common Privilege Grants in Enterprise Manager You can revoke common privilege grants from the root.

4.8 Managing User Roles

A user role is a named collection of privileges that you can create and assign to other users.

About User Roles

User roles are useful in a variety of situations, such as restricting DDL usage.

Predefined Roles in an Oracle Database Installation

Oracle Database provides a set of predefined roles to help in database administration.

Creating a Role

You can create a role that is authenticated with or without a password. You also can create external or global roles.

Specifying the Type of Role Authorization

You can configure a role to be authorized through different sources, such the database or an external source.

Granting and Revoking Roles

You can grant or revoke privileges to and from roles, and then grant these roles to users or to other roles.

Dropping Roles

Dropping a role affects the security domains of users or roles who had been granted the role.

Restricting SQL*Plus Users from Using Database Roles

You should restrict SQL*Plus users from using database roles, which helps to safeguard the database from intruder attacks.

Role Privileges and Secure Application Roles

A secure application role can be enabled only by an authorized PL/SQL package or procedure.

4.8.1 About User Roles

User roles are useful in a variety of situations, such as restricting DDL usage.

· What Are User Roles?

A user **role** is a named group of related privileges that you can grant as a group to users or other roles.

The Functionality of Roles

Roles are useful for quickly and easily granting permissions to users.

Properties of Roles and Why They Are Advantageous

Roles have special properties that make their management very easy, such reduced privilege administration.

Typical Uses of Roles

In general, you create a role to manage privileges.

Common Uses of Application Roles

You can use application roles to control privileges to use applications.

Common Uses of User Roles

You can create a user role for a group of database users with common privilege grant requirements.

· How Roles Affect the Scope of a User's Privileges

Each role and user has its own unique security domain.



How Roles Work in PL/SQL Blocks

Role behavior in a PL/SQL block is determined by the type of block and by definer's rights or invoker's rights.

How Roles Aid or Restrict DDL Usage

A user requires one or more privileges to successfully execute a DDL statement, depending on the statement.

How Operating Systems Can Aid Roles

In some environments, you can administer database security using the operating system.

How Roles Work in a Distributed Environment
 In a distributed database environment, all necessary roles must be set as the default role
 for a distributed (remote) session.

4.8.1.1 What Are User Roles?

A user **role** is a named group of related privileges that you can grant as a group to users or other roles.

Managing and controlling privileges is easier when you use roles.

Within a database, each role name must be unique, different from all user names and all other role names. Unlike schema objects, roles are not contained in any schema. Therefore, a user who creates a role can be dropped with no effect on the role.

Related Topics

Managing Common Roles and Local Roles
 A common role is a role that is created in the root; a local role is created in a PDB.



Managing Common Roles and Local Roles

4.8.1.2 The Functionality of Roles

Roles are useful for quickly and easily granting permissions to users.

Although you can use Oracle Database-defined roles, you have more control and continuity if you create your own roles that contain only the privileges pertaining to your requirements. Oracle may change or remove the privileges in an Oracle Database-defined role.

Roles have the following functionality:

- A role can be granted system or object privileges.
- Any role can be granted to any database user.
- Each role granted to a user is, at a given time, either enabled or disabled. A user's security
 domain includes the privileges of all roles currently enabled for the user and excludes the
 privileges of any roles currently disabled for the user. Oracle Database allows database
 applications and users to enable and disable roles to provide selective availability of
 privileges.
- A role can be granted to other roles. However, a role cannot be granted to itself and cannot be granted circularly. For example, role role1 cannot be granted to role role2 if role role2 has previously been granted to role role1.

- If a role is not password authenticated or a secure application role, then you can grant the role indirectly to the user. An indirectly granted role is a role granted to the user through another role that has already been granted to this user. For example, suppose you grant user psmith the role1 role. Then you grant the role2 and role3 roles to the role1 role. Roles role2 and role3 are now under role1. This means psmith has been indirectly granted the roles role2 and role3, in addition to the direct grant of role1. Enabling the direct role1 for psmith enables the indirect roles role2 and role3 for this user as well.
- Optionally, you can make a directly granted role a default role. You enable or disable the default role status of a directly granted role by using the DEFAULT ROLE clause of the ALTER USER statement. Ensure that the DEFAULT ROLE clause refers only to roles that have been directly granted to the user. To find the directly granted roles for a user, query the DBA_ROLE_PRIVS data dictionary view. This view does not include the user's indirectly granted roles. To find roles that are granted to other roles, query the ROLE_ROLE_PRIVS view.
- If the role is password authenticated or a secure application role, then you cannot grant it
 indirectly to the user, nor can you make it a default role. You only can grant this type of role
 directly to the user. Typically, you enable password authenticated or secure application
 roles by using the SET ROLE statement.

4.8.1.3 Properties of Roles and Why They Are Advantageous

Roles have special properties that make their management very easy, such reduced privilege administration.

Table 4-2 describes the properties of roles that enable easier privilege management within a database.

Table 4-2 Properties of Roles and Their Description

Property	Description
Reduced privilege administration	Rather than granting the same set of privileges explicitly to several users, you can grant the privileges for a group of related users to a role, and then only the role must be granted to each member of the group.
Dynamic privilege management	If the privileges of a group must change, then only the privileges of the role need to be modified. The security domains of all users granted the group's role automatically reflect the changes made to the role.
Selective availability of privileges	You can selectively enable or disable the roles granted to a user. This allows specific control of a user's privileges in any given situation.
Application awareness	The data dictionary records which roles exist, so you can design applications to query the dictionary and automatically enable (or disable) selective roles when a user attempts to execute the application by way of a given user name.
Application-specific security	You can protect role use with a password. Applications can be created specifically to enable a role when supplied the correct password. Users cannot enable the role if they do not know the password.

Database administrators often create roles for a database application. You should grant a secure application role all privileges necessary to run the application. You then can grant the secure application role to other roles or users. An application can have several different roles, each granted a different set of privileges that allow for more or less data access while using the application.



The DBA can create a role with a password to prevent unauthorized use of the privileges granted to the role. Typically, an application is designed so that when it starts, it enables the proper role. As a result, an application user does not need to know the password for an application role.

Related Topics

How Roles Aid or Restrict DDL Usage
 A user requires one or more privileges to successfully execute a DDL statement, depending on the statement.

4.8.1.4 Typical Uses of Roles

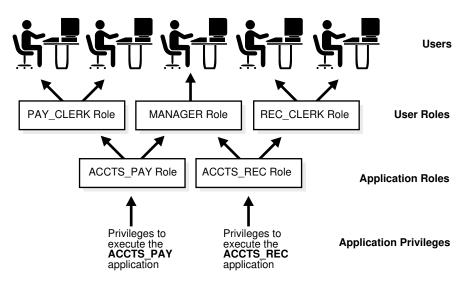
In general, you create a role to manage privileges.

Reasons are as follows:

- To manage the privileges for a database application
- To manage the privileges for a user group

The following diagram describes the two uses of roles.

Figure 4-1 Common Uses for Roles



Related Topics

- Common Uses of Application Roles
 You can use application roles to control privileges to use applications.
- Common Uses of User Roles
 You can create a user role for a group of database users with common privilege grant requirements.

4.8.1.5 Common Uses of Application Roles

You can use application roles to control privileges to use applications.

You should grant an application role all privileges necessary to run a given database application. Then, grant the secure application role to other roles or to specific users.

An application can have several different roles, with each role assigned a different set of privileges that allow for more or less data access while using the application.

4.8.1.6 Common Uses of User Roles

You can create a user role for a group of database users with common privilege grant requirements.

You can manage user privileges by granting secure application roles and privileges to the user role and then granting the user role to appropriate users.

4.8.1.7 How Roles Affect the Scope of a User's Privileges

Each role and user has its own unique security domain.

The security domain of a role includes the privileges granted to the role plus those privileges granted to any roles that are granted to the role.

The security domain of a user includes privileges on all schema objects in the corresponding schema, the privileges granted to the user, and the privileges of roles granted to the user that are **currently enabled**. (A role can be simultaneously enabled for one user and disabled for another.) This domain also includes the privileges and roles granted to the role PUBLIC. The PUBLIC role represents all users in the database.

4.8.1.8 How Roles Work in PL/SQL Blocks

Role behavior in a PL/SQL block is determined by the type of block and by definer's rights or invoker's rights.

- Roles Used in Named Blocks with Definer's Rights
 All roles are disabled in any named PL/SQL block that executes with definer's rights.
- Roles Used in Named Blocks with Invoker's Rights and Anonymous PL/SQL Blocks Named PL/SQL blocks that execute with invoker's rights and anonymous PL/SQL blocks are executed based on privileges granted through enabled roles.

4.8.1.8.1 Roles Used in Named Blocks with Definer's Rights

All roles are disabled in any named PL/SQL block that executes with definer's rights.

Examples of named PL/SQL blocks are stored procedures, functions, and triggers.

Roles are not used for privilege checking and you cannot set roles within a definer's rights procedure.

The SESSION_ROLES data dictionary view shows all roles that are currently enabled and if a PL/SQL block executes with definer's rights. If a named PL/SQL block that executes with definer's rights queries SESSION_ROLES, then the query does not return any rows.



Oracle Database Reference for more information about the ${\tt SESSION_ROLES}$ data dictionary view



4.8.1.8.2 Roles Used in Named Blocks with Invoker's Rights and Anonymous PL/SQL Blocks

Named PL/SQL blocks that execute with invoker's rights and anonymous PL/SQL blocks are executed based on privileges granted through enabled roles.

Current roles are used for privilege checking within an invoker's rights PL/SQL block. You can use dynamic SQL to set a role in the session.

See Also:

- Oracle Database PL/SQL Packages and Types Reference for an explanation of how invoker's and definer's rights can be used for name resolution and privilege checking
- Oracle Database PL/SQL Packages and Types Reference for information about dynamic SQL in PL/SQL

4.8.1.9 How Roles Aid or Restrict DDL Usage

A user requires one or more privileges to successfully execute a DDL statement, depending on the statement.

For example, to create a table, the user must have the CREATE TABLE or CREATE ANY TABLE system privilege.

To create a view of a table that belongs to another user, the creator must have the CREATE VIEW or CREATE ANY VIEW system privilege and either the SELECT *object* privilege for the table or the SELECT ANY TABLE system privilege.

Oracle Database avoids the dependencies on privileges received by way of roles by restricting the use of specific privileges in certain DDL statements. The following rules describe these privilege restrictions concerning DDL statements:

- All system privileges and object privileges that permit a user to perform a DDL operation are usable when received through a role. For example:
 - System privileges: CREATE TABLE, CREATE VIEW, and CREATE PROCEDURE privileges
 - Object privileges: ALTER and INDEX privileges for a table

You cannot use the REFERENCES object privilege for a table to define the foreign key of a table if the privilege is received through a role.

• All system privileges and object privileges that allow a user to perform a DML operation that is required to issue a DDL statement are *not* usable when received through a role. The security domain does not contain roles when a CREATE VIEW statement is used. For example, a user who is granted the SELECT ANY TABLE system privilege or the SELECT object privilege for a table through a role cannot use either of these privileges to create a view on a table that belongs to another user. This is because views are definer's rights objects, so when creating them you cannot use any privileges (neither system privileges or object privileges) granted to you through a role. If the privilege is granted directly to you, then you can use the privilege. However, if the privilege is revoked at a later time, then the view definition becomes invalid ("contains errors") and must recompiled before it can be used again.



The following example further clarifies the permitted and restricted uses of privileges received through roles.

Assume that a user is:

- Granted a role that has the CREATE VIEW system privilege
- Directly granted a role that has the SELECT object privilege for the employees table
- Directly granted the SELECT object privilege for the departments table

Given these directly and indirectly granted privileges:

- The user can issue SELECT statements on both the employees and departments tables.
- Although the user has both the CREATE VIEW and SELECT privilege for the employees table through a role, the user cannot create a view on the employees table, because the SELECT object privilege for the employees table was granted through a role.
- The user can create a view on the departments table, because the user has the
 CREATE VIEW privilege through a role and the SELECT privilege for the departments table
 directly.

4.8.1.10 How Operating Systems Can Aid Roles

In some environments, you can administer database security using the operating system.

The operating system can be used to grant and revoke database roles and to manage their password authentication. This capability is not available on all operating systems.



Your operating system-specific Oracle Database documentation for details about managing roles through the operating system

4.8.1.11 How Roles Work in a Distributed Environment

In a distributed database environment, all necessary roles must be set as the default role for a distributed (remote) session.

These roles cannot be enabled when the user connects to a remote database from within a local database session. For example, the user cannot execute a remote procedure that attempts to enable a role at the remote site.



Oracle Database Heterogeneous Connectivity User's Guide

4.8.2 Predefined Roles in an Oracle Database Installation

Oracle Database provides a set of predefined roles to help in database administration.



These predefined role are automatically defined for Oracle databases when you run the standard scripts (such as <code>catalog.sql</code> and <code>catproc.sql</code>) that are part of database creation, and they are considered common roles. If you install other options or products, then other predefined roles may be created. You can find roles that are created and maintained by Oracle by querying the <code>ROLE</code> and <code>ORACLE_MAINTAINED</code> columns of the <code>DBA_ROLES</code> data dictionary view. If the output for <code>ORACLE_MAINTAINED</code> is <code>Y</code>, then you must not modify the role except by running the script that was used to create it.

Table 4-3 Oracle Database Predefined Roles

Predefined Role	Description
ACCHK_READ	Provides privileges to use Application Continuity Protection Check (ACCHK), which includes the ability to query the following data dictionary views:
	• DBA_ACCHK_EVENTS
	DBA_ACCHK_EVENTS_SUMMARY
	• DBA_ACCHK_STATISTICS
	• DBA_ACCHK_STATISTICS_SUMMARY
	Database administrators and PDB administrators grant this role to developers to read their results from ACCHK.
ADM_PARALLEL_EXECUTE_TASK	Provides privileges to update table data in parallel by using the DBMS_PARALLEL_EXECUTE PL/SQL package.
AQ_ADMINISTRATOR_ROLE	Provides privileges to administer Advanced Queuing. Includes ENQUEUE ANY QUEUE, DEQUEUE ANY QUEUE, and MANAGE ANY QUEUE, SELECT privileges on Advanced Queuing tables and EXECUTE privileges on Advanced Queuing packages.
AQ_USER_ROLE	De-supported, but kept mainly for release 8.0 compatibility. Provides EXECUTE privileges on the DBMS_AQ and DBMS_AQIN packages.
AUDIT_ADMIN	Provides privileges to create unified and fine-grained audit policies, use the AUDIT and NOAUDIT SQL statements, view audit data, and manage the audit trail administration
AUDIT_VIEWER	Provides privileges to view and analyze audit data
AUTHENTICATEDUSER	Used by the XDB protocols to define any user who has logged in to the system.
CAPTURE_ADMIN	Provides the privileges necessary to create and manage privilege analysis policies.
CDB_DBA	Provides the privileges required for administering a CDB, such as SET CONTAINER, SELECT ON PDB_PLUG_IN_VIOLATIONS, and SELECT ON CDB_LOCAL_ADMIN_PRIVS. If your site requires additional privileges, then you car create a role (either common or local) to cover these privileges, and then grant this role to the CDB_DBA role.
CONNECT	Provides the CREATE SESSION system privilege.
	This role is provided for compatibility with previous releases of Oracle Database. You can determine the privileges encompassed by this role by querying the DBA SYS PRIVS data dictionary view.
	Note: Oracle recommends that you design your own roles for database security rather than relying on this role. This role may not be created automatically by future releases of Oracle Database.
CSW_USR_ROLE	Provides user privileges to manage the Catalog Services for the Web (CSW) component of Oracle Spatial.
CTXAPP	Provides privileges to create Oracle Text indexes and index preferences, and to use PL/SQL packages. This role should be granted to Oracle Text users.



Table 4-3 (Cont.) Oracle Database Predefined Roles

Predefined Role	Description
CWM_USER	Provides privileges to manage Common Warehouse Metadata (CWM), which is a repository standard used by Oracle data warehousing and decision support.
DATAPUMP_EXP_FULL_DATABASE	Provides privileges to export data from an Oracle database using Oracle Data Pump.
	Caution: This is a very powerful role because it provides a user access to any data in any schema in the database. Use caution when granting this role to users.
DATAPUMP_IMP_FULL_DATABASE	Provides privileges to import data into an Oracle database using Oracle Data Pump.
	Caution: This is a very powerful role because it provides a user access to any data in any schema in the database. Use caution when granting this role to users.
DBA	Provides a large number of system privileges, including the ANY privileges (such as the DELETE ANY TABLE and GRANT ANY PRIVILEGE privileges).
	This role is provided for compatibility with previous releases of Oracle Database. You can find the privileges that are encompassed by this role by querying the DBA_SYS_PRIVS data dictionary view.
	Note: Oracle recommends that you design your own roles for database security rather than relying on this role. This role may not be created automatically by future releases of Oracle Database.
DBFS_ROLE	Provides access to the DBFS (the Database Filesystem) packages and objects.
EJBCLIENT	Provides privileges to connect to EJBs from a Java stored procedure.
EM_EXPRESS_ALL	Enables users to connect to Oracle Enterprise Manager (EM) Express and use all the functionality provided by EM Express (read and write access to all EM Express features). The EM_EXPRESS_ALL role includes the EM_EXPRESS_BASIC role.
EM_EXRESS_BASIC	Enables users to connect to EM Express and to view the pages in read-only mode. The EM_EXPRESS_BASIC role includes the SELECT_CATALOG_ROLE role.
EXECUTE_CATALOG_ROLE	Provides EXECUTE privileges on objects in the data dictionary.
EXP_FULL_DATABASE	Provides the privileges required to perform full and incremental database exports using the Export utility (later replaced with Oracle Data Pump). It includes these privileges: SELECT ANY TABLE, BACKUP ANY TABLE, EXECUTE ANY PROCEDURE, EXECUTE ANY TYPE, ADMINISTER RESOURCE MANAGER, and INSERT, DELETE, and UPDATE on the tables SYS.INCVID, SYS.INCFIL, and SYS.INCEXP. Also includes the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
	This role is provided for convenience in using the export and import utilities.
	Caution: This is a very powerful role because it provides a user access to any data in any schema in the database. Use caution when granting this role to users.
GATHER_SYSTEM_STATISTICS	Provides privileges to update system statistics, which are collected using the DBMS_STATS.GATHER_SYSTEM_STATISTICS procedure
GLOBAL_AQ_USER_ROLE	Provides privileges to establish a connection to an LDAP server, for use with Oracle Database Advanced Queuing.
HS_ADMIN_EXECUTE_ROLE	Provides the EXECUTE privilege for users who want to use the Heterogeneous Services (HS) PL/SQL packages.
HS_ADMIN_ROLE	Provides privileges to both use the Heterogeneous Services (HS) PL/SQL packages and query the HS-related data dictionary views.
HS_ADMIN_SELECT_ROLE	Provides privileges to query the Heterogeneous Services data dictionary views.



Table 4-3 (Cont.) Oracle Database Predefined Roles

Predefined Role	Description
IMP_FULL_DATABASE	Provides the privileges required to perform full database imports using the Import utility (later replaced with Oracle Data Pump). Includes an extensive list of system privileges (use view DBA_SYS_PRIVS to view privileges) and the following roles: EXECUTE_CATALOG_ROLE and SELECT_CATALOG_ROLE.
	This role is provided for convenience in using the export and import utilities.
	Caution: This is a very powerful role because it provides a user access to any data in any schema in the database. Use caution when granting this role to users.
JAVADEBUGPRIV	Provides privileges to run the Oracle Database Java applications debugger.
JAVAIDPRIV	Deprecated for this release.
JAVASYSPRIV	Provides major permissions to use Java2, including updating Oracle JVM-protected packages.
JAVAUSERPRIV	Provides limited permissions to use Java2.
JAVA_ADMIN	Provides administrative permissions to update policy tables for Oracle Database Java applications.
JMXSERVER	Provides privileges to start and maintain a JMX agent in a database session.
LBAC_DBA	Provides permissions to use the SA_SYSDBA PL/SQL package.
LOGSTDBY_ADMINISTRATOR	Provides administrative privileges to manage the SQL Apply (logical standby database) environment.
OEM_ADVISOR	Provides privileges to create, drop, select (read), load (write), and delete a SQL tuning set through the DBMS_SQLTUNE PL/SQL package, and to access to the Advisor framework using the ADVISOR PL/SQL package.
OEM_MONITOR	Provides privileges needed by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database.
OLAP_DBA	Provides administrative privileges to create dimensional objects in different schemas for Oracle OLAP.
OLAP_USER	Provides application developers privileges to create dimensional objects in their own schemas for Oracle OLAP.
OLAP_XS_ADMIN	Provides privileges to administer security for Oracle OLAP.
OPTIMIZER_PROCESSING_RATE	Provides privileges to execute the GATHER_PROCESSING_RATE, SET_PROCESSING_RATE, and DELETE_PROCESSING_RATE procedures in the DBMS_STATS package. These procedures manage the processing rate of a system for automatic degree of parallelism (Auto DOP). Auto DOP uses these processing rates to determine the optimal degree of parallelism for a SQL statement.
PDB_DBA	Granted automatically to the local user that is created when you create a new PDB from the seed PDB. No privileges are provided with this role.
PROVISIONER	Provides privileges to register and update global callbacks for Oracle Database Real Application sessions and to provision principals.
RECOVERY_CATALOG_OWNER	Provides privileges for owner of the recovery catalog. Includes: CREATE SESSION, ALTER SESSION, CREATE SYNONYM, CREATE ANY SYNONYM, DROP ANY SYNONYM, CREATE VIEW, CREATE DATABASE LINK, CREATE TABLE, CREATE CLUSTER, CREATE SEQUENCE, CREATE TRIGGER, CREATE ANY TRIGGER, QUERY REWRITE, CREATE ANY CONTEXT, EXECUTE ON DBMS_RLS, ADMINISTER DATABASE, and CREATE PROCEDURE



Table 4-3 (Cont.) Oracle Database Predefined Roles

Predefined Role	Description
RESOURCE	Provides the following system privileges: CREATE CLUSTER, CREATE INDEXTYPE, CREATE OPERATOR, CREATE PROCEDURE, CREATE SEQUENCE, CREATE TABLE, CREATE TRIGGER, CREATE TYPE.
	Be aware that RESOURCE no longer provides the UNLIMITED TABLESPACE system privilege.
	This role is provided for compatibility with previous releases of Oracle Database. You can determine the privileges encompassed by this role by querying the DBA_SYS_PRIVS data dictionary view.
	Note: Oracle recommends that you design your own roles for database security rather than relying on this role. This role may not be created automatically by future releases of Oracle Database.
SCHEDULER_ADMIN	Allows the grantee to execute the procedures of the <code>DBMS_SCHEDULER</code> package. It includes all of the job scheduler system privileges and is included in the <code>DBA</code> role.
SELECT_CATALOG_ROLE	Provides SELECT privilege on objects in the data dictionary.
SODA_APP	Provides privileges to use the SODA APIs, in particular, to create, drop, and list document collections.
WM_ADMIN_ROLE	Provides administrative privileges for Oracle Workspace Manager. This enables users to run any <code>DBMS_WM</code> procedures on all version enabled tables, workspaces, and savepoints regardless of their owner. It also enables the user to modify the system parameters specific to Workspace Manager.
XDBADMIN	Allows the grantee to register an XML schema globally, as opposed to registering it for use or access only by its owner. It also lets the grantee bypass access control list (ACL) checks when accessing Oracle XML DB Repository.
XDB_SET_INVOKER	Allows the grantee to define invoker's rights handlers and to create or update the resource configuration for XML repository triggers. By default, Oracle Database grants this role to the DBA role but not to the XDBADMIN role.
XDB_WEBSERVICES	Allows the grantee to access Oracle Database Web services over HTTPS. However, it does not provide the user access to objects in the database that are public. To allow public access, you need to grant the user the XDB_WEBSERVICES_WITH_PUBLIC role. For a user to use these Web services, SYS must enable the Web service servlets.
XDB_WEBSERVICES_OVER_HTTP	Allows the grantee to access Oracle Database Web services over HTTP. However, it does not provide the user access to objects in the database that are public. To allow public access, you need to grant the user the XDB_WEBSERVICES_WITH_PUBLIC role.
XDB_WEBSERVICES_WITH_PUBLIC	Allows the grantee access to public objects through Oracle Database Web services.
XS_CACHE_ADMIN	In Oracle Database Real Application Security, enables the grantee to manage the mid-tier cache. It is required for caching the security policy at the mid-tier level for the <code>checkAcl</code> (authorization) method of the <code>XSAccessController</code> class. Grant this role to the application connection user or the Real Application Security dispatcher.
XS_NSATTR_ADMIN	In Oracle Database Real Application Security, enables the grantee to manage and manipulate the namespace and attribute for a session. Grant this role to the Real Application Security session user.



Table 4-3 (Cont.) Oracle Database Predefined Roles

Predefined Role	Description
XS_RESOURCE	In Oracle Database Real Application Security, enables the grantee to manage objects in the attached schema, through the XS_ACL PL/SQL package. This package creates procedures to create and manage access control lists (ACLs). It contains the ADMIN SEC POLICY privilege. It is similar to the Oracle Database RESOURCE role.
XS_SESSION_ADMIN	In Oracle Database Real Application Security, enables the grantee to manage the life cycle of a session, including the ability to create, attach, detach, and destroy the session. Grant this role to the application connection user or Real Application Security dispatcher.



Each installation should create its own roles and assign only those privileges that are needed, thus retaining detailed control of the privileges in use. This process also removes any need to adjust existing roles, privileges, or procedures whenever Oracle Database changes or removes roles that Oracle Database defines. For example, the CONNECT role now has only one privilege: CREATE SESSION.

4.8.3 Creating a Role

You can create a role that is authenticated with or without a password. You also can create external or global roles.

- About the Creation of Roles
 - You can create a role by using the CREATE ROLE statement.
- Creating a Role That Is Authenticated With a Password
 You can create a password authenticated role by using the IDENTIFIED BY clause.
- Creating a Role That Has No Password Authentication
 You can create a role that does not require a password by omitting the IDENTIFIED BY
 clause.
- Creating a Role That Is External or Global
 External or global roles allow services that are outside the database to associate database roles to authenticated users.
- Altering a Role

The ALTER ROLE statement can modify the authorization method for a role.

4.8.3.1 About the Creation of Roles

You can create a role by using the CREATE ROLE statement.

To create the role, you must have the CREATE ROLE system privilege. Typically, only security administrators have this system privilege. After you create a role, the role has no privileges associated with it. Your next step is to grant either privileges or other roles to the new role.

You must give each role that you create a unique name among existing user names and role names of the database. Roles are not contained in the schema of any user. In a database that uses a multi-byte character set, Oracle recommends that each role name contain at least one

single-byte character. If a role name contains only multi-byte characters, then the encrypted role name and password combination is considerably less secure. See Guideline 1 in Guidelines for Securing Passwords for password guidelines.

You can use the <code>IDENTIFIED</code> BY clause to authorize the role with a password. This clause specifies how the user must be authorized before the role can be enabled for use by a specific user to which it has been granted. If you do not specify this clause, or if you specify <code>NOTIDENTIFIED</code>, then no authorization is required when the role is enabled. Roles can be specified to be authorized by the following:

- The database using a password
- An application using a specified package
- Externally by the operating system, network, or other external source
- Globally by an enterprise directory service

As an alternative to creating password-protected roles, Oracle recommends that you use secure application roles instead.

Note the following restrictions about the creation of roles:

- A role and a user cannot have the same name.
- The role name cannot start with the value of the COMMON_USER_PREFIX parameter (which defaults to C##) unless this role is a CDB common role.

Related Topics

- Role Privileges and Secure Application Roles
 A secure application role can be enabled only by an authorized PL/SQL package or procedure.
- Creating Secure Application Roles to Control Access to Applications
 A secure application role is only enabled through its associated PL/SQL package or procedure.
- Rules for Creating Common Roles
 When you create a common role, you must follow special rules.

4.8.3.2 Creating a Role That Is Authenticated With a Password

You can create a password authenticated role by using the IDENTIFIED BY clause.

• To create a password-authenticated role, use the CREATE ROLE statement with the IDENTIFIED BY clause.

For example:

CREATE ROLE clerk IDENTIFIED BY password;



Note:

- You can enable password-protected roles in a proxy session. Both secure application roles and password-protected roles provide a secure method for enabling a role in a session. Oracle recommends using secure password roles instead of password-protected roles where the password has to be maintained and transmitted over insecure channels or if more than one person needs to know the password. Password-protected roles in a proxy session are suitable for situations where automation is used to set the role.
- If you set the SQLNET.ALLOWED_LOGON_VERSION_SERVER parameter is to 11 or higher, then you must recreate roles that have been created with the IDENTIFIED BY clause.

Related Topics

- Role Privileges and Secure Application Roles
 A secure application role can be enabled only by an authorized PL/SQL package or procedure.
- Management of Case Sensitivity for Secure Role Passwords
 For better security, you should ensure that the passwords for secure roles are case sensitive.

4.8.3.3 Creating a Role That Has No Password Authentication

You can create a role that does not require a password by omitting the IDENTIFIED BY clause.

 Use the CREATE ROLE statement with no clauses to create a role that has no password authentication.

For example:

CREATE ROLE salesclerk:

4.8.3.4 Creating a Role That Is External or Global

External or global roles allow services that are outside the database to associate database roles to authenticated users.

Database external roles are associated with operating system and RADIUS groups. This way, database user authorization can be managed externally from the database.

An external user must be authorized by an external service, such as an operating system or a third-party service, before the external user can enable the role.

Global roles are used by globally authenticated users, using centrally managed users or Oracle Enterprise User Security. A global user must be authorized to use the role by the enterprise directory service before the role is enabled at login time.

• To create a role that is to be authorized externally, include the IDENTIFIED EXTERNALLY clause in the CREATE ROLE statement.

For example:

CREATE ROLE clerk external IDENTIFIED EXTERNALLY;

To create a role to be authorized globally, use the CREATE ROLE statement.



For example:

CREATE ROLE clerk global IDENTIFIED GLOBALLY;

You can authorize roles globally to a user through a directory service mapping such as with centrally managed users.

Related Topics

Grants of Roles Using the Operating System or Network

Using the operating system or network to manage roles can help centralize the role management in a large enterprise.

Configuring RADIUS Authentication

RADIUS is a client/server security protocol widely used to enable remote authentication and access.

Mapping a Directory Group to a Global Role

Database global roles mapped to directory groups give member users additional privileges and roles above what they have been granted through their login schemas.

Oracle Database Enterprise User Security Administrator's Guide

4.8.3.5 Altering a Role

The ALTER ROLE statement can modify the authorization method for a role.

To alter the authorization method for a role, you must have the ALTER ANY ROLE system privilege or have been granted the role with ADMIN option.

Remember that you can only directly grant secure application roles or password-authenticated roles to a user. Be aware that if you create a common role in the root, you cannot change it to a local role.

• To alter a role, use the ALTER ROLE statement.

For example, to alter the clerk role to specify that the user must be authorized by an external source before enabling the role:

ALTER ROLE clerk IDENTIFIED EXTERNALLY;

4.8.4 Specifying the Type of Role Authorization

You can configure a role to be authorized through different sources, such the database or an external source.

- Authorizing a Role by Using the Database
 You can protect a role authorized by the database by assigning the role a password.
- Authorizing a Role by Using an Application
 An application role can be enabled only by applications that use an authorized PL/SQL package.
- Authorizing a Role by Using an External Source
 Oracle Database supports the use of external roles but with certain limitations.
- Authorizing a Role by Using the Operating System
 Oracle Database supports role authentication through the operating system but with certain limitations.
- Authorizing a Role by Using a Network Client
 Oracle Database supports role authentication by a network client but you must be aware of
 security risks.

Authorizing a Global Role by an Enterprise Directory Service
 A global role enables a global user to be authorized only by an enterprise directory service.

4.8.4.1 Authorizing a Role by Using the Database

You can protect a role authorized by the database by assigning the role a password.

If you are granted a role protected by a password, then you can enable or disable the role by supplying the proper password for the role in the SET ROLE statement. You cannot authenticate a password-authenticated role on logon, even if the role is a member of your list of default roles. You must explicitly enable it with the SET ROLE statement using the required password.

1. Use the CREATE ROLE statement with the IDENTIFIED BY clause to create the password-authenticated role.

Creating a Role That Is Authenticated With a Password shows a CREATE ROLE statement that creates a role called clerk. When the role is enabled, the password must be supplied.

2. Use the SET ROLE statement to set the password-authenticated role.

The following example shows how to set a password-authenticated role by using the SET ROLE statement.

SET ROLE clerk IDENTIFIED BY password;

See Guidelines for Securing Passwords.

4.8.4.2 Authorizing a Role by Using an Application

An application role can be enabled only by applications that use an authorized PL/SQL package.

Application developers do not need to secure a role by embedding passwords inside applications. Instead, they can create an application role (secure application role) and specify which PL/SQL package is authorized to enable the role.

• To create a role enabled by an authorized PL/SQL package, use the IDENTIFIED USING package name clause in the CREATE ROLE SQL statement.

For example, to indicate that the role <code>admin_role</code> is an application role and the role can only be enabled by any module defined inside the PL/SOL package <code>hr.admin</code>:

CREATE ROLE admin role IDENTIFIED USING hr.admin;

Related Topics

- Role Privileges and Secure Application Roles
 A secure application role can be enabled only by an authorized PL/SQL package or procedure.
- Creating Secure Application Roles to Control Access to Applications
 A secure application role is only enabled through its associated PL/SQL package or procedure.



4.8.4.3 Authorizing a Role by Using an External Source

Oracle Database supports the use of external roles but with certain limitations.

You can define an external role locally in the database, but you cannot grant the external role to global users, to global roles, or to any other roles in the database. You can create roles that are authorized by the operating system or network clients.

• To authorize a role by using an external source, use the CREATE ROLE statement with the IDENTIFIED EXTERNALLY clause.

For example:

CREATE ROLE accts rec IDENTIFIED EXTERNALLY;

4.8.4.4 Authorizing a Role by Using the Operating System

Oracle Database supports role authentication through the operating system but with certain limitations.

Role authentication through the operating system is useful only when the operating system is able to dynamically link operating system privileges with applications.

When a user starts an application, the operating system grants an operating system privilege to the user. The granted operating system privilege corresponds to the role associated with the application. At this point, the application can enable the application role. When the application is terminated, the previously granted operating system privilege is revoked from the operating system account of the user.

• If a role is authorized by the operating system, then configure information for each user at the operating system level. This operation is operating system dependent.

If roles are granted by the operating system, then you do not need to have the operating system authorize them also.

Related Topics

Grants of Roles Using the Operating System or Network
 Using the operating system or network to manage roles can help centralize the role
 management in a large enterprise.

4.8.4.5 Authorizing a Role by Using a Network Client

Oracle Database supports role authentication by a network client but you must be aware of security risks.

If users connect to the database over Oracle Net, then by default, the operating system cannot authenticate their roles. This includes connections through a shared server configuration, as this connection requires Oracle Net. This restriction is the default because a remote user could impersonate another operating system user over a network connection. Oracle recommends that you set REMOTE OS ROLES to FALSE, which is the default.

• If you are not concerned with this security risk and want to use operating system role authentication for network clients, then set the initialization parameter REMOTE_OS_ROLES in the database initialization parameter file to TRUE.

The change takes effect the next time you start the instance and mount the database.



4.8.4.6 Authorizing a Global Role by an Enterprise Directory Service

A global role enables a global user to be authorized only by an enterprise directory service.

You define the global role locally in the database by granting privileges and roles to it, but you cannot grant the global role itself to any user or other role in the database. When a global user attempts to connect to the database, the enterprise directory is queried to obtain any global roles associated with the user. Global roles are one component of enterprise user security. A global role only applies to one database, but you can grant it to an enterprise role defined in the enterprise directory. An enterprise role is a directory structure that contains global roles on multiple databases and can be granted to enterprise users.

• To create a global role to be authorized by an enterprise directory service, use the CREATE ROLE statement with the IDENTIFIED GLOBALLY clause.

For example:

CREATE ROLE supervisor IDENTIFIED GLOBALLY;

See Also:

- Global User Authentication and Authorization for a general discussion of global authentication and authorization of users, and its role in enterprise user management
- Oracle Database Enterprise User Security Administrator's Guide for information about implementing enterprise user management

4.8.5 Granting and Revoking Roles

You can grant or revoke privileges to and from roles, and then grant these roles to users or to other roles.

- About Granting and Revoking Roles
 You can grant system or object privileges to a role, and grant any role to any database user or to another role.
- Who Can Grant or Revoke Roles?
 The GRANT ANY ROLE system privilege enables users to grant or revoke any role except global roles to or from other users or roles.
- Granting and Revoking Roles to and from Program Units
 You can grant roles to function, procedure, and PL/SQL package program units.

4.8.5.1 About Granting and Revoking Roles

You can grant system or object privileges to a role, and grant any role to any database user or to another role.

However, a role cannot be granted to itself, nor can the role be granted circularly, that is, role x cannot be granted to role y if role y has previously been granted to role y.

To provide selective availability of privileges, Oracle Database permits applications and users to enable and disable roles. Each role granted to a user is, at any given time, either enabled or

disabled. The security domain of a user includes the privileges of all roles currently enabled for the user and excludes the privileges of any roles currently disabled for the user.

A role granted to a role is called an indirectly granted role. You can explicitly enable or disable it for a user. However, whenever you enable a role that contains other roles, you implicitly enable all indirectly granted roles of the directly granted role.

You grant roles by using the GRANT statement, and revoke them by using the REVOKE statement. Privileges are granted to and revoked from roles using the same statements.

You cannot grant a secure role (that is, an IDENTIFIED BY role, IDENTIFIED USING role, or IDENTIFIED EXTERNALLY role) to either another secure role or to a non-secure role. You can use the SET ROLE statement to enable the secure role for the session.

4.8.5.2 Who Can Grant or Revoke Roles?

The GRANT ANY ROLE system privilege enables users to grant or revoke any role except global roles to or from other users or roles.

A global role is managed in a directory, such as Oracle Internet Directory, but its privileges are contained within a single database. By default, the SYS or SYSTEM user has the GRANT ANY ROLE privilege. You should grant this system privilege conservatively because it is very powerful.

Any user granted a role with the ADMIN OPTION can grant or revoke that role to or from other users or roles of the database. This option allows administrative powers for roles to be granted on a selective basis.

See Also:

Oracle Database Enterprise User Security Administrator's Guide for information about global roles

4.8.5.3 Granting and Revoking Roles to and from Program Units

You can grant roles to function, procedure, and PL/SQL package program units.

The role then becomes enabled during the execution of the program unit, but not during the compilation of the program unit. This enables you to temporarily escalate privileges in the PL/SQL code without granting the role directly to the user. It also increases security for applications and helps to enforce the principle of least privilege.

Use the GRANT or REVOKE statement to grant or revoke a role to a program unit.

The following example shows how to grant the same role to the PL/SQL package checkstats_pkg:

GRANT clerk admin TO package psmith.checkstats pkg;

This example shows how to revoke the clerk_admin role from the PL/SQL package checkstats pkg:

REVOKE clerk_admin FROM package psmith.checkstats_pkg;

The following example shows how to grant the role clerk_admin to the procedure psmith.check stats proc.



GRANT clerk admin TO PROCEDURE psmith.checkstats proc;

Related Topics

Using Code Based Access Control for Definer's Rights and Invoker's Rights
 Code based access control, used to attach database roles to PL/SQL functions,
 procedures, or packages, works well with invoker's rights and definer's procedures.

4.8.6 Dropping Roles

Dropping a role affects the security domains of users or roles who had been granted the role.

That is, the security domains of all users and roles that were granted to the dropped role are changed to reflect the absence of the dropped role privileges.

All indirectly granted roles of the dropped role are also removed from affected security domains. Dropping a role automatically removes the role from all user default role lists.

Because the existence of objects is not dependent on the privileges received through a role, tables and other objects are not dropped when a role is dropped.

To drop a role, you must have the DROP ANY ROLE system privilege or have been granted the role with the ADMIN option.

To drop a role, use the DROP ROLE statement.

For example, to drop the role CLERK:

DROP ROLE clerk;

4.8.7 Restricting SQL*Plus Users from Using Database Roles

You should restrict SQL*Plus users from using database roles, which helps to safeguard the database from intruder attacks.

- Potential Security Problems of Using Ad Hoc Tools
 Ad hoc tools can pose problems if malicious users have access to such tools.
- How the PRODUCT_USER_PROFILE System Table Can Limit Roles
 The SYSTEM schema PRODUCT_USER_PROFILE table can disable SQL and SQL*Plus
 commands in the SQL*Plus environment for each user.
- How Stored Procedures Can Encapsulate Business Logic
 Stored procedures encapsulate privileges use with business logic so that privileges are only exercised in the context of a well-formed business transaction.

4.8.7.1 Potential Security Problems of Using Ad Hoc Tools

Ad hoc tools can pose problems if malicious users have access to such tools.

Prebuilt database applications explicitly control the potential actions of a user, including the enabling and disabling of user roles while using the application. By contrast, ad hoc query tools such as SQL*Plus, permit a user to submit any SQL statement (which may or may not succeed), including enabling and disabling a granted role.

Potentially, an application user can exercise the privileges attached to that application to issue destructive SQL statements against database tables by using an ad hoc tool.

For example, consider the following scenario:

The Vacation application has a corresponding vacation role.



- The vacation role includes the privileges to issue SELECT, INSERT, UPDATE, and DELETE statements against the emp tab table.
- The Vacation application controls the use of privileges obtained through the vacation role.

Now, consider a user who has been granted the <code>vacation</code> role. Suppose that, instead of using the Vacation application, the user executes SQL*Plus. At this point, the user is restricted only by the privileges granted to him explicitly or through roles, including the <code>vacation</code> role. Because SQL*Plus is an ad hoc query tool, the user is not restricted to a set of predefined actions, as with designed database applications. The user can query or modify data in the <code>emp_tab</code> table as he or she chooses.

4.8.7.2 How the PRODUCT_USER_PROFILE System Table Can Limit Roles

The SYSTEM schema PRODUCT_USER_PROFILE table can disable SQL and SQL*Plus commands in the SQL*Plus environment for each user.

SQL*Plus, not the Oracle Database, enforces this security. You can even restrict access to the GRANT, REVOKE, and SET ROLE commands to control user ability to change their database privileges.

The PRODUCT_USER_PROFILE table enables you to list roles that you do not want users to activate with an application. You can also explicitly disable the use of various commands, such as SET ROLE.

For example, you could create an entry in the PRODUCT USER PROFILE table to:

- Disallow the use of the clerk and manager roles with SQL*Plus
- Disallow the use of SET ROLE with SQL*Plus

Suppose user Marla connects to the database using SQL*Plus. Marla has the clerk, manager, and analyst roles. As a result of the preceding entry in PRODUCT_USER_PROFILE, Marla is only able to exercise her analyst role with SQL*Plus. Also, when Ginny attempts to issue a SET ROLE statement, she is explicitly prevented from doing so because of the entry in the PRODUCT_USER_PROFILE table prohibiting use of SET ROLE.

Be aware that the PRODUCT_USER_PROFILE table does not completely guarantee security, for multiple reasons. In the preceding example, while SET ROLE is disallowed with SQL*Plus, if Marla had other privileges granted to her directly, then she could exercise these using SQL*Plus.

See Also:

SQL*Plus User's Guide and Reference for more information about the PRODUCT_USER_PROFILE table

4.8.7.3 How Stored Procedures Can Encapsulate Business Logic

Stored procedures encapsulate privileges use with business logic so that privileges are only exercised in the context of a well-formed business transaction.

For example, an application developer can create a procedure to update the employee name and address in the <code>employees</code> table, which enforces that the data can only be updated in normal business hours.



In addition, rather than grant a human resources clerk the <code>UPDATE</code> privilege on the <code>employees</code> table, a security administrator may grant the privilege on the procedure only. Then, the human resources clerk can exercise the privilege only in the context of the procedures, and cannot update the <code>employees</code> table directly.

4.8.8 Role Privileges and Secure Application Roles

A secure application role can be enabled only by an authorized PL/SQL package or procedure.

The PL/SQL package itself reflects the security policies that are necessary to control access to the application.

This method of role creation restricts the enabling of this type of role to the invoking application. For example, the application can perform authentication and customized authorization, such as checking whether the user has connected through a proxy.

This type of role strengthens security because passwords are not embedded in application source code or stored in a table. This way, the actions the database performs are based on the implementation of your security policies, and these definitions are stored in one place, the database, rather than in your applications. If you need to modify the policy, you do so in one place without having to modify your applications. No matter how users connect to the database, the result is always the same, because the policy is bound to the role.

To enable the secure application role, you must execute its underlying package by invoking it directly from the application when the user logs in, before the user exercises the privileges granted by the secure application role. You cannot use a logon trigger to enable a secure application role, nor can you have this type of role be a default role.

When you enable the secure application role, Oracle Database verifies that the authorized PL/SQL package is on the calling stack, that is, it verifies that the authorized PL/SQL package is issuing the command to enable the role.

You can use secure application roles to ensure the existence of a database connection. Because a secure application role is a role implemented by a package, the package can validate that users can connect to the database through a middle tier or from a specific IP address. In this way, the secure application role prevents users from accessing data outside an application. They are forced to work within the framework of the application privileges that they have been granted.

Related Topics

Creating Secure Application Roles to Control Access to Applications
 A secure application role is only enabled through its associated PL/SQL package or procedure.

4.9 Restricting Operations on PDBs Using PDB Lockdown Profiles

You can use PDB lockdown profiles in a multitenant environment to restrict sets of user operations in pluggable databases (PDBs).

This section contains the following topics:

About PDB Lockdown Profiles
 A PDB lockdown profile is a named set of features that controls a group of operations.



PDB Lockdown Profile Inheritance

PDB lockdown profiles have inheritance behaviors between the CDB root, the application root, and their associated PDBs.

Default PDB Lockdown Profiles

Oracle Database provides a set of default PDB lockdown profiles that you can customize for your site requirements.

Creating a PDB Lockdown Profile

To create a PDB lockdown profile, you must have the CREATE LOCKDOWN PROFILE system privilege.

- Enabling or Disabling a PDB Lockdown Profile

 To enable or disable a PDB lockdown profile, use the PDB LOCKDOWN initialization parameter
- Dropping a PDB Lockdown Profile
 To drop a PDB lockdown profile, you must have the DROP LOCKDOWN PROFILE system privilege and be logged into the CDB or application root.

4.9.1 About PDB Lockdown Profiles

A PDB lockdown profile is a named set of features that controls a group of operations.

In some cases, you can enable or disable operations individually. For example, a PDB lockdown profile can contain settings to disable specific clauses that come with the ALTER SYSTEM statement.

PDB lockdown profiles restrict user access to the functionality the features provided, similar to resource limits that are defined for users. As the name suggests, you use PDB lockdown profiles in a CDB, for an application container, or for a PDB or application PDB. You can create custom profiles to accommodate the requirements of your site. PDB profiles enable you to define custom security policies for an application. In addition, you can create a lockdown profile that is based on another profile, called a **base profile**. You can configure this profile to be dynamically updated when the base profile is modified, or configure it to be static (unchanging) when the base profile is updated. Lockdown profiles are designed for both Oracle Cloud and on-premises environments.

When identities are shared between PDBs, elevated privileges may exist. You can use lockdown profiles to prevent this elevation of privileges. Identities can be shared in the following situations:

- At the operating system level, when the database interacts with operating system resources such as files or processes
- At the network level, when the database communicates with other systems, and network identity is important
- Inside the database, as PDBs access or create common objects or they communicate across container boundaries using features such as database links

The features that use shared identifies and that benefit from PDB lockdown profiles are in the following categories:

- Network access features. These are operations that use the network to communicate outside the PDB. For example, the PL/SQL packages UTL_TCP, UTL_HTTP, UTL_MAIL, UTL_SNMP, UTL_INADDR, and DBMS_DEBUG_JDWP perform these kinds of operations. Currently, ACLs are used to control this kind of access to share network identity.
- Common user or object access. These are operations in which a local user in the PDB
 can proxy through common user accounts or access objects in a common schema. These
 kinds of operations include adding or replacing objects in a common schema, granting

privileges to common objects, accessing common directory objects, granting the INHERIT PRIVILEGES role to a common user, and manipulating a user proxy to a common user.

- Operating System access. For example, you can restrict access to the UTL_FILE or DBMS FILE TRANSFER PL/SQL packages.
- Connections. For example, you can restrict common users from connecting to the PDB or you can restrict a local user who has the SYSOPER administrative privilege from connecting to a PDB that is open in restricted mode.

The general procedure for creating a PDB lockdown profile is to first create it in the CDB root or the application root using the CREATE LOCKDOWN PROFILE statement, and then use the ALTER LOCKDOWN PROFILE statement to add the restrictions.

To enable a PDB lockdown profile, you can use the ALTER SYSTEM statement to set the PDB_LOCKDOWN parameter. You can find information about existing PDB lockdown profiles by connecting to CDB or application root and querying the DBA_LOCKDOWN_PROFILES data dictionary view. A local user can find the contents of a PDB lockdown parameter by querying the V\$LOCKDOWN RULES dynamic data dictionary view.

4.9.2 PDB Lockdown Profile Inheritance

PDB lockdown profiles have inheritance behaviors between the CDB root, the application root, and their associated PDBs.

- The inheritance path between PDBs and their respective roots is as follows:
 - The PDB_LOCKDOWN parameter setting in a CDB PDB takes precedence over the PDB_LOCKDOWN parameter setting in the CDB root. Similarly, the PDB_LOCKDOWN setting in an application PDB takes precedence over a PDB_LOCKDOWN setting in the application root.
 - If a CDB PDB (or an application PDB) does not have the PDB_LOCKDOWN parameter set, then the PDB inherits the settings of the PDB_LOCKDOWN parameter in the CDB root (or the application root).
 - If the application root does not have the PDB_LOCKDOWN parameter set, then the
 application root inherits the settings of the PDB_LOCKDOWN parameter in the CDB root.
- If the PDB_LOCKDOWN parameter in a CDB PDB or an application PDB is set to a CDB lockdown profile, then the PDB ignores any lockdown profiles that are set by the PDB LOCKDOWN parameter in the CDB root or the application root.
- PDB lockdown parameters can inherit rules that are stipulated in an application lockdown profile, including the disable rules that come from a CDB lockdown profile that was set in its nearest ancestor (that is, an application root or the CDB root). This applies in the case of when a PDB_LOCKDOWN parameter in an application PDB is set to an application lockdown profile while the PDB_LOCKDOWN parameter in the application root or the CDB root is set to a CDB lockdown profile.
- Sometimes a conflict arises between the rules that comprise a CDB lockdown profile and an application lockdown profile. In this case, the rules in the CDB lockdown profile take precedence. For example, the setting for an OPTION_VALUE clause in the CDB lockdown profile takes precedence over the setting for the OPTION_VALUE clause in an application lockdown profile.



4.9.3 Default PDB Lockdown Profiles

Oracle Database provides a set of default PDB lockdown profiles that you can customize for your site requirements.

By default, most of these profiles are empty. They are designed to be a placeholder or template for you to configure, depending on your deployment requirements.

Detailed information about these profiles is as follows:

- PRIVATE_DBAAS incorporates restrictions that are suitable for private Cloud Database-as-a-Service (DBaaS) deployments. These restrictions are:
 - Must have the same database administrator for each PDB
 - Different users permitted to connect to the database
 - Different applications permitted

PRIVATE_DBAAS permits users to connect to the PDBs but prevents them from using Oracle Database administrative features.

- SAAS incorporates restrictions that are suitable for Software-as-a-Service (SaaS) deployments. These restrictions are:
 - Must have the same database administrator for each PDB
 - Different users permitted to connect to the database
 - Must use the same application

The SAAS lockdown profile is more restrictive than the PRIVATE_DBAAS profile. Users can be different, but the application code is the same; users are prevented from directly connecting and must connect only through the application; and users are not granted the ability to perform any administrative features.

- PUBLIC_DBAAS incorporates restrictions that are suitable for public Cloud Database-as-a-Service (DBaaS) deployments. The restrictions are as follows:
 - Different DBAs in each PDB
 - Different users
 - Different applications

The PUBLIC_DBAAS lockdown profile is the most restrictive of the lockdown profiles.

4.9.4 Creating a PDB Lockdown Profile

To create a PDB lockdown profile, you must have the CREATE LOCKDOWN PROFILE system privilege.

After you create the lockdown profile, you can add restrictions before enabling it.

 Connect to the CDB root or the application root as a user who has the CREATE LOCKDOWN PROFILE system privilege.

For example, to connect to the CDB root:

CONNECT c##sec_admin
Enter password: password



2. Run the CREATE LOCKDOWN PROFILE statement to create the profile by using the following syntax:

```
CREATE LOCKDOWN PROFILE profile_name
[FROM static base profile | INCLUDING dynamic base profile];
```

In this specification:

- profile_name is the name that you assign the lockdown profile. You can find existing
 names by querying the PROFILE_NAMES column of the DBA_LOCKDOWN_PROFILES data
 dictionary view.
- FROM static_base_profile creates a new lockdown profile by using the values from an existing profile. Any subsequent changes to the base profile will not affect the new profile.
- INCLUDING dynamic_base_profile also creates a new lockdown profile by using the values from an existing base profile, except that this new lockdown profile will inherit the DISABLE STATEMENT rules that comprise the base profile, as well as any subsequent changes to the base profile. If rules that are explicitly added to the new profile conflict with the rules in the base profile, then the rules in the base profile take precedence. For example, an OPTION_VALUE clause in the base profile takes precedence over the OPTION_VALUE clause in the new profile.

The following two PDB lockdown profile statements demonstrate how the inheritance works:

```
CREATE LOCKDOWN PROFILE hr_prof INCLUDING PRIVATE_DBAAS; CREATE LOCKDOWN PROFILE hr prof2 FROM hr prof;
```

In the first statement, <code>hr_prof</code> inherits any changes made to the <code>PRIVATE_DBAAS</code> base profile. If a new statement is enabled for <code>PRIVATE_DBAAS</code>, then it is enabled for <code>hr_prof</code>. In the second statement, in contrast, when <code>hr_prof</code> changes, then <code>hr_prof2</code> does not change because it is independent of its base profile.

3. Run the ALTER LOCKDOWN PROFILE statement to provide restrictions for the profile.

For example:

```
ALTER LOCKDOWN PROFILE hr_prof DISABLE STATEMENT = ('ALTER SYSTEM');
ALTER LOCKDOWN PROFILE hr_prof ENABLE STATEMENT = ('ALTER SYSTEM') clause = ('flush shared_pool');
ALTER LOCKDOWN PROFILE hr prof DISABLE FEATURE = ('XDB PROTOCOLS');
```

In the preceding example:

- DISABLE STATEMENT = ('ALTER SYSTEM') disables the use of all ALTER SYSTEM statements for the PDB.
- ENABLE STATEMENT = ('ALTER SYSTEM') clause = ('flush shared_pool') enables only the use of the FLUSH SHARED POOL clause for ALTER SYSTEM.
- DISABLE FEATURE = ('XDB_PROTOCOLS') prohibits the use of the XDB protocols (FTP, HTTP, HTTPS) by this PDB

After you create a PDB lockdown profile, you are ready to enable it by using the ALTER SYSTEM SET PDB LOCKDOWN SQL statement.

4.9.5 Enabling or Disabling a PDB Lockdown Profile

To enable or disable a PDB lockdown profile, use the PDB LOCKDOWN initialization parameter

You can use ALTER SYSTEM SET PDB_LOCKDOWN to enable a lockdown profile in any of the following contexts:

- CDB (affects all PDBs)
- Application root (affects all application PDBs in the container)
- Application PDB
- PDB



It is not necessary to restart the instance to enable the profile. When the ALTER SYSTEM SET PDB_LOCKDOWN statement completes, the profile rules take effect immediately.

When you set PDB_LOCKDOWN in the CDB root, every PDB and application root inherits this setting unless PDB_LOCKDOWN is set at the container level. To disable lockdown profiles, set PDB_LOCKDOWN to null. If you set this parameter to null in the CDB root, then lockdown profiles are disabled for all PDBs except those that explicitly set a profile within the PDB.

A CDB common user who has been commonly granted the SYSDBA administrative privilege or the ALTER SYSTEM system privilege can set PDB_LOCKDOWN only to a lockdown profile that was created in the CDB root. An application common user with the application common SYSDBA administrative privilege or the ALTER SYSTEM system privilege can set PDB_LOCKDOWN only to a lockdown profile created in an application root.

 Log in to the desired container as a user who has the commonly granted ALTER SYSTEM or commonly granted SYSDBA privilege.

For example, to enable the profile for all PDBs, log in to the CDB root:

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

2. Run the Alter system set PDB Lockdown statement.

For example, the following statement enables the lockdown profile named hr_prof for all PDBs:

```
ALTER SYSTEM SET PDB LOCKDOWN = hr prof;
```

The following statement resets the PDB LOCKDOWN parameter:

ALTER SYSTEM RESET PDB LOCKDOWN;



This variation of the preceding statement includes the SCOPE clause::

```
ALTER SYSTEM RESET PDB LOCKDOWN SCOPE = BOTH;
```

The following statement disables all lockdown profiles in the CDB except those that are explicitly set at the PDB level:

```
ALTER SYSTEM SET PDB LOCKDOWN = '' SCOPE = BOTH;
```

To find the names of PDB lockdown profiles, query the PROFILE_NAME column of the DBA LOCKDOWN PROFILES data dictionary view.

3. Optionally, review information about the profiles by querying DBA_LOCKDOWN_PROFILES.

For example, run the following query:

```
SET LINESIZE 150

COL PROFILE_NAME FORMAT a20

COL RULE FORMAT a20

COL CLAUSE FORMAT a25

SELECT PROFILE NAME, RULE, CLAUSE, STATUS FROM CDB LOCKDOWN PROFILES;
```

Sample output appears below:

PROFILE_NAME	RULE	CLAUSE	STATUS
HR_PROF	XDB_PROTOCOLS		DISABLE
HR_PROF	ALTER SYSTEM		DISABLE
HR_PROF	ALTER SYSTEM	FLUSH SHARED_POOL	ENABLE
HR_PROF2			EMPTY
PRIVATE_DBAAS			EMPTY
PUBLIC_DBAAS			EMPTY
SAAS			EMPTY

4.9.6 Dropping a PDB Lockdown Profile

To drop a PDB lockdown profile, you must have the DROP LOCKDOWN PROFILE system privilege and be logged into the CDB or application root.

You can find the names of existing PDB lockdown profiles by querying the DBA LOCKDOWN PROFILES data dictionary view.

 Connect to the CDB root or the application root as a user who has the DROP LOCKDOWN PROFILE system privilege.

For example, to connect to the CDB root:

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

2. Run the DROP LOCKDOWN_PROFILE statement.

For example:

DROP LOCKDOWN PROFILE hr prof2;

3. Optionally, review the current list of profiles by querying DBA LOCKDOWN PROFILES.

For example, run the following query:

```
SET LINESIZE 150

COL PROFILE_NAME FORMAT a20

COL RULE FORMAT a20

COL CLAUSE FORMAT a25

SELECT PROFILE NAME, RULE, CLAUSE, STATUS FROM CDB LOCKDOWN PROFILES;
```

Sample output appears below:

PROFILE_NAME	RULE	CLAUSE	STATUS
HR_PROF	XDB_PROTOCOLS		DISABLE
HR_PROF	ALTER SYSTEM		DISABLE
HR_PROF	ALTER SYSTEM	FLUSH SHARED_POOL	ENABLE
PRIVATE_DBAAS			EMPTY
PUBLIC_DBAAS			EMPTY
SAAS			EMPTY

4.10 Managing Object Privileges

Object privileges enable you to perform actions on schema objects, such as tables or indexes.

About Object Privileges

An object privilege grants permission to perform a particular action on a specific schema object.

Who Can Grant Object Privileges?

A user automatically has all object privileges for schema objects contained in his or her schema.

Grants and Revokes of Object Privileges

You can grant privileges to or revoke privileges from objects either directly to a user or through roles.

READ and SELECT Object Privileges

The READ and SELECT privileges provide different layers of query privileges.

Object Privilege Use with Synonyms

The CREATE SYNONYM statement create synonyms for database objects.

Sharing Application Common Objects

Database objects can be configured so that their metadata links, data links, and extended data links can be shared in the application root.

4.10.1 About Object Privileges

An object privilege grants permission to perform a particular action on a specific schema object.

Different object privileges are available for different types of schema objects. The privilege to delete rows from the departments table is an example of an object privilege.

Some schema objects, such as clusters, indexes, triggers, and database links, do not have associated object privileges. Their use is controlled with system privileges. For example, to alter a cluster, a user must own the cluster or have the ALTER ANY CLUSTER system privilege.

Some examples of object privileges include the right to:

- Use an edition
- Update a table
- Select rows from another user's table
- Execute a stored procedure of another user

See Also:

- How Commonly Granted Object Privileges Work
- Oracle Database SQL Language Reference for a list of object privileges and the operations they authorize

4.10.2 Who Can Grant Object Privileges?

A user automatically has all object privileges for schema objects contained in his or her schema.

A user with the GRANT ANY OBJECT PRIVILEGE system privilege can grant any specified object privilege to another user with or without the WITH GRANT OPTION clause of the GRANT statement. A user with the GRANT ANY OBJECT PRIVILEGE privilege can also use that privilege to revoke any object privilege that was granted either by the object owner or by some other user with the GRANT ANY OBJECT PRIVILEGE privilege.

If the grantee does not have the GRANT ANY OBJECT PRIVILEGE privilege or had been granted the privilege without the WITH GRANT OPTION clause of the GRANT statement, then this user cannot grant the privilege to other users.

The WITH GRANT OPTION can be used only with object privilege grants to users. It cannot be used for object privilege grants to roles.



Oracle Database SQL Language Reference for information about GRANT and GRANT ANY OBJECT PRIVILEGE

4.10.3 Grants and Revokes of Object Privileges

You can grant privileges to or revoke privileges from objects either directly to a user or through roles.

About Granting and Revoking Object Privileges
 Object privileges can be granted to and revoked from users and roles.



How the ALL Clause Grants or Revokes All Available Object Privileges
 Each type of object has different privileges associated with it, which can be controlled by
 the ALL clause.

4.10.3.1 About Granting and Revoking Object Privileges

Object privileges can be granted to and revoked from users and roles.

If you grant object privileges to roles, then you can make the privileges selectively available To grant object privileges, you can use the GRANT statement; to revoke object privileges, you can use the REVOKE statement.

4.10.3.2 How the ALL Clause Grants or Revokes All Available Object Privileges

Each type of object has different privileges associated with it, which can be controlled by the ALL clause.

You can specify ALL [PRIVILEGES] to grant or revoke all available object privileges for an object. ALL is not a privilege. Rather, it is a shortcut, or a way of granting or revoking all object privileges with one GRANT and REVOKE statement. If all object privileges are granted using the ALL shortcut, then individual privileges can still be revoked.

Similarly, you can revoke all individually granted privileges by specifying ALL. However, if you REVOKE ALL, and revoking causes integrity constraints to be deleted (because they depend on a REFERENCES privilege that you are revoking), then you must include the CASCADE CONSTRAINTS option in the REVOKE statement.

Example 4-3 revokes all privileges on the orders table in the HR schema using CASCADE CONSTRAINTS.

Example 4-3 Revoking All Object Privileges Using CASCADE CONSTRAINTS

REVOKE ALL
ON ORDERS FROM HR
CASCADE CONSTRAINTS;

4.10.4 READ and SELECT Object Privileges

The READ and SELECT privileges provide different layers of query privileges.

- About Managing READ and SELECT Object Privileges
 You can grant users either the READ or the SELECT object privilege.
- Enabling Users to Use the READ Object Privilege to Query Any Table in the Database The READ ANY TABLE system privilege provides the READ object privilege for querying any table in the database.
- Restrictions on the READ and READ ANY TABLE Privileges
 There are special restrictions on the READ and READ ANY TABLE privileges.

4.10.4.1 About Managing READ and SELECT Object Privileges

You can grant users either the READ or the SELECT object privilege.

The grant of these privileges depend on the level of access that you want to allow the user.

Follow these guidelines:

• If you want the user only to be able to query tables, views, materialized views, or synonyms, then you should grant the READ object privilege. For example:

```
GRANT READ ON HR.EMPLOYEES TO psmith;
```

- If you want the user to be able to perform the following actions in addition to performing the query, then you should grant the user the SELECT object privilege:
 - LOCK TABLE table name IN EXCLUSIVE MODE;
 - SELECT ... FROM table name FOR UPDATE;

For example:

```
GRANT SELECT ON HR.EMPLOYEES TO psmith;
```

In either case, user psmith would use a SELECT statement to perform query.

Related Topics

Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges
 The CREATE AUDIT POLICY statement can audit the READ ANY TABLE and SELECT ANY TABLE privileges.

4.10.4.2 Enabling Users to Use the READ Object Privilege to Query Any Table in the Database

The READ ANY TABLE system privilege provides the READ object privilege for querying any table in the database.

• To enable a user to have the READ object privilege for any table in the database, grant the user the READ ANY TABLE system privilege.

For example:

```
GRANT READ ANY TABLE TO psmith;
```

As with the READ object privilege, the READ ANY TABLE system privilege does not enable users to lock tables in exclusive mode nor select tables for update operations. Conversely, the SELECT ANY TABLE system privilege enables users to lock the rows of a table, or lock the entire table, through a SELECT ... FOR UPDATE statement, in addition to guerying any table.

4.10.4.3 Restrictions on the READ and READ ANY TABLE Privileges

There are special restrictions on the READ and READ ANY TABLE privileges.

These privileges are as follows:

- The READ object privilege has no effect on the requirements of the SQL92_SECURITY standard. If the SQL92_SECURITY initialization parameter has been set to TRUE, then its requirement that users must be granted the SELECT object privilege in addition to UPDATE or DELETE in order to execute the UPDATE or DELETE statements is not relaxed to require that READ is sufficient instead of SELECT.
- If Oracle Database Vault is enabled, remember that the SQL92_SECURITY initialization parameter is automatically set to TRUE. Hence, UPDATE and DELETE statements will fail if the user has only been granted the READ object privilege or the READ ANY TABLE system privilege. In this case, you must grant the user the SELECT object privilege or, if the user is a trusted user, the SELECT ANY TABLE system privilege.



4.10.5 Object Privilege Use with Synonyms

The CREATE SYNONYM statement create synonyms for database objects.

You can create synonyms for the following objects: tables, views, sequences, operators, procedures, stored functions, packages, materialized views, Java class schema objects, user-defined object types, or other synonyms.

If you grant users the privilege to use the synonym, then the object privileges granted on the underlying objects apply whether the user references the base object by name or by using the synonym.

For example, suppose user OE creates the following synonym for the CUSTOMERS table:

```
CREATE SYNONYM customer_syn FOR CUSTOMERS;
```

Then OE grants the READ privilege on the customer syn synonym to user HR.

```
GRANT READ ON customer_syn TO HR;
```

User HR then tries either of the following queries:

```
SELECT COUNT(*) FROM OE.customer_syn;
SELECT COUNT(*) FROM OE.CUSTOMERS;
```

Both queries will yield the same result:

```
COUNT (*)
```

Be aware that when you grant the synonym to another user, the grant applies to the underlying object that the synonym represents, not to the synonym itself. For example, if user HR queries the ALL TAB PRIVS data dictionary view for his privileges, he will learn the following:

The results show that in addition to other privileges, he has the READ privilege for the underlying object of the customer syn synonym, which is the OE.CUSTOMER table.

At this point, if user OE then revokes the READ privilege on the customer_syn synonym from HR, here are the results if HR checks his privileges again:

```
TABLE_SCHEMA TABLE_NAME PRIVILEGE
-----
OE OE INHERIT PRIVILEGES
```

User HR no longer has the READ privilege for the OE.CUSTOMER table. If he tries to query the OE.CUSTOMERS table, then the following error appears:

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



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

4.10.6 Sharing Application Common Objects

Database objects can be configured so that their metadata links, data links, and extended data links can be shared in the application root.

- Metadata-Linked Application Common Objects
 - A metadata link enables database objects in an application pluggable database (PDB) to share metadata with objects in the application root.
- Data-Linked Application Common Objects
 Data links manage references and privileges for objects in a multitenant environment.
- Extended Data-Linked Application Common Objects
 Extended data links can combine data from an application pluggable database (PDB) with an application root.



Oracle Database Administrator's Guide for information about creating application common objects: metadata-linked objects, data-linked objects, and extended data-linked objects

4.10.6.1 Metadata-Linked Application Common Objects

A metadata link enables database objects in an application pluggable database (PDB) to share metadata with objects in the application root.

Metadata links are useful for reducing disk and memory requirements because they store only one copy of an object's metadata (such as the source code for a PL/SQL package) for identically defined objects (such as Oracle-suppled PL/SQL packages). This improves the performance of upgrade operations because changes to this metadata will be made in one place, the application root.

You must configure the metadata link from the application root. You can use the <code>DBMS_PDB.SET_MEDATADATA_LINKED PL/SQL</code> procedure to change the database object to a metadata link.

The following example shows how to use the <code>DBMS_PDB.SET_METADATA_LINKED</code> procedure to change the <code>update_emp_rating</code> procedure in the <code>hr_mgr</code> schema to a metadata-linked application common object.

Example 4-4 Changing an Object to a Metadata-Linked Application Common Object

```
BEGIN
   DBMS_PDB.SET_METADATA_LINKED (
   SCHEMA_NAME => 'hr_mgr',
   OBJECT_NAME => 'update_emp_rating',
   NAMESPACE => 1);
END;
//
```

Any common user can own metadata links. Metadata links can only be used to share the metadata of application common objects that their creator in the application root owns.

To find if an object has a metadata link, query the SHARING column of the DBA_OBJECTS data dictionary view.



Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS_PDB.SET_METADATA_LINKED procedure

4.10.6.2 Data-Linked Application Common Objects

Data links manage references and privileges for objects in a multitenant environment.

A data link (previously called an object link) enables references to, and privilege grants on, objects in an application root from an application pluggable database (PDB) that belong to the same application container.

If an application common user who owns an application common object wants to grant access to that object to a user in a PDB, then the application common user can accomplish this by granting the privilege on a data link that points to the common object. For example, you can create data links for objects such as tables, views, clusters, sequences, or PL/SQL packages if you want to ensure that an operation on the object (such as a query, a DML, an EXECUTE statement, and so on) that refers to this operation affects the same object regardless of the container in which the operation is performed.

You must configure the data link from an application root. You can use the DBMS_PDB.SET_DATA_LINKED PL/SQL procedure to change the data link. You should use this procedure only when you want to convert an existing object to become data linked.

The following example shows how to use the <code>DBMS_PDB.SET_DATA_LINKED</code> procedure to change the <code>emp_ratings</code> table in the <code>hr_mgr</code> schema to a data-linked application common object.

Example 4-5 Changing an Object to a Data-Linked Application Common Object

```
BEGIN
  DBMS_PDB.SET_DATA_LINKED (
  SCHEMA_NAME => 'hr_mgr',
  OBJECT_NAME => 'emp_ratings',
  NAMESPACE => 1);
END;
//
```

Any common user can own data links.

To find if an object has an data link, query the SHARING column of the DBA_OBJECTS data dictionary view. The NAMESPACE column of this view provides the namespace number.

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS PDB.SET DATA LINKED procedure

4.10.6.3 Extended Data-Linked Application Common Objects

Extended data links can combine data from an application pluggable database (PDB) with an application root.

An extended data link enables a data link to combine data found in a table in the PDB with data from a corresponding table in the application root.

You can think of an extended data link as a hybrid of a metadata link and a data link. An extended data-link object in an application PDB inherits metadata from the extended data link object in the application root. The data for the object is stored in the application root and, optionally, in each application PDB. You can create extended data links for tables and views only. When you query the <code>DBA_OBJECTS</code> data dictionary view for an extended data link object, this view returns extended data link-related rows from both the application PDB and the application root.

You must configure the extended data link from an application root. You can use the <code>DBMS_PDB.SET_EXT_DATA_LINKED PL/SQL</code> procedure to change the database object to an extended data link.

The following example shows how to use the <code>DBMS_PDB.SET_EXT_DATA_LINKED</code> procedure to change the <code>emp_salaries</code> data dictionary view in the <code>hr_mgr</code> schema to an extended data-linked application common object.

Example 4-6 Changing an Object to an Extended Data-Linked Application Common Object

```
BEGIN
  DBMS_PDB.SET_EXT_DATA_LINKED (
  SCHEMA_NAME => 'hr_mgr',
  OBJECT_NAME => 'emp_salaries',
  NAMESPACE => 1);
END;
/
```

Any common user can own extended data links.

To find if an object has an extended data link, query the SHARING column of the DBA_OBJECTS data dictionary view.

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS PDB.SET EXT DATA LINKED procedure

4.11 Table Privileges

Object privileges for tables enable table security at the DML or DDL level of operation.

How Table Privileges Affect Data Manipulation Language Operations
 You can grant privileges to use the DELETE, INSERT, SELECT, and UPDATE DML operations
 on tables and views.

How Table Privileges Affect Data Definition Language Operations
 The ALTER, INDEX, and REFERENCES privileges allow DDL operations to be performed on a table.

4.11.1 How Table Privileges Affect Data Manipulation Language Operations

You can grant privileges to use the DELETE, INSERT, SELECT, and UPDATE DML operations on tables and views.

Grant these privileges only to users and roles that need to query or manipulate data in a table.

You can restrict INSERT and UPDATE privileges for a table to specific columns of the table. With a selective INSERT privilege, a privileged user can insert a row with values for the selected columns. All other columns receive NULL or the default value of the column. With a selective UPDATE privilege, a user can update only specific column values of a row. You can use selective INSERT and UPDATE privileges to restrict user access to sensitive data.

For example, if you do not want data entry users to alter the salary column of the employees table, then selective INSERT or UPDATE privileges can be granted that exclude the salary column. Alternatively, a view that excludes the salary column could satisfy this need for additional security.



Oracle Database SQL Language Reference for more information about DML operations

4.11.2 How Table Privileges Affect Data Definition Language Operations

The ALTER, INDEX, and REFERENCES privileges allow DDL operations to be performed on a table.

Because these privileges allow other users to alter or create dependencies on a table, you should grant these privileges conservatively. A user attempting to perform a DDL operation on a table may need additional system or object privileges. For example, to create a trigger on a table, the user requires both the ALTER TABLE object privilege for the table and the CREATE TRIGGER system privilege.

As with the INSERT and UPDATE privileges, you can grant the REFERENCES privilege on specific columns of a table. The REFERENCES privilege enables the grantee to use the table on which the grant is made as a parent key to any foreign keys that the grantee wishes to create in his or her own tables. This action is controlled with a special privilege because the presence of foreign keys restricts the data manipulation and table alterations that can be done to the parent key. A column-specific REFERENCES privilege restricts the grantee to using the named columns (which, of course, must include at least one primary or unique key of the parent table).

See Also:

Oracle Database Concepts for more information about how data integrity works with primary keys, unique keys, and integrity constraints



4.12 View Privileges

You can apply DML object privileges to views, similar to tables.

- Privileges Required to Create Views
 To create a view, you must have specific privileges.
- Privileges to Query Views in Other Schemas
 A view owner must be granted SELECT WITH GRANT OPTION on the base table of their view before users can query the view from a schema that is different from the schema in which the view is located.
- The Use of Views to Increase Table Security
 Database views can increase table security by restricting the data that users can see.

4.12.1 Privileges Required to Create Views

To create a view, you must have specific privileges.

Object privileges for a view allow various DML operations, which affect the base tables from which the view is derived.

These privileges to create a view are as follows:

- You must be granted one of the following system privileges, either explicitly or through a role:
 - The CREATE VIEW system privilege (to create a view in your schema)
 - The CREATE ANY VIEW system privilege (to create a view in the schema of another user)
- You must be explicitly granted one of the following privileges:
 - The SELECT, INSERT, UPDATE, or DELETE object privileges on all base objects underlying the view
 - The SELECT ANY TABLE, INSERT ANY TABLE, UPDATE ANY TABLE, or DELETE ANY TABLE system privileges
- In addition, before you can grant other users access to you view, you must have object privileges to the base objects with the GRANT OPTION clause or appropriate system privileges with the ADMIN OPTION clause. If you do not have these privileges, then you cannot to grant other users access to your view. If you try, an ORA-01720: grant option does not exist for object_name error is raised, with object_name referring to the view's underlying object for which you do not have the sufficient privilege.



Oracle Database SQL Language Reference

4.12.2 Privileges to Query Views in Other Schemas

A view owner must be granted SELECT WITH GRANT OPTION on the base table of their view before users can query the view from a schema that is different from the schema in which the view is located.



4.12.3 The Use of Views to Increase Table Security

Database views can increase table security by restricting the data that users can see.

To use a view, the user must have the appropriate privileges but only for the view itself, not its underlying objects. However, if access privileges for the underlying objects of the view are removed, then the user no longer has access.

This behavior occurs because the security domain that is used when a user queries the view is that of the definer of the view. If the privileges on the underlying objects are revoked from the view's definer, then the view becomes invalid, and no one can use the view. Therefore, even if a user has been granted access to the view, the user may not be able to use the view if the definer's rights have been revoked from the view's underlying objects.

For example, suppose User A creates a view. User A has definer's rights on the underlying objects of the view. User A then grants the SELECT privilege on that view to User B so that User B can query the view. But if User A no longer has access to the underlying objects of that view, then User B no longer has access either.

Views add two more levels of security for tables, column-level security and value-based security, as follows:

A view can provide access to selected columns of base tables. For example, you can
define a view on the employees table to show only the employee_id, last_name, and
manager id columns:

```
CREATE VIEW employees_manager AS SELECT last name, employee id, manager id FROM employees;
```

A view can provide value-based security for the information in a table. A WHERE clause
in the definition of a view displays only selected rows of base tables. Consider the following
two examples:

```
CREATE VIEW lowsal AS

SELECT * FROM employees

WHERE salary < 10000;
```

The lowsal view allows access to all rows of the employees table that have a salary value less than 10000. Notice that all columns of the employees table are accessible in the lowsal view.

```
CREATE VIEW own_salary AS

SELECT last_name, salary

FROM employees

WHERE last name = USER;
```

In the <code>own_salary</code> view, only the rows with an <code>last_name</code> that matches the current user of the view are accessible. The <code>own_salary</code> view uses the <code>user</code> pseudo column, whose values always refer to the current user. This view combines both column-level security and value-based security.

4.13 Procedure Privileges

The EXECUTE privilege enables users to run procedures and functions, either standalone or in packages.

The Use of the EXECUTE Privilege for Procedure Privileges
 The EXECUTE privilege is a very powerful privilege that should be handled with caution.

Procedure Execution and Security Domains

The EXECUTE object privilege for a procedure can be used to execute a procedure or compile a program unit that references the procedure.

System Privileges Required to Create or Replace a Procedure

You must have specific privileges to create or replace a procedure in your own schema or in another user's schema.

System Privileges Required to Compile a Procedure

You must have specific privileges to compile both standalone procedures and procedures that are part of a package.

How Procedure Privileges Affect Packages and Package Objects
 The powerful EXECUTE privilege enables users to run any public procedures or functions within a package.

4.13.1 The Use of the EXECUTE Privilege for Procedure Privileges

The EXECUTE privilege is a very powerful privilege that should be handled with caution.

The EXECUTE privilege is the only **object privilege** for procedures, including standalone procedures and functions, and for those within packages.

You should grant this privilege only to users who must run a procedure or compile another procedure that calls a desired procedure. You can find the privileges that a user has been granted by querying the DBA SYS PRIVS data dictionary view.

4.13.2 Procedure Execution and Security Domains

The EXECUTE object privilege for a procedure can be used to execute a procedure or compile a program unit that references the procedure.

Oracle Database performs a run-time privilege check when any PL/SQL unit is called. A user with the EXECUTE ANY PROCEDURE system privilege can execute any procedure in the database. Privileges to run procedures can be granted to a user through roles.

See Also:

- About Definer's Rights and Invoker's Rights
- Oracle Database PL/SQL Packages and Types Reference for more information about how Oracle Database checks privileges at run-time

4.13.3 System Privileges Required to Create or Replace a Procedure

You must have specific privileges to create or replace a procedure in your own schema or in another user's schema.

To create or replace a procedure in your own schema, you must have the CREATE PROCEDURE system privilege. To create or replace a procedure in another user's schema, you must have the CREATE ANY PROCEDURE system privilege.

The user who owns the procedure also must have privileges for schema objects referenced in the procedure body. To create a procedure, you need to have been explicitly granted the necessary privileges (system or object) on all objects referenced by the procedure. You cannot obtain the required privileges through roles. This includes the EXECUTE privilege for any procedures that are called inside the procedure being created.



Triggers require that privileges on referenced objects be granted directly to the owner of the trigger. Anonymous PL/SQL blocks can use any privilege, whether the privilege is granted explicitly or through a role.

4.13.4 System Privileges Required to Compile a Procedure

You must have specific privileges to compile both standalone procedures and procedures that are part of a package.

To compile a standalone procedure, you should run the ALTER PROCEDURE statement with the COMPILE clause. To compile a procedure that is part of a package, you should run the ALTER PACKAGE statement.

The following example shows how to compile a standalone procedure.

ALTER PROCEDURE psmith.remove_emp COMPILE;

If the standalone or packaged procedure is in another user's schema, you must have the ALTER ANY PROCEDURE privilege to recompile it. You can recompile procedures in your own schema without any privileges.

4.13.5 How Procedure Privileges Affect Packages and Package Objects

The powerful EXECUTE privilege enables users to run any public procedures or functions within a package.

- About the Effect of Procedure Privileges on Packages and Package Objects
 The EXECUTE object privilege for a package applies to any procedure or function within this package.
- Example: Procedure Privileges Used in One Package
 The CREATE PACKAGE BODY statement can create a package body that contains procedures to manage procedure privileges used in one package.
- Example: Procedure Privileges and Package Objects
 The CREATE PACKAGE BODY statement can create a package body containing procedure definitions to manage procedure privileges and package objects.

4.13.5.1 About the Effect of Procedure Privileges on Packages and Package Objects

The EXECUTE object privilege for a package applies to any procedure or function within this package.

A user with the EXECUTE object privilege for a package can execute any public procedure or function in the package, and can access or modify the value of any public package variable.

You cannot grant specific EXECUTE privileges for individual constructs in a package. Therefore, you may find it useful to consider two alternatives for establishing security when developing

procedures, functions, and packages for a database application. The following examples describe these alternatives.

4.13.5.2 Example: Procedure Privileges Used in One Package

The CREATE PACKAGE BODY statement can create a package body that contains procedures to manage procedure privileges used in one package.

Example 4-7 shows four procedures created in the bodies of two packages.

Example 4-7 Procedure Privileges Used in One Packagee

```
CREATE PACKAGE BODY hire fire AS
  PROCEDURE hire (...) IS
   BEGIN
     INSERT INTO employees . . .
   END hire;
  PROCEDURE fire (...) IS
    BEGIN
     DELETE FROM employees . . .
   END fire;
END hire fire;
CREATE PACKAGE BODY raise bonus AS
 PROCEDURE give raise(...) IS
   BEGIN
     UPDATE employees SET salary = . . .
   END give raise;
  PROCEDURE give bonus (...) IS
    BEGIN
      UPDATE employees SET bonus = . . .
    END give bonus;
END raise bonus;
```

The following GRANT EXECUTE statements enable the big_bosses and little_bosses roles to run the appropriate procedures:

```
GRANT EXECUTE ON hire_fire TO big_bosses;
GRANT EXECUTE ON raise_bonus TO little_bosses;
```

4.13.5.3 Example: Procedure Privileges and Package Objects

The CREATE PACKAGE BODY statement can create a package body containing procedure definitions to manage procedure privileges and package objects.

Example 4-8 shows four procedure definitions within the body of a single package. Two additional standalone procedures and a package are created specifically to provide access to the procedures defined in the main package.

Example 4-8 Procedure Privileges and Package Objects

```
CREATE PACKAGE BODY employee_changes AS

PROCEDURE change_salary(...) IS BEGIN ... END;

PROCEDURE change_bonus(...) IS BEGIN ... END;

PROCEDURE insert_employee(...) IS BEGIN ... END;

PROCEDURE delete_employee(...) IS BEGIN ... END;

END employee_changes;

CREATE PROCEDURE hire

BEGIN

employee_changes.insert_employee(...)
```



```
END hire;

CREATE PROCEDURE fire

BEGIN

employee_changes.delete_employee(...)

END fire;

PACKAGE raise_bonus IS

PROCEDURE give_raise(...) AS

BEGIN

employee_changes.change_salary(...)

END give_raise;

PROCEDURE give_bonus(...)

BEGIN

employee_changes.change_bonus(...)

END give bonus;
```

Using this method, the procedures that actually do the work (the procedures in the <code>employee_changes</code> package) are defined in a single package and can share declared global variables, cursors, on so on. By declaring top-level procedures, hire and fire, and an additional package, <code>raise_bonus</code>, you can grant selective <code>EXECUTE</code> privileges on procedures in the main package:

```
GRANT EXECUTE ON hire, fire TO big_bosses;
GRANT EXECUTE ON raise bonus TO little bosses;
```

Be aware that granting EXECUTE privilege for a package provides uniform access to all package objects.

4.14 Type Privileges

You can control system and object privileges for types, methods, and objects.

- System Privileges for Named Types
 - System privileges for named types can enable users to perform actions such as creating named types in their own schemas.
- Object Privileges for Named Types

 The only object privilege that applies to pamed types is
 - The only object privilege that applies to named types is EXECUTE.
- Method Execution Model for Named Types
 The method execution for named types is the
 - The method execution for named types is the same as any other stored PL/SQL procedure.
- Privileges Required to Create Types and Tables Using Types
 To create a type, you must have the appropriate privileges.
- Example: Privileges for Creating Types and Tables Using Types
 The EXECUTE privilege with the GRANT OPTION is required for users to grant the EXECUTE privilege on a type to other users.
- Privileges on Type Access and Object Access
 Existing column-level and table-level privileges for DML statements apply to both column objects and row objects.
- Type Dependencies
 - As with stored objects, such as procedures and tables, types that are referenced by other objects are called dependencies.



4.14.1 System Privileges for Named Types

System privileges for named types can enable users to perform actions such as creating named types in their own schemas.

Table 4-4 lists system privileges for named types (object types, VARRAYS, and nested tables).

Table 4-4 System Privileges for Named Types

Privilege	Enables you to
CREATE TYPE	Create named types in your own schemas
CREATE ANY TYPE	Create a named type in any schema
ALTER ANY TYPE	Alter a named type in any schema
DROP ANY TYPE	Drop a named type in any schema
EXECUTE ANY TYPE	Use and reference a named type in any schema

The RESOURCE role includes the CREATE TYPE system privilege. The DBA role includes all of these privileges.

4.14.2 Object Privileges for Named Types

The only object privilege that applies to named types is EXECUTE.

If the EXECUTE privilege exists on a named type, then a user can use the named type to:

- Define a table
- Define a column in a relational table
- Declare a variable or parameter of the named type

The EXECUTE privilege permits a user to invoke the methods in the type, including the type constructor. This is similar to the EXECUTE privilege on a stored PL/SQL procedure.

4.14.3 Method Execution Model for Named Types

The method execution for named types is the same as any other stored PL/SQL procedure.

Users must be granted the appropriate privileges for using the named types, such as the <code>EXECUTE</code> privilege. As with all privilege grants, only grant these privileges to trusted users. You can find the privileges that a user has been granted by querying the <code>DBA_SYS_PRIVS</code> data dictionary view.

Related Topics

Procedure Privileges

The EXECUTE privilege enables users to run procedures and functions, either standalone or in packages.

4.14.4 Privileges Required to Create Types and Tables Using Types

To create a type, you must have the appropriate privileges.

These privileges are as follows:



- You must have the CREATE TYPE system privilege to create a type in your schema or the CREATE ANY TYPE system privilege to create a type in the schema of another user. These privileges can be acquired explicitly or through a role.
- The owner of the type must be explicitly granted the EXECUTE object privileges to access all other types referenced within the definition of the type, or have been granted the EXECUTE ANY TYPE system privilege. The owner cannot obtain the required privileges through roles.
- If the type owner intends to grant access to the type to other users, then the owner must receive the EXECUTE privileges to the referenced types with the GRANT OPTION or the EXECUTE ANY TYPE system privilege with the ADMIN OPTION. If not, then the type owner has insufficient privileges to grant access on the type to other users.

To create a table using types, you must meet the requirements for creating a table and the following additional requirements:

- The owner of the table must have been directly granted the EXECUTE Object privilege to
 access all types referenced by the table, or has been granted the EXECUTE ANY TYPE system
 privilege. The owner cannot exercise the required privileges if these privileges were
 granted through roles.
- If the table owner intends to grant access to the table to other users, then the owner must have the EXECUTE privilege to the referenced types with the GRANT OPTION or the EXECUTE ANY TYPE system privilege with the ADMIN OPTION. If not, then the table owner has insufficient privileges to grant access on the table.

Related Topics

Table Privileges
 Object privileges for tables enable table security at the DML or DDL level of operation.

4.14.5 Example: Privileges for Creating Types and Tables Using Types

The EXECUTE privilege with the GRANT OPTION is required for users to grant the EXECUTE privilege on a type to other users.

Assume that three users exist with the CONNECT and RESOURCE roles:

- user1
- user2
- user3

The following DDL is run in the schema of user1:

```
CREATE TYPE type1 AS OBJECT (
   attr1 NUMBER);

CREATE TYPE type2 AS OBJECT (
   attr2 NUMBER);

GRANT EXECUTE ON type1 TO user2;

GRANT EXECUTE ON type2 TO user2 WITH GRANT OPTION;
```

The following DDL is performed in the schema of user2:

```
CREATE TABLE tab1 OF user1.type1;
CREATE TYPE type3 AS OBJECT (
  attr3 user1.type2);
CREATE TABLE tab2 (
  col1 user1.type2);
```



The following statements succeed because user2 has EXECUTE privilege on user1.type2 with the GRANT OPTION:

```
GRANT EXECUTE ON type3 TO user3; GRANT SELECT ON tab2 TO user3;
```

However, the following grant fails because user2 does not have EXECUTE privilege on user1.type1 with the GRANT OPTION:

```
GRANT SELECT ON tab1 TO user3;
```

The following statements can be successfully run by user3:

```
CREATE TYPE type4 AS OBJECT (
  attr4 user2.type3);
CREATE TABLE tab3 OF type4;
```



The CONNECT role presently retains only the CREATE SESSION and SET CONTAINER privileges.

4.14.6 Privileges on Type Access and Object Access

Existing column-level and table-level privileges for DML statements apply to both column objects and row objects.

Table 4-5 lists the privileges for object tables.

Table 4-5 Privileges for Object Tables

Privilege	Enables you to
SELECT	Access an object and its attributes from the table
UPDATE	Modify the attributes of the objects that make up the rows in the table
INSERT	Create new objects in the table
DELETE	Delete rows

Similar table privileges and column privileges apply to column objects. Retrieving instances does not in itself reveal type information. However, clients must access named type information to interpret the type instance images. When a client requests type information, Oracle Database checks for the EXECUTE privilege on the type.

Consider the following schema:

```
CREATE TYPE emp_type (
    eno NUMBER, ename CHAR(31), eaddr addr_t);
CREATE TABLE emp OF emp_t;
```

In addition, consider the following two queries:

```
SELECT VALUE(emp) FROM emp; SELECT eno, ename FROM emp;
```



For either query, Oracle Database checks the SELECT privilege of the user for the emp table. For the first query, the user must obtain the emp_type type information to interpret the data. When the query accesses the emp_type type, Oracle Database checks the EXECUTE privilege of the user.

The second query, however, does not involve named types, so Oracle Database does not check type privileges.

In addition, by using the schema from the previous section, user3 can perform the following queries:

```
SELECT tab1.col1.attr2 FROM user2.tab1 tab1; SELECT attr4.attr3.attr2 FROM tab3;
```

Note that in both SELECT statements, user3 does not have explicit privileges on the underlying types, but the statement succeeds because the type and table owners have the necessary privileges with the GRANT OPTION.

Oracle Database checks privileges on the following events, and returns an error if the client does not have the privilege for the action:

- Pinning an object in the object cache using its REF value causes Oracle Database to check for the SELECT privilege on the containing object table.
- Modifying an existing object or flushing an object from the object cache causes Oracle Database to check for the UPDATE privilege on the destination object table.
- Flushing a new object causes Oracle Database to check for the INSERT privilege on the destination object table.
- Deleting an object causes Oracle Database to check for the DELETE privilege on the destination table.
- Pinning an object of a named type causes Oracle Database to check EXECUTE privilege on the object.

Modifying the attributes of an object in a client third-generation language application causes Oracle Database to update the entire object. Therefore, the user needs the <code>UPDATE</code> privilege on the object table. Having the <code>UPDATE</code> privilege on only certain columns of the object table is not sufficient, even if the application only modifies attributes corresponding to those columns. Therefore, Oracle Database does not support column-level privileges for object tables.

4.14.7 Type Dependencies

As with stored objects, such as procedures and tables, types that are referenced by other objects are called dependencies.

There are some special issues for types on which tables depend. Because a table contains data that relies on the type definition for access, any change to the type causes all stored data to become inaccessible. Changes that can cause this are when necessary privileges required to use the type are revoked, or the type or dependent types are dropped. If these actions occur, then the table becomes invalid and cannot be accessed.

A table that is invalid because of missing privileges can automatically become valid and accessible if the required privileges are granted again. A table that is invalid because a dependent type was dropped can never be accessed again, and the only permissible action is to drop the table.

Because of the severe effects that revoking a privilege on a type or dropping a type can cause, the SQL statements REVOKE and DROP TYPE, by default, implement restricted semantics. This



means that if the named type in either statement has table or type dependents, then an error is received and the statement cancels. However, if the FORCE clause for either statement is used, then the statement always succeeds. If there are depended-upon tables, then they are invalidated.



Oracle Database SQL Language Reference for details about using the REVOKE and DROP TYPE SQL statements

4.15 Grants of User Privileges and Roles

The GRANT statement provides privileges for a user to perform specific actions, such as executing a procedure.

- Granting System Privileges and Roles to Users and Roles
 Before you grant system privileges and roles to users and roles, be aware of how
 privileges for these types of grants work.
- Granting Object Privileges to Users and Roles
 You can grant object privileges to users and roles, and enable the grantee to grant the
 privilege to other users.

4.15.1 Granting System Privileges and Roles to Users and Roles

Before you grant system privileges and roles to users and roles, be aware of how privileges for these types of grants work.

- Privileges for Grants of System Privileges and Roles to Users and Roles
 You can use the GRANT SQL statement to grant system privileges and roles to users and
 roles.
- Example: Granting a System Privilege and a Role to a User
 You can use the GRANT statement to grant system privileges and roles to users.
- Example: Granting the EXECUTE Privilege on a Directory Object
 You can use the GRANT statement to grant the EXECUTE privilege on a directory object.
- Use of the ADMIN Option to Enable Grantee Users to Grant the Privilege
 The WITH ADMIN OPTION clause can be used to expand the capabilities of a privilege grant.
- Creating a New User with the GRANT Statement
 You can create a new user and grant this user a privilege in one GRANT SQL statement.

4.15.1.1 Privileges for Grants of System Privileges and Roles to Users and Roles

You can use the GRANT SQL statement to grant system privileges and roles to users and roles.

The following privileges are required:

- To grant a system privilege, a user must be granted the system privilege with the ADMIN option or must be granted the GRANT ANY PRIVILEGE system privilege.
- To grant a role, a user must be granted the role with the ADMIN option or was granted the GRANT ANY ROLE system privilege.



Object privileges cannot be granted along with system privileges and roles in the same GRANT statement.

4.15.1.2 Example: Granting a System Privilege and a Role to a User

You can use the GRANT statement to grant system privileges and roles to users.

Example 4-9 grants the system privilege CREATE SESSION and the accts_pay role to the user jward.

Example 4-9 Granting a System Privilege and a Role to a User

GRANT CREATE SESSION, accts pay TO jward;

4.15.1.3 Example: Granting the EXECUTE Privilege on a Directory Object

You can use the GRANT statement to grant the EXECUTE privilege on a directory object.

Example 4-9 grants the EXECUTE privilege on the exec dir directory object to the user jward.

Example 4-10 Granting the EXECUTE Privilege on a Directory Object

GRANT EXECUTE ON DIRECTORY exec dir TO jward;

4.15.1.4 Use of the ADMIN Option to Enable Grantee Users to Grant the Privilege

The WITH ADMIN OPTION clause can be used to expand the capabilities of a privilege grant.

These capabilities are as follows:

- The grantee can grant or revoke the system privilege or role to or from any other user or role in the database. Users cannot revoke a role from themselves.
- The grantee can grant the system privilege or role with the ADMIN option.
- The grantee of a role can alter or drop the role.

Example 4-11 grants the new dba role with the WITH ADMIN OPTION clause to user michael.

Example 4-11 Granting the ADMIN Option

GRANT new dba TO michael WITH ADMIN OPTION;

User michael is able to not only use all of the privileges implicit in the <code>new_dba</code> role, but he can also grant, revoke, and drop the <code>new_dba</code> role as deemed necessary. Because of these powerful capabilities, use caution when granting system privileges or roles with the <code>ADMIN</code> option. These privileges are usually reserved for a security administrator, and are rarely granted to other administrators or users of the system. Be aware that when a user creates a role, the role is automatically granted to the creator with the <code>ADMIN</code> option.

4.15.1.5 Creating a New User with the GRANT Statement

You can create a new user and grant this user a privilege in one GRANT SQL statement.

In most cases, you will want to grant the user the CREATE SESSION privilege.

 To create a new user with the GRANT statement, include the privilege and the IDENTIFIED BY clause.

For example, to create user psmith as a new user while granting psmith the CREATE SESSION system privilege:

GRANT CREATE SESSION TO psmith IDENTIFIED BY password;

If you specify a password using the IDENTIFIED BY clause, and the user name does not exist in the database, then a new user with that user name and password is created.

Related Topics

Creating User Accounts

A user account can have restrictions such as profiles, a default role, and tablespace restrictions.

Minimum Requirements for Passwords
 Oracle provides a set of minimum requirements for passwords.

4.15.2 Granting Object Privileges to Users and Roles

You can grant object privileges to users and roles, and enable the grantee to grant the privilege to other users.

- About Granting Object Privileges to Users and Roles
 You can use the GRANT statement to grant object privileges to roles and users.
- How the WITH GRANT OPTION Clause Works

The WITH GRANT OPTION clause with the GRANT statement can enable a grantee to grant object privileges to other users.

· Grants of Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege enables users to grant and revoke any object privilege on behalf of the object owner.

Grants of Privileges on Columns

You can grant INSERT, UPDATE, or REFERENCES privileges on individual columns in a table.

Row-Level Access Control

You can provide access control at the row level, that is, within objects, but not with the GRANT statement.

4.15.2.1 About Granting Object Privileges to Users and Roles

You can use the GRANT statement to grant object privileges to roles and users.

To grant an object privilege, you must fulfill one of the following conditions:

- You own the object specified.
- You have been granted the GRANT ANY OBJECT PRIVILEGE system privilege. This privilege enables you to grant and revoke privileges on behalf of the object owner.
- The WITH GRANT OPTION clause was specified when you were granted the object privilege.





System privileges and roles cannot be granted along with object privileges in the same GRANT statement.

The following example grants the READ, INSERT, and DELETE object privileges for all columns of the emp table to the users jfee and tsmith.

GRANT READ, INSERT, DELETE ON emp TO jfee, tsmith;

To grant all object privileges on the salary view to user jfee, use the ALL keyword as shown in the following example:

GRANT ALL ON salary TO jfee;



A grantee cannot regrant access to objects unless the original grant included the $\mbox{\tt GRANT}$ OPTION. Thus in the example just given, jfee cannot use the $\mbox{\tt GRANT}$ statement to grant object privileges to anyone else.

4.15.2.2 How the WITH GRANT OPTION Clause Works

The WITH GRANT OPTION clause with the GRANT statement can enable a grantee to grant object privileges to other users.

The user whose schema contains an object is automatically granted all associated object privileges with the WITH GRANT OPTION clause. This special privilege allows the grantee several expanded privileges:

- The grantee can grant the object privilege to any user in the database, with or without the GRANT OPTION, and to any role in the database.
- If both of the following conditions are true, then the grantee can create views on the table, and grant the corresponding privileges on the views to any user or role in the database:
 - The grantee receives object privileges for the table with the GRANT OPTION.
 - The grantee has the CREATE VIEW or CREATE ANY VIEW system privilege.



The WITH GRANT OPTION clause is not valid if you try to grant an object privilege to a role. Oracle Database prevents the propagation of object privileges through roles so that grantees of a role cannot propagate object privileges received by means of roles.

4.15.2.3 Grants of Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege enables users to grant and revoke any object privilege on behalf of the object owner.

This privilege provides a convenient means for database and application administrators to grant access to objects in any schema without requiring that they connect to the schema. Login credentials do not need to be maintained for schema owners who have this privilege, which reduces the number of connections required during configuration.

This system privilege is part of the Oracle Database supplied DBA role and is thus granted (with the ADMIN option) to any user connecting AS SYSDBA (user SYS). As with other system privileges, the GRANT ANY OBJECT PRIVILEGE system privilege can only be granted by a user who possesses the ADMIN option.

The *recorded* grantor of access rights to an object is either the object owner or the person exercising the GRANT ANY OBJECT PRIVILEGE system privilege. If the grantor with GRANT ANY OBJECT PRIVILEGE does *not* have the object privilege with the GRANT OPTION, then the object owner is shown as the grantor. Otherwise, when that grantor has the object privilege with the GRANT OPTION, then that grantor is recorded as the grantor of the grant.



The audit record generated by the ${\tt GRANT}$ statement always shows the actual user who performed the grant.

For example, consider the following scenario. User adams possesses the GRANT ANY OBJECT PRIVILEGE system privilege. He does not possess any other grant privileges. He issues the following statement:

```
GRANT SELECT ON HR.EMPLOYEES TO blake WITH GRANT OPTION;
```

If you examine the DBA_TAB_PRIVS view, then you will see that hr is shown as the grantor of the privilege:

Now assume that user blake also has the GRANT ANY OBJECT PRIVILEGE system. He issues the following statement:

```
GRANT SELECT ON HR.EMPLOYEES TO clark;
```

In this case, when you query the DBA_TAB_PRIVS view again, you see that blake is shown as being the grantor of the privilege:

GRANTEE	GRANTOR	PRIVILEGE	GRANTABLE
BLAKE	HR	SELECT	YES
CLARK	BLAKE	SELECT	NO

This occurs because blake already possesses the SELECT privilege on HR.EMPLOYEES with the GRANT OPTION.



Related Topics

• Revokes of Object Privileges on Behalf of the Object Owner
The GRANT ANY OBJECT PRIVILEGE system privilege can be used to revoke any object privilege where the object owner is the grantor.

4.15.2.4 Grants of Privileges on Columns

You can grant INSERT, UPDATE, or REFERENCES privileges on individual columns in a table.



Before granting a column-specific <code>INSERT</code> privilege, determine if the table contains any columns on which <code>NOT NULL</code> constraints are defined. Granting selective insert capability without including the <code>NOT NULL</code> columns prevents the user from inserting any rows into the table. To avoid this situation, ensure that each <code>NOT NULL</code> column can either be inserted into or has a non-<code>NULL</code> default value. Otherwise, the grantee will not be able to insert rows into the table and will receive an error.

The following statement grants the INSERT privilege on the acct_no column of the accounts table to user psmith:

```
GRANT INSERT (acct no) ON accounts TO psmith;
```

In the following example, object privilege for the ename and job columns of the emp table are granted to the users jfee and tsmith:

```
GRANT INSERT (ename, job) ON emp TO jfee, tsmith;
```

You can grant the INSERT and UPDATE privileges on individual columns of a view.

4.15.2.5 Row-Level Access Control

You can provide access control at the row level, that is, within objects, but not with the GRANT statement.

To perform this kind of access control, you must use either Oracle Virtual Private Database (VPD) or Oracle Label Security (OLS).

See Also:

- Using Oracle Virtual Private Database to Control Data Access
- Policies for Column-Level Oracle Virtual Private Database
- Oracle Label Security Administrator's Guide

4.16 Revokes of Privileges and Roles from a User

When you revoke system or object privileges, be aware of the cascading effects of revoking a privilege.

Revokes of System Privileges and Roles

The REVOKE SQL statement revokes system privileges and roles.

Revokes of Object Privileges

You can revoke multiple object privileges, object privileges on behalf of an object owner, column-selective object privileges, and the REFERENCES object privilege.

Cascading Effects of Revoking Privileges

There are no cascading effects for revoked object privileges related to DDL operations, but there are cascading effects for object privilege revocations.

4.16.1 Revokes of System Privileges and Roles

The REVOKE SQL statement revokes system privileges and roles.

Any user with the ADMIN option for a system privilege or role can revoke the privilege or role from any other database user or role. The revoker does not have to be the user that originally granted the privilege or role. Users with GRANT ANY ROLE can revoke *any* role.

Example 4-12 revokes the CREATE TABLE system privilege and the accts_rec role from user psmith:

Example 4-12 Revoking a System Privilege and a Role from a User

REVOKE CREATE TABLE, accts rec FROM psmith;

Be aware that the ADMIN option for a system privilege or role cannot be selectively revoked. Instead, revoke the privilege or role, and then grant the privilege or role again but without the ADMIN option.

4.16.2 Revokes of Object Privileges

You can revoke multiple object privileges, object privileges on behalf of an object owner, column-selective object privileges, and the REFERENCES object privilege.

About Revokes of Object Privileges

To revoke an object privilege, you must meet the appropriate requirements.

Revokes of Multiple Object Privileges

The REVOKE statement can revoke multiple privileges on one object.

· Revokes of Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege can be used to revoke any object privilege where the object owner is the grantor.

Revokes of Column-Selective Object Privileges

GRANT and REVOKE operations for column-specific operations have different privileges and restrictions.

Revokes of the REFERENCES Object Privilege

When you revoke the REFERENCES object privilege, it affects foreign key constraints.

4.16.2.1 About Revokes of Object Privileges

To revoke an object privilege, you must meet the appropriate requirements.

The requirements are either of the following conditions:

You previously granted the object privilege to the user or role.



 You possess the GRANT ANY OBJECT PRIVILEGE system privilege that enables you to grant and revoke privileges on behalf of the object owner.

You can only revoke the privileges that you, the person who granted the privilege, directly authorized. You cannot revoke grants that were made by other users to whom you granted the GRANT OPTION. However, there is a cascading effect. If the object privileges of the user who granted the privilege are revoked, then the object privilege grants that were propagated using the GRANT OPTION are revoked as well.

4.16.2.2 Revokes of Multiple Object Privileges

The REVOKE statement can revoke multiple privileges on one object.

Assuming you are the original grantor of the privilege, the following statement revokes the SELECT and INSERT privileges on the emp table from users jfee and psmith:

```
REVOKE SELECT, INSERT ON emp FROM jfee, psmith;
```

The following statement revokes all object privileges for the dept table that you originally granted to the human resource role:

REVOKE ALL ON dept FROM human resources;



The GRANT OPTION for an object privilege cannot be selectively revoked. Instead, revoke the object privilege and then grant it again but without the GRANT OPTION. Users cannot revoke object privileges from themselves.

4.16.2.3 Revokes of Object Privileges on Behalf of the Object Owner

The GRANT ANY OBJECT PRIVILEGE system privilege can be used to revoke any object privilege where the object owner is the grantor.

This occurs when the object privilege is granted by the object owner, or on behalf of the owner by any user holding the GRANT ANY OBJECT PRIVILEGE system privilege.

In a situation where the object privilege was granted by both the owner of the object and the user executing the REVOKE statement (who has both the specific object privilege and the GRANT OBJECT PRIVILEGE system privilege), Oracle Database only revokes the object privilege granted by the user issuing the REVOKE statement. This can be illustrated by continuing the example started in Grants of Object Privileges on Behalf of the Object Owner.

At this point, user blake granted the SELECT privilege on HR.EMPLOYEES to clark. Even though blake possesses the GRANT ANY OBJECT PRIVILEGE system privilege, he also holds the specific object privilege, thus this grant is attributed to him. Assume that user HR also grants the SELECT privilege on HR.EMPLOYEES to user clark. A query of the DBA_TAB_PRIVS view shows that the following grants are in effect for the HR.EMPLOYEES table:

GRANTEE	GRANTOR	PRIVILEGE	GRANTABLE
BLAKE	HR	SELECT	YES
CLARK	BLAKE	SELECT	NO
CLARK	HR	SELECT	NO



User blake now issues the following REVOKE statement:

```
REVOKE SELECT ON HR.EMPLOYEES FROM clark;
```

Only the object privilege for user clark granted by user blake is removed. The grant by the object owner, HR, remains.

GRANTEE	GRANTOR	PRIVILEGE	GRANTABLE
BLAKE	HR	SELECT	YES
CLARK	HR	SELECT	NO

If blake issues the REVOKE statement again, then this time the effect is to remove the object privilege granted by adams (on behalf of HR), using the GRANT ANY OBEJCT PRIVILEGE system privilege.

Related Topics

Grants of Object Privileges on Behalf of the Object Owner The GRANT ANY OBJECT PRIVILEGE system privilege enables users to grant and revoke any object privilege on behalf of the object owner.

4.16.2.4 Revokes of Column-Selective Object Privileges

GRANT and REVOKE operations for column-specific operations have different privileges and restrictions.

Although users can grant column-specific INSERT, UPDATE, and REFERENCES privileges for tables and views, they cannot selectively revoke column-specific privileges with a similar REVOKE statement. Instead, the grantor must first revoke the object privilege for all columns of a table or view, and then selectively repeat the grant of the column-specific privileges that the grantor intends to keep in effect.

For example, assume that role human_resources was granted the UPDATE privilege on the deptno and dname columns of the table dept. To revoke the UPDATE privilege on just the deptno column, issue the following two statements:

```
REVOKE UPDATE ON dept FROM human_resources;
GRANT UPDATE (dname) ON dept TO human resources;
```

The REVOKE statement revokes the UPDATE privilege on all columns of the dept table from the role human_resources. The GRANT statement then repeats, restores, or reissues the grant of the UPDATE privilege on the dname column to the role human resources.

4.16.2.5 Revokes of the REFERENCES Object Privilege

When you revoke the REFERENCES object privilege, it affects foreign key constraints.

If the grantee of the REFERENCES object privilege has used the privilege to create a foreign key constraint (that currently exists), then the grantor can revoke the privilege only by specifying the CASCADE CONSTRAINTS option in the REVOKE statement.

For example:

```
REVOKE REFERENCES ON dept FROM jward CASCADE CONSTRAINTS;
```

Any foreign key constraints currently defined that use the revoked REFERENCES privilege are dropped when the CASCADE CONSTRAINTS clause is specified.

4.16.3 Cascading Effects of Revoking Privileges

There are no cascading effects for revoked object privileges related to DDL operations, but there are cascading effects for object privilege revocations.

- Cascading Effects When Revoking System Privileges
 There are no cascading effects when you revoke a system privilege that is related to DDL operations.
- Cascading Effects When Revoking Object Privileges
 Revoking an object privilege can have cascading effects.

4.16.3.1 Cascading Effects When Revoking System Privileges

There are no cascading effects when you revoke a system privilege that is related to DDL operations.

This applies regardless of whether the privilege was granted with or without the ADMIN option.

For example, assume the following:

- 1. The security administrator grants the CREATE TABLE system privilege to user jfee with the ADMIN option.
- 2. User jfee creates a table.
- 3. User jfee grants the CREATE TABLE system privilege to user tsmith.
- 4. User tsmith creates a table.
- The security administrator revokes the CREATE TABLE system privilege from user jfee.
- 6. The table created by user jfee continues to exist. User tsmith still has the table and the CREATE TABLE system privilege.

You can observe cascading effects when you revoke a system privilege related to a DML operation. If the SELECT ANY TABLE privilege is revoked from a user, then all procedures contained in the user's schema relying on this privilege can no longer be executed successfully until the privilege is reauthorized.

4.16.3.2 Cascading Effects When Revoking Object Privileges

Revoking an object privilege can have cascading effects.

Note the following:

- Object definitions that depend on a DML object privilege can be affected if the DML object privilege is revoked. For example, assume that the body of the test procedure includes a SQL statement that queries data from the emp table. If the SELECT privilege on the emp table is revoked from the owner of the test procedure, then the procedure can no longer be executed successfully.
- When a REFERENCES privilege for a table is revoked from a user, any foreign key integrity constraints that are defined by the user and require the dropped REFERENCES privilege are automatically dropped. For example, assume that user jward is granted the REFERENCES privilege for the deptno column of the dept table. This user now creates a foreign key on the deptno column in the emp table that references the deptno column of the dept table. If the REFERENCES privilege on the deptno column of the



dept table is revoked, then the foreign key constraint on the deptno column of the emp table is dropped in the same operation.

• The object privilege grants propagated using the GRANT OPTION are revoked if the object privilege of a grantor is revoked. For example, assume that user1 is granted the SELECT object privilege on the emp table with the GRANT OPTION, and grants the SELECT privilege on emp to user2. Subsequently, the SELECT privilege is revoked from user1. This REVOKE statement is also cascaded to user2. Any objects that depend on the revoked SELECT privilege of user1 and user2 can also be affected, as described earlier.

Object definitions that require the ALTER and INDEX DDL object privileges are not affected if the ALTER or INDEX object privilege is revoked. For example, if the INDEX privilege is revoked from a user that created an index on a table that belongs to another user, then the index continues to exist after the privilege is revoked.

4.17 Grants and Revokes of Privileges to and from the PUBLIC Role

You can grant and revoke privileges and roles from the role PUBLIC.

Because PUBLIC is accessible to every database user, all privileges and roles granted to PUBLIC are accessible to every database user. By default, PUBLIC does not have privileges granted to it.

Security administrators and database users should grant a privilege or role to PUBLIC only if every database user requires the privilege or role. This recommendation reinforces the general rule that, at any given time, each database user should have only the privileges required to accomplish the current group tasks successfully.

Revoking a privilege from the PUBLIC role can cause significant cascading effects. If any privilege related to a DML operation is revoked from PUBLIC (for example, SELECT ANY TABLE OR UPDATE ON emp), then all procedures in the database, including functions and packages, must be *reauthorized* before they can be used again. Therefore, be careful when you grant and revoke DML-related privileges to or from PUBLIC.

See Also:

- Oracle Database Administrator's Guide for more information about managing object dependencies
- Guidelines for Securing Data

4.18 Grants of Roles Using the Operating System or Network

Using the operating system or network to manage roles can help centralize the role management in a large enterprise.

About Granting Roles Using the Operating System or Network
 The operating system on which Oracle Database runs can be used to grant roles to users
 at connect time.

Operating System Role Identification

The OS_ROLES initialization parameter can be used to control how the operating system identifies roles.

Operating System Role Management

When you use operating system-managed roles, remember that database roles are being granted to an operating system user.

- Role Grants and Revokes When OS_ROLES Is Set to TRUE
 Setting the OS_ROLES initialization parameter to TRUE enables the operating system to manage role grants and revokes to users.
- Role Enablements and Disablements When OS_ROLES Is Set to TRUE
 Setting the OS_ROLES initialization parameter to TRUE enables the SET ROLE statement to dynamically enable roles granted by the operating system.
- Network Connections with Operating System Role Management
 By default, users cannot connect to the database through a shared server if the operating system manages roles.

4.18.1 About Granting Roles Using the Operating System or Network

The operating system on which Oracle Database runs can be used to grant roles to users at connect time.

This feature is an alternative to a security administrator explicitly having to granting and revoking database roles to and from users using GRANT and REVOKE statements.

Roles can be administered using the operating system and passed to Oracle Database when a user creates a session. As part of this mechanism, the default roles of a user and the roles granted to a user with the ADMIN option can be identified. If the operating system is used to authorize users for roles, then all roles must be created in the database and privileges assigned to the role with GRANT statements.

Roles can also be granted through a network service.

The advantage of using the operating system to identify the database roles of a user is that privilege management for an Oracle database can be externalized. The security facilities offered by the operating system control user privileges. This option may offer advantages of centralizing security for several system activities, such as the following situation:

- MVS Oracle administrators want RACF groups to identify database user roles.
- UNIX Oracle administrators want UNIX groups to identify database user roles.
- VMS Oracle administrators want to use rights identifiers to identify database user roles.

The main disadvantage of using the operating system to identify the database roles of a user is that privilege management can only be performed at the role level. Individual privileges cannot be granted using the operating system, but they can still be granted inside the database using GRANT statements.

A second disadvantage of using this feature is that, by default, users cannot connect to the database through the shared server or any other network connection if the operating system is managing roles. However, you can change this default as described in Network Connections with Operating System Role Management.

In a multitenant environment, you can use operating system authentication for a database administrator only for the CDB root. You cannot use it for PDBs, the application root, or application PDBs.



The features described in this section are available only on some operating systems. See your operating system-specific Oracle Database documentation to determine if you can use these features.

4.18.2 Operating System Role Identification

The OS_ROLES initialization parameter can be used to control how the operating system identifies roles.

To have the database use the operating system to identify the database roles of each user when a session is created, you can set the initialization parameter OS ROLES to TRUE.

If the instance is current running, you must restart the instance. When a user tries to create a session with the database, Oracle Database initializes the user security domain using the database roles identified by the operating system.

To identify database roles for a user, the operating system account for each Oracle Database user must have operating system identifiers (these may be called groups, rights identifiers, or other similar names) that indicate which database roles are to be available for the user. Role specification can also indicate which roles are the default roles of a user and which roles are available with the ADMIN option. No matter which operating system is used, the role specification at the operating system level follows the format:

```
ora_ID_ROLE[[_d][_a][_da]]
```

In this specification:

• ID has a definition that varies on different operating systems. For example, on VMS, ID is the instance identifier of the database; on VMS, it is the computer type; and on UNIX, it is the system ID.

ID is case-sensitive to match your ORACLE SID. ROLE is not case-sensitive.

- ROLE is the name of the database role.
- d is an optional character that indicates this role is to be a default role of the database user.
- a is an optional character that indicates this role is to be granted to the user with the ADMIN option. This allows the user to grant the role to other roles only. Roles cannot be granted to users if the operating system is used to manage roles.

If either the d or a character is specified, then precede that character by an underscore (_).

For example, suppose an operating system account has the following roles identified in its profile:

```
ora_PAYROLL_ROLE1
ora_PAYROLL_ROLE2_a
ora_PAYROLL_ROLE3_d
ora_PAYROLL_ROLE4_da
```

When the corresponding user connects to the payroll instance of Oracle Database, role3 and role4 are defaults, while role2 and role4 are available with the ADMIN option.

4.18.3 Operating System Role Management

When you use operating system-managed roles, remember that database roles are being granted to an operating system user.

Any database user to which the operating system user is able to connect will have the authorized database roles enabled. For this reason, you should consider defining all Oracle Database users as <code>IDENTIFIED EXTERNALLY</code> if you are using <code>OS_ROLES = TRUE</code>, so that the database accounts are tied to the operating system account that was granted privileges.

4.18.4 Role Grants and Revokes When OS_ROLES Is Set to TRUE

Setting the OS_ROLES initialization parameter to TRUE enables the operating system to manage role grants and revokes to users.

Any previous granting of roles to users using GRANT statements do not apply. However, they are still listed in the data dictionary. Only the role grants to users made at the operating system level apply. Users can still grant privileges to roles and users.



If the operating system grants a role to a user with the ${\tt ADMIN}$ option, then the user can grant the role only to other roles.

4.18.5 Role Enablements and Disablements When OS_ROLES Is Set to TRUE

Setting the <code>OS_ROLES</code> initialization parameter to <code>TRUE</code> enables the <code>SET_ROLE</code> statement to dynamically enable roles granted by the operating system.

This still applies, even if the role was defined to require a password or operating system authorization. However, any role not identified in the operating system account of a user cannot be specified in a SET ROLE statement, even if a role was granted using a GRANT statement when OS_ROLES = FALSE. (If you specify such a role, then Oracle Database ignores it.)

When OS_ROLES is set to TRUE, then the user can enable up to 148 roles. Remember that this number includes other roles that may have been granted to the role.

4.18.6 Network Connections with Operating System Role Management

By default, users cannot connect to the database through a shared server if the operating system manages roles.

This restriction is the default because a remote user could impersonate another operating system user over an unsecure connection.

If you are not concerned with this security risk and want to use operating system role management with the shared server, or any other network connection, then set the initialization parameter <code>REMOTE_OS_ROLES</code> to <code>TRUE</code>. The change takes effect the next time you start the instance and mount the database. The default setting of this parameter is <code>FALSE</code>.



4.19 How Grants and Revokes Work with SET ROLE and Default Role Settings

Privilege grants and the SET ROLE statement affect when and how grants and revokes take place.

- When Grants and Revokes Take Effect
 - Depending on the privilege that is granted or revoked, a grant or revoke takes effect at different times.
- How the SET ROLE Statement Affects Grants and Revokes
 During a user session, a user or an application can use the SET ROLE statement multiple times to change the roles enabled for the session.
- Specifying the Default Role for a User
 When a user logs on, Oracle Database enables all privileges granted explicitly to the user and all privileges in the user's default roles.
- The Maximum Number of Roles That a User Can Have Enabled
 You can grant a user as many roles as you want, but no more than 148 roles can be enabled for a logged-in user at any given time.

4.19.1 When Grants and Revokes Take Effect

Depending on the privilege that is granted or revoked, a grant or revoke takes effect at different times.

The grants and revokes take effect as follows:

- All grants and revokes of system and object privileges to anything (users, roles, and PUBLIC) take immediate effect.
- All grants and revokes of roles to anything (users, other roles, PUBLIC) take effect only when a current user session issues a SET ROLE statement to reenable the role after the grant and revoke, or when a new user session is created after the grant or revoke.

You can see which roles are currently enabled by examining the <code>SESSION_ROLES</code> data dictionary view.

4.19.2 How the SET ROLE Statement Affects Grants and Revokes

During a user session, a user or an application can use the SET ROLE statement multiple times to change the roles enabled for the session.

The user must already be granted the roles that are named in the SET ROLE statement.

The following example enables the role clerk, which you have already been granted, and specifies the password.

SET ROLE clerk IDENTIFIED BY password;

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

The following example shows how to use **SET ROLE** to disable all roles.

SET ROLE NONE;



4.19.3 Specifying the Default Role for a User

When a user logs on, Oracle Database enables all privileges granted explicitly to the user and all privileges in the user's default roles.

- Ensure that the user who you want to set the default role for has been directly granted the role with a GRANT statement, or that the role was created by the user with the CREATE ROLE privilege.
- 2. Use the ALTER USER statement with the DEFAULT ROLE clause to specify the default role for the user.

For example, to set the default roles payclerk and pettycash for user jane:

ALTER USER jane DEFAULT ROLE payclerk, pettycash;

For information about the restrictions of the DEFAULT ROLE clause of the ALTER USER statement, see *Oracle Database SQL Language Reference*.

You cannot set default roles for a user in the CREATE USER statement. When you first create a user, the default user role setting is ALL, which causes all roles subsequently granted to the user to be default roles. Use the ALTER USER statement to limit the default user roles.

Note:

When you create a role (other than a global role or an application role), it is granted implicitly to you, and your set of default roles is updated to include the new role. Be aware that only 148 roles can be enabled for a user session. When aggregate roles, such as the DBA role, are granted to a user, the roles granted to the role are included in the number of roles the user has. For example, if a role has 20 roles granted to it and you grant that role to the user, then the user now has 21 additional roles. Therefore, when you grant new roles to a user, use the DEFAULT ROLE clause of the ALTER USER statement to ensure that not too many roles are specified as that user's default roles.

4.19.4 The Maximum Number of Roles That a User Can Have Enabled

You can grant a user as many roles as you want, but no more than 148 roles can be enabled for a logged-in user at any given time.

Therefore, not all privileges will be available to this user during the user session. As a best practice, restrict the number of roles granted to a user to the minimum roles the user needs.

Related Topics

Guidelines for Securing Roles
 Oracle provides guidelines for role management.

4.20 User Privilege and Role Data Dictionary Views

You can use special queries to find information about various types of privilege and role grants.

Data Dictionary Views to Find Information about Privilege and Role Grants
 Oracle Database provides data dictionary views that describe privilege and role grants.

Query to List All System Privilege Grants

The DBA_SYS_PRIVS data dictionary view returns all system privilege grants made to roles and users.

Query to List All Role Grants

The DBA ROLE PRIVS query returns all the roles granted to users and other roles.

· Query to List Object Privileges Granted to a User

The DBA_TAB_PRIVS and DBA_COL_PRIVS data dictionary views list object privileges that have bee granted to users.

Query to List the Current Privilege Domain of Your Session

The SESSION_ROLES and SESSION_PRIVS data dictionary views list the current privilege domain of a database session.

Query to List Roles of the Database

The DBA_ROLES data dictionary view lists all roles of a database and the authentication used for each role.

Query to List Information About the Privilege Domains of Roles

The ROLE_ROLE_PRIVS, ROLE_SYS_PRIVS, and ROLE_TAB_PRIVS data dictionary views list information about the privilege domains of roles.

4.20.1 Data Dictionary Views to Find Information about Privilege and Role Grants

Oracle Database provides data dictionary views that describe privilege and role grants.

The following table lists views that you can query to access information about grants of privileges and roles.

Table 4-6 Data Dictionary Views That Display Privilege and Role Information

View	Description
ALL_COL_PRIVS	Describes all column object grants for which the current user or PUBLIC is the object owner, grantor, or grantee
ALL_COL_PRIVS_MADE	Lists column object grants for which the current user is object owner or grantor
ALL_COL_PRIVS_RECD	Describes column object grants for which the current user or PUBLIC is the grantee
ALL_TAB_PRIVS	Lists the grants on objects where the user or ${\tt PUBLIC}$ is the grantee
ALL_TAB_PRIVS_MADE	Lists the all object grants made by the current user or made on the objects owned by the current user
ALL_TAB_PRIVS_RECD	Lists object grants for which the user or PUBLIC is the grantee
DBA_COL_PRIVS	Describes all column object grants in the database
DBA_CONTAINER_DATA	In a multitenant environment, displays default (user-level) and object-specific CONTAINER_DATA attributes. Objects that are created with the CONTAINER_DATA clause include CONTAINER_DATA attributes.
DBA_EPG_DAD_AUTHORIZATION	Describes the database access descriptors (DAD) that are authorized to use a different user's privileges
DBA_LOCKDOWN_PROFILES	Describes information that pertains to PDB lockdown profiles



Table 4-6 (Cont.) Data Dictionary Views That Display Privilege and Role Information

View	Description
DBA_OBJECTS	Lists objects that have object links or metadata links. To find these objects, query the <code>OBJECT_NAME</code> and <code>SHARING</code> columns.
DBA_TAB_PRIVS	Lists all grants on all objects in the database
DBA_ROLES	Lists all roles that exist in the database, including secure application roles. Note that it does not list the PUBLIC role
DBA_ROLE_PRIVS	Lists roles directly granted to users and roles
DBA_SYS_PRIVS	Lists system privileges granted to users and roles
ROLE_ROLE_PRIVS	Lists roles granted to other roles. Information is provided only about roles to which the user has access
ROLE_SYS_PRIVS	Lists system privileges granted to roles. Information is provided only about roles to which the user has access
ROLE_TAB_PRIVS	Lists object privileges granted to roles. Information is provided only about roles to which the user has access
SESSION_PRIVS	Lists the privileges that are currently enabled for the user
SESSION_ROLES	Lists all roles that are enabled for the current user. Note that it does not list the ${\tt PUBLIC}$ role
USER_COL_PRIVS	Describes column object grants for which the current user is the object owner, grantor, or grantee
USER_COL_PRIVS_MADE	Describes column object grants for which the current user is the object owner
USER_COL_PRIVS_RECD	Describes column object grants for which the current user is the grantee
USER_EPG_DAD_AUTHORIZATION	Describes the database access descriptors (DAD) that are authorized to use a different user's privileges
USER_ROLE_PRIVS	Lists roles directly granted to the current user
USER_TAB_PRIVS	Lists grants on all objects where the current user is the grantee
USER_SYS_PRIVS	Lists system privileges granted to the current user
USER_TAB_PRIVS_MADE	Lists grants on all objects owned by the current user
USER_TAB_PRIVS_RECD	Lists object grants for which the current user is the grantee
V\$PWFILE_USERS	Lists all users in the current PDB who have been granted administrative privileges

The following table lists views that you can query to access information about grants of privileges and roles.

This section provides some examples of using these views. For these examples, assume the following statements were issued:

```
CREATE ROLE security_admin IDENTIFIED BY password;

GRANT CREATE PROFILE, ALTER PROFILE, DROP PROFILE,

CREATE ROLE, DROP ANY ROLE, GRANT ANY ROLE, AUDIT ANY,

AUDIT SYSTEM, CREATE USER, BECOME USER, ALTER USER, DROP USER

TO security_admin WITH ADMIN OPTION;

GRANT READ, DELETE ON SYS.AUD$ TO security admin;
```



```
GRANT security_admin, CREATE SESSION TO swilliams;

GRANT security_admin TO system_administrator;

GRANT CREATE SESSION TO jward;

GRANT READ, DELETE ON emp TO jward;

GRANT INSERT (ename, job) ON emp TO swilliams, jward;
```

Related Topics

Oracle Database Reference

4.20.2 Query to List All System Privilege Grants

The DBA_SYS_PRIVS data dictionary view returns all system privilege grants made to roles and users.

For example:

SELECT GRANTEE, PRIVILEGE, ADM FROM DBA_SYS_PRIVS;

GRANTEE	PRIVILEGE	ADM
SECURITY_ADMIN	ALTER PROFILE	YES
SECURITY_ADMIN	ALTER USER	YES
SECURITY_ADMIN	AUDIT ANY	YES
SECURITY_ADMIN	AUDIT SYSTEM	YES
SECURITY_ADMIN	BECOME USER	YES
SECURITY_ADMIN	CREATE PROFILE	YES
SECURITY_ADMIN	CREATE ROLE	YES
SECURITY_ADMIN	CREATE USER	YES
SECURITY_ADMIN	DROP ANY ROLE	YES
SECURITY_ADMIN	DROP PROFILE	YES
SECURITY_ADMIN	DROP USER	YES
SECURITY_ADMIN	GRANT ANY ROLE	YES
SWILLIAMS	CREATE SESSION	NO
JWARD	CREATE SESSION	NO



Oracle Database Reference for detailed information about the DBA SYS PRIVS view

4.20.3 Query to List All Role Grants

The DBA ROLE PRIVS query returns all the roles granted to users and other roles.

For example:

SELECT * FROM DBA ROLE PRIVS;

GRANTEE	GRANTED_	ROLE	ADM
SWILLIAMS	SECURITY	ADMIN	NO





Oracle Database Reference for detailed information about the DBA ROLE PRIVS view

4.20.4 Query to List Object Privileges Granted to a User

The DBA_TAB_PRIVS and DBA_COL_PRIVS data dictionary views list object privileges that have bee granted to users.

The DBA_TAB_PRIVS data dictionary view returns all object privileges (not including column-specific privileges) granted to the specified user.

For example:

```
SELECT TABLE_NAME, PRIVILEGE, GRANTABLE FROM DBA_TAB_PRIVS
WHERE GRANTEE = 'jward';
```

TABLE_NAME	PRIVILEGE	GRANTABLE
EMP	SELECT	NO
EMP	DELETE	NO

To list all the column-specific privileges that have been granted, you can use the following query:

```
SELECT GRANTEE, TABLE_NAME, COLUMN_NAME, PRIVILEGE
FROM DBA_COL_PRIVS;
```

GRANTEE	TABLE_NAME	COLUMN_NAME	PRIVILEGE
SWILLIAMS	EMP	ENAME	INSERT
SWILLIAMS	EMP	JOB	INSERT
JWARD	EMP	NAME	INSERT
JWARD	EMP	JOB	INSERT



Oracle Database Reference for detailed information about the DBA TAB PRIVS view

4.20.5 Query to List the Current Privilege Domain of Your Session

The SESSION_ROLES and SESSION_PRIVS data dictionary views list the current privilege domain of a database session.

The **SESSION** ROLES view lists all roles currently enabled for the issuer.

For example:

```
SELECT * FROM SESSION ROLES;
```

If user swilliams has the security_admin role enabled and issues the previous query, then Oracle Database returns the following information:

```
ROLE
-----SECURITY_ADMIN
```

The following query lists all system privileges currently available in the security domain of the issuer, both from explicit privilege grants and from enabled roles:

```
SELECT * FROM SESSION PRIVS;
```

If user swilliams has the security_admin role enabled and issues the previous query, then Oracle Database returns the following results:

If the security_admin role is disabled for user swilliams, then the first query would return no rows, while the second query would only return a row for the CREATE SESSION privilege grant.



Oracle Database Reference for detailed information about the SESSION_ROLES view

4.20.6 Query to List Roles of the Database

The DBA_ROLES data dictionary view lists all roles of a database and the authentication used for each role.

For example:

SELECT * FROM DBA_ROLES;

ROLE	PASSWORD
CONNECT	NO
RESOURCE	NO
DBA	NO
SECURITY_ADMIN	YES



Oracle Database Reference for detailed information about the DBA ROLES view

4.20.7 Query to List Information About the Privilege Domains of Roles

The ROLE_PRIVS, ROLE_SYS_PRIVS, and ROLE_TAB_PRIVS data dictionary views list information about the privilege domains of roles.

For example:

```
SELECT GRANTED_ROLE, ADMIN_OPTION
FROM ROLE_ROLE_PRIVS
WHERE ROLE = 'SYSTEM_ADMIN';

GRANTED_ROLE ADM
-----
SECURITY_ADMIN NO
```

The following query lists all the system privileges granted to the security admin role:

```
SELECT * FROM ROLE_SYS_PRIVS WHERE ROLE = 'SECURITY_ADMIN';
```

ROLE	PRIVILEGE	ADM
SECURITY_ADMIN	ALTER PROFILE	YES
SECURITY_ADMIN	ALTER USER	YES
SECURITY_ADMIN	AUDIT ANY	YES
SECURITY_ADMIN	AUDIT SYSTEM	YES
SECURITY_ADMIN	BECOME USER	YES
SECURITY_ADMIN	CREATE PROFILE	YES
SECURITY_ADMIN	CREATE ROLE	YES
SECURITY_ADMIN	CREATE USER	YES
SECURITY_ADMIN	DROP ANY ROLE	YES
SECURITY ADMIN	DROP PROFILE	YES
SECURITY_ADMIN	DROP USER	YES
SECURITY_ADMIN	GRANT ANY ROLE	YES

The following query lists all the object privileges granted to the security admin role:

```
SELECT TABLE_NAME, PRIVILEGE FROM ROLE_TAB_PRIVS WHERE ROLE = 'SECURITY ADMIN';
```

TABLE_NAME	PRIVILEGE
AUD\$	DELETE
AUD\$	SELECT

Related Topics

Oracle Database Reference

Performing Privilege Analysis to Identify Privilege Use

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

- What Is Privilege Analysis?
 - Privilege analysis increases the security of your applications and database operations by helping you to implement least privilege best practices for database roles and privileges.
- Creating and Managing Privilege Analysis Policies
 You can create and manage privilege analysis policies by using tools such as SQL*Plus,
 SQL Developer, or Enterprise Manager Cloud Control.
- Creating Roles and Managing Privileges Using Cloud Control
 You can create new roles using privileges found in a privilege analysis report and then
 grant this role to users.
- Tutorial: Using Capture Runs to Analyze ANY Privilege Use
 This tutorial demonstrates how to create capture runs to analyze the use of the READ ANY TABLE system privilege.
- Tutorial: Analyzing Privilege Use by a User Who Has the DBA Role
 This tutorial demonstrates how to analyze the privilege use of a user who has the DBA role
 and performs database tuning operations.
- Privilege Analysis Policy and Report Data Dictionary Views
 Oracle Database provides data dictionary views that show information about analyzed privileges.

5.1 What Is Privilege Analysis?

Privilege analysis increases the security of your applications and database operations by helping you to implement least privilege best practices for database roles and privileges.

- About Privilege Analysis
 - Running inside the Oracle Database kernel, privilege analysis helps reduce the attack surface of user, tooling, and application accounts by identifying used and unused privileges to implement the least-privilege model.
- Benefits and Use Cases of Privilege Analysis
 Analyzing privilege use is beneficial in finding unnecessarily granted privileges and implementing least privilege best practices.
- Who Can Perform Privilege Analysis?
 To use privilege analysis, you must be granted the CAPTURE ADMIN role.
- Types of Privilege Analysis
 - You can create different types of privilege analysis policies to achieve specific goals.
- How Does a Multitenant Environment Affect Privilege Analysis?
 You can create and use privilege analysis policies in a multitenant environment.

How Privilege Analysis Works with Pre-Compiled Database Objects
 Privilege analysis can be used to capture the privileges that have been exercised on pre-compiled database objects.

5.1.1 About Privilege Analysis

Running inside the Oracle Database kernel, privilege analysis helps reduce the attack surface of user, tooling, and application accounts by identifying used and unused privileges to implement the least-privilege model.

Privilege analysis dynamically captures privileges used by database users and applications during a specified window of time.

The use of privilege analysis can help to quickly and efficiently enforce least privilege guidelines. In the least-privilege model, users are only given the privileges and access they need to do their jobs. Frequently, even though users perform different tasks, users are all granted the same set of powerful privileges. Without privilege analysis, figuring out the privileges that each user must have can be hard work and in many cases, users could end up with some common set of privileges even though they have different tasks. Even in organizations that manage privileges, users tend to accumulate privileges over time and rarely lose any privileges. Separation of duty breaks a single process into separate tasks for different users. Least privileges enforces the separation so users can only do their required tasks. The enforcement of separation of duty is beneficial for internal control, but it also reduces the risk from malicious users who steal privileged credentials.

Privilege analysis captures privileges used by database users and applications at runtime and writes its findings to data dictionary views that you can query. If your applications include definer's rights and invoker's rights procedures, then privilege analysis captures the privileges that are required to compile a procedure and execute it, even if the procedure was compiled before the privilege capture was created and enabled. Instead of revoking a privilege from the user, you can audit the user's use of the privilege and use an application such as Oracle Audit Vault and Database Firewall to send an alert to the appropriate administrator.

5.1.2 Benefits and Use Cases of Privilege Analysis

Analyzing privilege use is beneficial in finding unnecessarily granted privileges and implementing least privilege best practices.

- Least Privileges Best Practice
 - The privileges of the account that accesses a database should be limited to the privileges that are strictly required by the application or the user.
- Development of Secure Applications
 - During the application development phase, some administrators may grant many powerful system privileges and roles, and the SYSDBA administrative privilege, to application developers.

5.1.2.1 Least Privileges Best Practice

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

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

For example, to select from application data and run application procedures, the system privileges <code>select</code> any <code>table</code> and <code>execute</code> any <code>procedure</code> are granted to an application account <code>appsys</code>. The account <code>appsys</code> now can access non-application data even if he or she does not intend to. In this situation, you can analyze the privilege usage by user <code>appsys</code>, and then based on the results, revoke and grant privileges as necessary.

Application accounts also frequently have additional privileges needed to install and maintain the application on the database. These are only needed during application maintenance periods, but yet are available all the time. A better process would be to add the privileges needed for application maintenance into a separate role and grant that to the application only during maintenance periods.

5.1.2.2 Development of Secure Applications

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

The administrators may do this because at that stage they may not know what privileges the application developer needs or is not concerned with privileges and roles during development.

Once the application is developed and working, the privileges that the application developer needs — and does not need — become more apparent. Capturing privilege analysis while the application is run through a full regression test can capture most, if not all the privileges needed by the application for runtime use. Capturing privilege analysis when testing a maintenance update can provide the privileges needed during an update of the production system. At that time, the security administrator can begin to revoke unnecessary privileges. However, the application developer may resist this idea on the basis that the application is currently working without problems. The administrator can use privilege analysis to examine each privilege that the application uses, to ensure that when he or she does revoke any privileges, the application will continue to work.

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

5.1.3 Who Can Perform Privilege Analysis?

To use privilege analysis, you must be granted the CAPTURE ADMIN role.

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

5.1.4 Types of Privilege Analysis

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

Context-based privilege use capture. You must specify a Boolean expression only with
the SYS_CONTEXT function. The used privileges will be captured if the condition evaluates to
TRUE. This method can be used to capture privileges and roles used by a database user by
specifying the user in SYS CONTEXT.



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

Note the following restrictions:

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

5.1.5 How Does a Multitenant Environment Affect Privilege Analysis?

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

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

5.1.6 How Privilege Analysis Works with Pre-Compiled Database Objects

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

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

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



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

5.2 Creating and Managing Privilege Analysis Policies

You can create and manage privilege analysis policies by using tools such as SQL*Plus, SQLcl, SQL Developer, or Enterprise Manager Cloud Control.

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

5.2.1 About Creating and Managing Privilege Analysis Policies

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

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

Related Topics

Oracle Database PL/SQL Packages and Types Reference

5.2.2 General Steps for Managing Privilege Analysis

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

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

This step begins recording the privilege use that the policy defined. Optionally, specify a name for this capture run. Each time you enable a privilege analysis policy, you can create

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

This step stops capturing the privilege use for the policy.

5. Generate privilege analysis results.

This step writes the results to the privilege analysis policy and report data dictionary views.

Optionally, disable and then drop the privilege analysis policy and capture run.Dropping a privilege analysis policy deletes the data captured by the policy.

Related Topics

Privilege Analysis Policy and Report Data Dictionary Views
 Oracle Database provides data dictionary views that show information about analyzed privileges.

5.2.3 Creating a Privilege Analysis Policy

You can use the <code>DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE</code> procedure to create a privilege analysis policy.

After you create the privilege analysis policy, you can find it listed in the DBA_PRIV_CAPTURES data dictionary view. When a policy is created, it resides in the Oracle data dictionary and the SYS schema. However, both SYS and the user who created the policy can drop it. After you create the policy, you must manually enable it so that it can begin to analyze privilege use.

- Log in to the database instance as a user who has the CAPTURE ADMIN role.
- Use the following syntax for the DBMS PRIVILEGE CAPTURE.CREATE CAPTURE procedure:

In this specification:

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

Enter one of the following types:

 DBMS_PRIVILEGE_CAPTURE.G_DATABASE: Captures all privileges used in the entire database, except privileges from user SYS.

- DBMS_PRIVILEGE_CAPTURE.G_ROLE: Captures privileges for the sessions that have the roles enabled. If you enter DBMS_PRIVILEGE_CAPTURE.G_ROLE for the type parameter, then you must also specify the roles parameter. For multiple roles, separate each role name with a comma.
- DBMS_PRIVILEGE_CAPTURE.G_CONTEXT: Captures privileges for the sessions that have the condition specified by the condition parameter evaluating to TRUE. If you enter DBMS_PRIVILEGE_CAPTURE.G_CONTEXT for the type parameter, then you must also specify the condition parameter.
- DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT: Captures privileges for the sessions that have the role enabled and the context condition evaluating to TRUE. If you enter DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT for the type parameter, then you must also specify both the roles and condition parameters.
- roles: Specifies the roles whose used privileges will be analyzed. That is, if a privilege from one of the given roles is used, then the privilege will be analyzed. You must specify this argument if you specify DBMS_PRIVILEGE_CAPTURE.G_ROLE or DBMS_PRIVILEGE_CAPTURE.G_ROLE_AND_CONTEXT for the type argument. Each role you enter must exist in the database. (You can find existing roles by querying the DBA_ROLES data dictionary view.) For multiple roles, use varray type role_name_list to enter the role names. You can specify up to 10 roles.

For example, to specify two roles:

```
roles => role name list('role1', 'role2'),
```

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

The condition expression syntax is as follows:

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

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

After you create the privilege analysis policy, you must enable the policy to begin capturing privilege and role use.

* You can add as many constant values as you need (for example, IN {constant_value1}, or IN {constant value1, constant value2, constant value3}).

Related Topics

Enabling a Privilege Analysis Policy
 After you create a privilege analysis policy, you must enable it to capture privilege use.

5.2.4 Examples of Creating Privilege Analysis Policies

You can create a variety of privilege analysis policies.

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

5.2.4.1 Example: Privilege Analysis of Database-Wide Privileges

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

Example 5-1 shows how to use the DBMS_PRIVILEGE_CAPTURE package to create a privilege analysis policy to record all privilege use in the database.

Example 5-1 Privilege Analysis of Database-Wide Privileges

5.2.4.2 Example: Privilege Analysis of Privilege Usage of Two Roles

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

Example 5-2 shows how to analyze the privilege usage of two roles.

Example 5-2 Privilege Analysis of Privilege Usage of Two Roles



5.2.4.3 Example: Privilege Analysis of Privileges During SQL*Plus Use

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

Example 5-3 shows how to analyze privileges used to run SQL*Plus.

Example 5-3 Privilege Analysis of Privileges During SQL*Plus Use

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

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

Example 5-4 shows how to analyze the privileges used by session user PSMITH when running SQL*Plus.

Example 5-4 Privilege Analysis of PSMITH Privileges During SQL*Plus Access

5.2.5 Enabling a Privilege Analysis Policy

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

The <code>DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE</code> procedure enables a privilege policy and creates a capture run name for it. The run name defines the period of time that the capture took place.

- 1. Log in to the database instance as a user who has the CAPTURE ADMIN role.
- 2. Query the NAME and ENABLED columns of the DBA_PRIV_CAPTURES data dictionary view to find the existing privilege analysis policies and whether they are currently enabled.
- 3. Run the DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE procedure to enable the policy and optionally create a name for a capture run.

For example, to enable the privilege analysis policy logon_users_analysis:

```
BEGIN
   DBMS_PRIVILEGE_CAPTURE.ENABLE_CAPTURE (
   name => 'logon_users_analysis_pol',
   run name => 'logon users 04092016');
```



```
END;
```

If you do not need to specify the run_name parameter, then you can enable the policy by only specifying its name, as follows:

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

5.2.6 Disabling a Privilege Analysis Policy

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

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

DBMS PRIVILEGE CAPTURE.DISABLE CAPTURE procedure to disable a privilege analysis policy.

- 1. Log in to the database instance as a user who has the CAPTURE ADMIN role.
- 2. Query the NAME and ENABLED columns of the DBA_PRIV_CAPTURES data dictionary view to find the existing privilege analysis policies and whether they are currently disabled.
- 3. Run the DBMS PRIVILEGE CAPTURE.DISBLE CAPTURE procedure to enable the policy.

```
For example, to disable the privilege analysis policy <code>logon_users_analysis</code>:
```

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

5.2.7 Generating a Privilege Analysis Report

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

- About Generating a Privilege Analysis Report
 After the privilege analysis policy has been disabled, you can generate a report based on
 the capture run that you created for the privilege analysis policy.
- General Process for Managing Multiple Named Capture Runs
 When you enable a privilege analysis policy, you can create a named capture run for the policy's findings.
- Generating a Privilege Analysis Report Using DBMS_PRIVILEGE_CAPTURE
 The DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure generates a report showing the results of a privilege capture.
- Generating a Privilege Analysis Report Using Cloud Control You can generate a privilege analysis report using Cloud Control.
- Accessing Privilege Analysis Reports Using Cloud Control
 A privilege analysis report provides information about both used and unused privileges.

5.2.7.1 About Generating a Privilege Analysis Report

After the privilege analysis policy has been disabled, you can generate a report based on the capture run that you created for the privilege analysis policy.

To view the report results in SQL*Plus, query the privilege analysis-specific data dictionary views. In Enterprise Manager Cloud Control, you can view the reports from the Privilege Analysis page **Actions** menu. If a privilege is used during the privilege analysis process and

then revoked before you generate the report, then the privilege is still reported as a used privilege, but without the privilege grant path.

Related Topics

Privilege Analysis Policy and Report Data Dictionary Views
 Oracle Database provides data dictionary views that show information about analyzed privileges.

5.2.7.2 General Process for Managing Multiple Named Capture Runs

When you enable a privilege analysis policy, you can create a named capture run for the policy's findings.

The capture run defines a period of time from when the capture is enabled (begun) and when it is disabled (stopped). This way, you can create multiple runs and then compare them when you generate the privilege capture results.

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

- 1. Create the policy.
- 2. Enable the policy for the first run.
- 3. After a period time to collect user behavior data, disable this policy and its run.
- 4. Generate the results and then query the privilege analysis data dictionary views for information about this capture run.
 - If you omit the run_name parameter from the DBMS_PRIVILEGE_CAPTURE.GENERATE_RESULT procedure, then this procedure looks at all records as a whole and then analyzes them.
- 5. Re-enable the policy for the second run. You cannot create a new capture run if the policy has not been disabled first.
- 6. After you have collected the user data, disable the policy and the second run.
- 7. Generate the results.
- 8. Query the privilege analysis data dictionary views. The results from both capture runs are available in the views. If you only want to show the results of one of the capture runs, then you can regenerate the results and requery the privilege analysis views. You can also filter the results on the run name.

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

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

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

Related Topics

Tutorial: Using Capture Runs to Analyze ANY Privilege Use
 This tutorial demonstrates how to create capture runs to analyze the use of the READ ANY TABLE system privilege.



5.2.7.3 Generating a Privilege Analysis Report Using DBMS PRIVILEGE CAPTURE

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

- 1. Log in to the database instance as a user who has the CAPTURE ADMIN role.
- 2. Query the NAME and ENABLED columns of the DBA_PRIV_CAPTURES data dictionary view to find the existing privilege analysis policies and whether they are currently disabled.

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

3. Run the DBMS PRIVILEGE CAPTURE.GENERATE RESULT procedure using the following syntax:

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

In this specification:

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

For example, to generate a report for the privilege analysis policy <code>logon_users_analysis</code>:

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

Query the used privileges from DBA USED * data dictionary views with privilege grant paths.

5.2.7.4 Generating a Privilege Analysis Report Using Cloud Control

You can generate a privilege analysis report using Cloud Control.

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

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

Click OK.

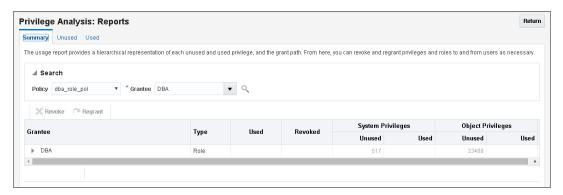
In the Privilege Analysis page, a Confirmation message notifies you that a report has been submitted. You can refresh the page until the job is complete. To view the report, select the policy name and then click **View Reports**.

5.2.7.5 Accessing Privilege Analysis Reports Using Cloud Control

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

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

The Privilege Analysis Reports page appears.



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

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

5.2.8 Dropping a Privilege Analysis Policy

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

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

- 1. Log in to the database instance as a user who has the CAPTURE ADMIN role.
- 2. Query the NAME and ENABLE columns of the DBA_PRIV_CAPTURES data dictionary view to find the policy and to check if it is enabled or disabled.
- 3. If the policy is enabled, then disable it.

For example:

```
EXEC DBMS PRIVILEGE CAPTURE.DISABLE CAPTURE ('logon users analysis pol');
```

Run the DBMS PRIVILEGE CAPTURE.DROP CAPTURE procedure to drop the policy.

For example:

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

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

Related Topics

Disabling a Privilege Analysis Policy
You must disable the privilege analysis policy before you can generate a privilege analysis
report.

5.3 Creating Roles and Managing Privileges Using Cloud Control

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

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

5.3.1 Creating a Role from a Privilege Analysis Report in Cloud Control

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

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

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

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



5.3.2 Revoking and Regranting Roles and Privileges Using Cloud Control

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

- 1. If Oracle Database Vault is enabled, then ensure that you are authorized as an owner of the Oracle System Privilege and Role Management realm.
 - In SQL*Plus, a user who has been granted the DV_OWNER role can check the authorization by querying the $DBA_DV_REALM_AUTH$ data dictionary view. To grant the user authorization, use the DBMS MACADM.ADD AUTH TO REALM procedure.
- 2. Generate the privilege analysis report.
- 3. In the Privilege Analysis page, select the policy on which you generated a report.
- 4. Select View Reports.
- 5. In the Privilege Analysis: Reports page, select the **Summary** tab.
- 6. Under Search, ensure that the **Policy** and **Grantee** menu options are set.
- 7. Under the Grantee area, expand the grantee options.
 - For example, for a role privilege analysis report for a role called ${\tt HR_ADMIN}$ role, you would expand the ${\tt HR_ADMIN}$ role to show the privileges that are associated with it.
- 8. Select each privilege to revoke and then click **Revoke**, or select **Regrant** to regrant the privilege to the role.

Related Topics

 Generating a Privilege Analysis Report Using Cloud Control You can generate a privilege analysis report using Cloud Control.

5.3.3 Generating a Revoke or Regrant Script Using Cloud Control

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

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

5.3.3.1 About Generating Revoke and Regrant Scripts

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

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



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

5.3.3.2 Generating a Revoke Script

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

- 1. If Oracle Database Vault is enabled, then ensure that you are authorized as an owner of the Oracle System Privilege and Role Management realm.
 - In SQL*Plus, a user who has been granted the DV_OWNER role can check the authorization by querying the DBA_DV_REALM_AUTH data dictionary view. To grant the user authorization, use the DBMS MACADM.ADD AUTH TO REALM procedure.
- 2. In Enterprise Manager, access the target Database home page as a user who has been granted the CAPTURE ADMIN role and the SELECT ANY DICTIONARY privilege.
 - See Oracle Database 2 Day DBA for more information.
- 3. From the Security menu, select Privilege Analysis.
- 4. Ensure that the privilege analysis reports that you want have been generated.
- 5. In the Privilege Analysis page, from the Actions menu, select Revoke Scripts.
- 6. On the Revoke Scripts page, click **Generate**.
 - The generate revoke script details wizard is displayed.
- 7. In the Script Details page, do the following: select a policy name from the **Policy Name** menu against which the revoke script needs to be prepared.
- 8. In the **Script Name** field, enter a unique name and for **Description**, a description for the script.

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

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

- 9. For Grantee (user/role), select All or Customize.
- Select All, None, or Customize for the Unused System Privileges, Unused Object Privileges, and Unused Roles settings.
- 11. Click Next.

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

12. Click Save.

The Revoke Scripts page appears.

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

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

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



Related Topics

 Generating a Privilege Analysis Report Using Cloud Control You can generate a privilege analysis report using Cloud Control.

5.3.3.3 Generating a Regrant Script

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

- 1. If Oracle Database Vault is enabled, then ensure that you are authorized as an owner of the Oracle System Privilege and Role Management realm.
 - In SQL*Plus, a user who has been granted the DV_OWNER role can check the authorization by querying the DBA_DV_REALM_AUTH data dictionary view. To grant the user authorization, use the DBMS MACADM.ADD AUTH TO REALM procedure.
- 2. In Enterprise Manager, access the target Database home page as a user who has been granted the CAPTURE ADMIN role and the SELECT ANY DICTIONARY privilege.
 - See Oracle Database 2 Day DBA for more information.
- From the Security menu, select Privilege Analysis.
- 4. Ensure that the reports you want have been generated.
 - See Generating a Privilege Analysis Report Using Cloud Control for more information.
- 5. In the Privilege Analysis page, select the policy on which the revoke script was based.
- 6. From the Actions menu, select Revoke Scripts.
- In the Revoke Scripts page, select the policy name that you had created earlier, and then click **Download Regrant Script** to download this script.

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

Related Topics

 Generating a Privilege Analysis Report Using Cloud Control You can generate a privilege analysis report using Cloud Control.

5.4 Tutorial: Using Capture Runs to Analyze ANY Privilege Use

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

- Step 1: Create User Accounts
 - You must create two users, one user to create the policy and a second user whose privilege use will be analyzed.
- Step 2: Create and Enable a Privilege Analysis Policy
 The user pa admin must create and enable the privilege analysis policy.
- Step 3: Use the READ ANY TABLE System Privilege
 User app user uses the READ ANY TABLE system privilege.
- Step 4: Disable the Privilege Analysis Policy
 You must disable the policy before you can generate a report that captures the actions of
 user app user.



Step 5: Generate and View a Privilege Analysis Report

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

Step 6: Create a Second Capture Run

Next, you are ready to create a second capture run for the <code>ANY_priv_analysis_pol</code> privilege analysis policy.

Step 7: Remove the Components for This Tutorial

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

5.4.1 Step 1: Create User Accounts

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

1. Log into the database instance as a user who has the CREATE USER system privilege.

For example:

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

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

For example:

```
sqlplus sec_admin@hrpdb
Enter password: password
```

To find the available PDBs, query the DBA_PDBs data dictionary view. To check the current PDB, run the show con name command.

2. Create the following users:

```
CREATE USER pa_admin IDENTIFIED BY password; CREATE USER app user IDENTIFIED BY password;
```

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

For example:

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

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

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

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

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

5.4.2 Step 2: Create and Enable a Privilege Analysis Policy

The user pa admin must create and enable the privilege analysis policy.

Connect as user pa admin.

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

2. Create the following privilege analysis policy:

In this example:

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

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

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

5.4.3 Step 3: Use the READ ANY TABLE System Privilege

User app user uses the READ ANY TABLE system privilege.

Connect as user app user.

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

2. Query the HR. EMPLOYEES table.

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

FIRST_NAME	LAST_NAME	SALARY
Steven	King	24000
Neena	Kochhar	17000
Lex	De Haan	17000
John	Russell	14000
Karen	Partners	13500
Michael	Hartstein	13000



Shelley	Higgins	12008
Nancy	Greenberg	12008

5.4.4 Step 4: Disable the Privilege Analysis Policy

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

1. Connect as user pa admin.

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

2. Disable the ANY priv analysis pol privilege policy.

```
EXEC DBMS PRIVILEGE CAPTURE.DISABLE CAPTURE ('ANY priv analysis pol');
```

5.4.5 Step 5: Generate and View a Privilege Analysis Report

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

1. As user pa admin, generate the privilege analysis results.

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

The generated results are stored in the privilege analysis data dictionary views.

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

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

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

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

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

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

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

5.4.6 Step 6: Create a Second Capture Run

Next, you are ready to create a second capture run for the <code>ANY_priv_analysis_pol</code> privilege analysis policy.

1. As user pa_admin, enable the ANY_priv_analysis_pol privilege analysis policy to use capture run ANY priv pol run 1.

2. Connect as user app user.

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

3. Query the HR. JOBS table.

```
SELECT MAX SALARY FROM HR.JOBS WHERE MAX SALARY > 20000;
```

Connect as user pa admin.

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

5. Disable the ANY priv analysis pol privilege policy.

```
EXEC DBMS PRIVILEGE CAPTURE.DISABLE CAPTURE ('ANY priv analysis pol');
```

Generate a second privilege analysis report.

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

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

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

SYS_PRIV	OBJECT_OWNER	OBJECT_NAME	RUN_NAME
READ ANY TABLE	HR SYS	EMPLOYEES DUAL	ANY_PRIV_POL_RUN_1 ANY_PRIV_POL_RUN_1
CREATE SESSION	515	DOAL	ANY_PRIV_POL_RUN_1
	SYS	DUAL	ANY_PRIV_POL_RUN_1
	SYSTEM	PRODUCT_PRIVS	ANY_PRIV_POL_RUN_1
	SYS	DBMS_APPLICATION_INFO	ANY_PRIV_POL_RUN_1
	SYS	DUAL	ANY_PRIV_POL_RUN_2
	SYS	DBMS_APPLICATION_INFO	ANY_PRIV_POL_RUN_2
	SYSTEM	PRODUCT_PRIVS	ANY PRIV POL RUN 2
	SYS	DUAL	ANY PRIV POL RUN 2
READ ANY TABLE	HR	JOBS	ANY_PRIV_POL_RUN_2

5.4.7 Step 7: Remove the Components for This Tutorial

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

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

```
EXEC DBMS PRIVILEGE CAPTURE.DROP CAPTURE ('ANY priv analysis pol');
```

Any capture runs that are associated with this policy are dropped automatically when you run the DBMS PRIVILEGE CAPTURE.DROP CAPTURE procedure.

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

2. Connect as the user who created the user accounts.

For example:

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

3. Drop the users pa admin and app user.

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

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

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

Step 1: Create User Accounts

user tjones.

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

- Step 2: Create and Enable a Privilege Analysis Policy
 User pa admin must create the and enable the privilege analysis policy.
- Step 3: Perform the Database Tuning Operations
 User tjones uses the DBA role to perform database tuning operations.
- Step 4: Disable the Privilege Analysis Policy
 You must disable the policy before you can generate a report that captures the actions of
- Step 5: Generate and View Privilege Analysis Reports
 With the privilege analysis policy disabled, user pa_admin can generate and view privilege analysis reports.
- Step 6: Remove the Components for This Tutorial
 You can remove the components that you created for this tutorial if you no longer need
 them.

5.5.1 Step 1: Create User Accounts

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

Log into the database instance as a user who has the CREATE USER system privilege.

For example:

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

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

For example:

```
sqlplus sec_admin@hrpdb
Enter password: password
```

To find the available PDBs, query the DBA_PDBs data dictionary view. To check the current PDB, run the show con name command.

2. Create the following users:

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

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

For example:

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

In SQL*Plus, a user who has been granted the $\mbox{DV}_\mbox{OWNER}$ role can check the authorization by querying the $\mbox{DBA}_\mbox{DV}_\mbox{REALM}_\mbox{AUTH}$ data dictionary view. To grant the user authorization, use the \mbox{DBMS} MACADM.ADD AUTH TO REALM procedure.

Grant the following roles and privileges to the users.

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

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

5.5.2 Step 2: Create and Enable a Privilege Analysis Policy

User pa admin must create the and enable the privilege analysis policy.

1. Connect as user pa admin.

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

2. Create the following privilege analysis policy:

```
BEGIN
DBMS_PRIVILEGE_CAPTURE.CREATE_CAPTURE(
  name => 'dba tuning priv analysis pol',
```

In this example:

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

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

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

5.5.3 Step 3: Perform the Database Tuning Operations

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

1. Connect as user tjones.

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

2. Run the following script to create the PLAN TABLE table.

```
@$ORACLE HOME/rdbms/admin/utlxplan.sql
```

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

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

```
EXPLAIN PLAN

SET STATEMENT_ID = 'Raise in Tokyo'

INTO PLAN_TABLE

FOR UPDATE HR.EMPLOYEES

SET SALARY = SALARY * 1.10

WHERE DEPARTMENT_ID =

(SELECT DEPARTMENT_ID FROM HR.DEPARTMENTS WHERE LOCATION_ID = 110);
```

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

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

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

Or

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

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

ANALYZE TABLE HR.EMPLOYEES LIST CHAINED ROWS INTO CHAINED_ROWS;



5.5.4 Step 4: Disable the Privilege Analysis Policy

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

1. Connect as user pa admin.

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

2. Disable the dba tuning priv analysis pol privilege policy.

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

5.5.5 Step 5: Generate and View Privilege Analysis Reports

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

1. As user pa admin, generate the privilege analysis results.

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

The generated results are stored in the privilege analysis data dictionary views.

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

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

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

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

Output similar to the following appears:

```
USENAME SYS_PRIV

------
PATH

TJONES ANALYZE ANY IMP_FULL_DATABASE

GRANT_PATH('TJONES', 'DBA')

TJONES ANALYZE ANY IMP_FULL_DATABASE

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

TJONES ANALYZE ANY IMP_FULL_DATABASE

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



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

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

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

Output similar to the following appears:

USERNAME	OBJ_PRIV	USED_ROLE	OBJECT_OWN	OBJECT_NAME	OBJECT_TYPE
TJONES	EXECUTE	PUBLIC	SYS	DBMS APPLICATION INFO	PACKAGE
TJONES	SELECT	PUBLIC	SYS	DUAL	TABLE
TJONES	SELECT	PUBLIC	SYS	DUAL	TABLE
TJONES	SELECT	PUBLIC	SYSTEM	PRODUCT_PRIVS	VIEW

Find the unused system privileges for user tjones.

```
col username format a9
col sys_priv format a35

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

USERNAME SYS_PRIV

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

5.5.6 Step 6: Remove the Components for This Tutorial

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

1. As user pa admin, drop the dba tuning priv analysis pol privilege analysis policy.

```
EXEC DBMS PRIVILEGE CAPTURE.DROP CAPTURE ('dba tuning priv analysis pol');
```

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

Connect as the user who created the user accounts.

For example:

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

3. Drop the users pa admin and tjones.

DROP USER pa_admin CASCADE;
DROP USER tjones;

5.6 Privilege Analysis Policy and Report Data Dictionary Views

Oracle Database provides data dictionary views that show information about analyzed privileges.

Table 5-1 lists these data dictionary views.

Table 5-1 Data Dictionary Views That Display Privilege Analysis Information

View	Description
DBA_PRIV_CAPTURES	Lists information about existing privilege analysis policies
DBA_USED_PRIVS	Lists the privileges and capture runs that have been used for reported privilege analysis policies
DBA_UNUSED_GRANTS	Lists the privilege grants that have not been used
DBA_UNUSED_PRIVS	Lists the privileges and capture runs that have not been used for reported privilege analysis policies
DBA_USED_OBJPRIVS	Lists the object privileges and capture runs that have been used for reported privilege analysis policies. It does not include the object grant paths.
DBA_UNUSED_OBJPRIVS	Lists the object privileges and capture runs that have not been used for reported privilege analysis policies. It does not include the object privilege grant paths.
DBA_USED_OBJPRIVS_PATH	Lists the object privileges and capture runs that have been used for reported privilege analysis policies. It includes the object privilege grant paths.
DBA_UNUSED_OBJPRIVS_PATH	Lists the object privileges and capture runs that have not been used for reported privilege analysis policies. It includes the object privilege grant paths.
DBA_USED_SYSPRIVS	Lists the system privileges and capture runs that have been used for reported privilege analysis policies. It does not include the system privilege grant paths.
DBA_UNUSED_SYSPRIVS	Lists the system privileges and capture runs that have not been used for reported privilege analysis policies. It does not include the system privilege grant paths.
DBA_USED_SYSPRIVS_PATH	Lists the system privileges and capture runs that have been used for reported privilege analysis policies. It includes the system privilege grant paths.
DBA_UNUSED_SYSPRIVS_PATH	Lists the system privileges and capture runs that have not been used for reported privilege analysis policies. It includes system privilege grant paths
DBA_USED_PUBPRIVS	Lists all the privileges and capture runs for the PUBLIC role that have been used for reported privilege analysis policies
DBA_USED_USERPRIVS	Lists the user privileges and capture runs that have been used for reported privilege analysis policies. It does not include the user privilege grant paths.



Table 5-1 (Cont.) Data Dictionary Views That Display Privilege Analysis Information

View	Description
DBA_UNUSED_USERPRIVS	Lists the user privileges and capture runs that have not been used for reported privilege analysis policies. It does not include the user privilege grant paths.
DBA_USED_USERPRIVS_PATH	Lists the user privileges and capture runs that have been used for reported privilege analysis policies. It includes the user privilege grant paths.
DBA_UNUSED_USERPRIVS_PATH	Lists the privileges and capture runs that have not been used for reported privilege analysis policies. It includes the user privilege grant paths.

Related Topics

Oracle Database Reference



Configuring Centrally Managed Users with Microsoft Active Directory

Oracle Database can authenticate and authorize Microsoft Active Directory users with the database directly without intermediate directories or Oracle Enterprise User Security.

- Introduction to Centrally Managed Users with Microsoft Active Directory
 Centrally managed users (CMU) provides a simpler integration with Microsoft Active
 Directory to allow centralized authentication and authorization of users.
- Configuring the Oracle Database-Microsoft Active Directory Integration
 Before you can use Microsoft Active Directory to authenticate and authorize users, you
 must configure the connection from the Oracle database to Active Directory.
- Configuring Authentication for Centrally Managed Users
 You can configure password authentication, Kerberos authentication, or public key
 infrastructure (PKI) authentication.
- Configuring Authorization for Centrally Managed Users
 With centrally managed users, you can manage the authorization for Active Directory users to access Oracle databases.
- Troubleshooting Centrally Managed Users
 Oracle provides error messages that help you troubleshoot common errors that may arise when a Microsoft Active Directory user tries to log in to an Oracle database.
- Integration of Oracle Database with Microsoft Active Directory Account Policies
 As part of the Oracle Database-Microsoft Active Directory integration, Oracle Database
 enforces the Active Directory account policies when Active Directory users log into the
 Oracle database.
- Configuring Centrally Managed Users with Oracle Autonomous Database
 You can deploy centrally managed users (CMU) on Oracle Autonomous Database.
- Troubleshooting Centrally Managed Users
 Oracle provides error messages that help you troubleshoot common errors that may arise when a Microsoft Active Directory user tries to log in to an Oracle database.

6.1 Introduction to Centrally Managed Users with Microsoft Active Directory

Centrally managed users (CMU) provides a simpler integration with Microsoft Active Directory to allow centralized authentication and authorization of users.

- About the Oracle Database-Microsoft Active Directory Integration
 Centrally managed users provides a simpler integration with Microsoft Active Directory to allow centralized authentication and authorization of users.
- How Centrally Managed Users with Microsoft Active Directory Works
 The integration works by mapping Microsoft Active Directory users and groups directly to Oracle database users and roles.

- Centrally Managed User-Microsoft Active Directory Architecture
 The CMU with Active Directory architecture enables Oracle Database users and roles to be managed in Active Directory.
- Supported Authentication Methods
 The Oracle Database-Microsoft Active Directory integration supports three common authentication methods.
- Users Supported by Centrally Managed Users with Microsoft Active Directory
 CMU with Active Directory supports exclusively mapped users, users mapped to shared schemas, and administrative users.
- How the Oracle Multitenant Option Affects Centrally Managed Users
 Multitenant database users in pluggable databases (PDBs) can connect to a central
 Microsoft Active Directory or, if required, users in an individual PDB can connect to a
 different Microsoft Active Directory.
- Centrally Managed Users with Database Links
 CMU supports both fixed user database links and connected user database links, but not current user database links.

6.1.1 About the Oracle Database-Microsoft Active Directory Integration

Centrally managed users provides a simpler integration with Microsoft Active Directory to allow centralized authentication and authorization of users.

The minimum version requirement for Active Directory server operating system is Microsoft Windows Server 2012. This minimum supported version will be updated when Microsoft drops support for older releases.

This integration enables organizations to use Active Directory to centrally manage users and roles in multiple Oracle databases with a single directory along with other Information Technology services. Active Directory users can authenticate to the Oracle database by using credentials that are stored in Active Directory. Active Directory users can also be associated with database users (schemas) and roles by using Active Directory groups. Microsoft Active Directory users can be mapped to exclusive or shared Oracle Database users (schemas), and be associated with database roles through their group membership in the directory. Active Directory account policies such as password expiration time and lockout after a specified number of failed login attempts are honored by the Oracle Database when users login.

Before Oracle Database 18c release 1 (18.1), database user authentication and authorization could be integrated with Active Directory by configuring Oracle Enterprise User Security and installing and configuring Oracle Internet Directory (or Oracle Universal Directory). This architecture is still available and will continue to be used by users who must use the Oracle enterprise domain and current user database link between trusted databases, complex enterprise roles, and having a single place for auditing database access privileges and roles.

The majority of organizations do not have these complex requirements. Instead, they can use centrally managed users (CMUs) with Active Directory. This integration is designed for organizations who prefer to use Active Directory as their centralized identity management solution. Oracle Net Naming Services continues to work as it did before with directory services.

Organizations can use Kerberos, PKI, or password authentication with CMU with Active Directory. Use of CMU with Active Directory is backward compatible with currently supported Oracle Database clients. This means that LDAP bind operations are not used for password authentication and you will need to add an Oracle filter to Active Directory along with an extension to the Active Directory schema to store password verifiers. Organizations using Kerberos or PKI will not need to add the filter or extend Active Directory schema.



The Oracle Database-Active Directory integration is particularly beneficial for the following types of users:

- Users who are currently using strong authentication such as Kerberos or Public Key Infrastructure (PKI). These users already use a centralized identity management system
- Users who currently use Oracle Enterprise User Security, Oracle Internet Directory, Oracle Unified Directory, Oracle Virtual Directory, and need to integrate with Active Directory.

6.1.2 How Centrally Managed Users with Microsoft Active Directory Works

The integration works by mapping Microsoft Active Directory users and groups directly to Oracle database users and roles.

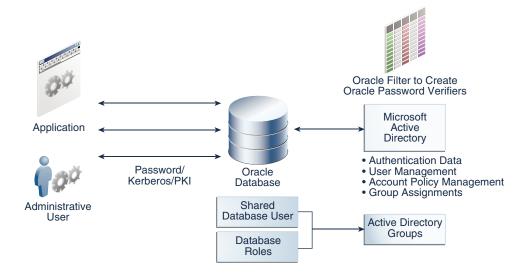
In order for the Oracle Database CMU with Active Directory integration to work, the Oracle database must be able to login to a service account specifically created for the database in Active Directory. The database uses this service account to query Active Directory for user and group information when a user logs into the database. This Active Directory service account must have all the privileges required to query the user and group information as well as being able to write updates related to the password policies in Active Directory (for example, failed login attempts, clear failed login attempts). Users can authenticate using passwords, Kerberos, or PKI and either be assigned to an exclusive schema or a shared schema. Mapping of an Active Directory user to a shared schema is determined by the association of the user to an Active Directory group that is mapped to the shared schema. Active Directory groups can also be mapped to database global roles. An Active Directory security administrator can assign a user to groups that are mapped to shared database global users (schemas) and/or database global roles, and hence update privileges and roles that are assigned to the Active Directory user in a database.

6.1.3 Centrally Managed User-Microsoft Active Directory Architecture

The CMU with Active Directory architecture enables Oracle Database users and roles to be managed in Active Directory.

The following figure illustrates the Oracle Database CMU feature. In this figure, users, either through applications as non-administrative users or administrative users, connect to the Oracle database with either password, Kerberos, or public key infrastructure (PKI) authentication. The database connection to Active Directory enables these users and roles to be mapped with Active Directory users and groups. If you plan to use password authentication, then you must install an Oracle filter in Active Directory. You can use an Oracle provided utility to install the Oracle filter that will generate Oracle password verifiers for individual users as needed. The utility can also be used to extend the Active Directory schema to hold the Oracle password verifiers. With Oracle Database centrally managed users, an Active Directory administrator can control the authentication, user management, account policies, and group assignments of Active Directory users and groups who have been mapped to Oracle Database users and roles.





6.1.4 Supported Authentication Methods

The Oracle Database-Microsoft Active Directory integration supports three common authentication methods.

These authentication methods are as follows:

- Password authentication
- Kerberos authentication
- Public key infrastructure (PKI) authentication (certificate-based authentication)

Related Topics

Configuring Authentication for Centrally Managed Users
 You can configure password authentication, Kerberos authentication, or public key infrastructure (PKI) authentication.

6.1.5 Users Supported by Centrally Managed Users with Microsoft Active Directory

CMU with Active Directory supports exclusively mapped users, users mapped to shared schemas, and administrative users.

These users are as follows:

Directory users that access an Oracle database using a shared schema.

This type of directory user can connect to a shared schema in the database by being part of a directory group that is mapped to the shared schema (database user). Using shared schemas allows centralized Active Directory management of database users and is the recommended best practices over using exclusive schemas (described next). Even if there is only one user associated with a schema (for example, an administrator responsible for database backup), it is easier to manage adding another backup administrator or removing the existing administrator by making changes only in Active Directory instead of making changes in all associated databases as well.

Users will be given additional privileges appropriate to their task using global roles that are mapped to groups in Active Directory. With this design, a user can change his or her tasks

within an organization and have new database privileges through a new group in Active Directory.

Active Directory users could accidentally (or on purpose) be a member of multiple groups in Active Directory that are mapped to different shared schemas on the same database. The user could also have an exclusive mapping to a database schema. In cases where the user has multiple possible schema mappings when they login, the following precedence rules apply:

- If an exclusive mapping exists for a user, then that mapping takes precedence over any other shared mappings.
- If multiple shared schema mappings exist for a user, then the shared user mapping with lowest schema ID (USER ID) takes precedence.

Oracle recommends only having one possible mapping per user so unexpected schema mappings do not occur.

- Exclusively mapped global users who are regular Oracle Database users in two- and three-tier applications, or users who have direct privilege grants in the database.
 - Oracle recommends that you grant privileges to these users through global roles. This type of privilege grant facilitates authorization management by centrally managing privileges and roles for a user instead of having to log in into each database to update privileges and roles for the user.
- Administrative global users, who have the following administrative privileges: SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, and SYSRAC.

You **cannot** grant these administrative privileges through global roles. To authorize an Active Directory user with these administrative privileges, you must map the directory user to a database user (exclusively or with a shared schema) that has the system administrative privilege already granted to the database user account.

Related Topics

Configuring Authorization for Centrally Managed Users
 With centrally managed users, you can manage the authorization for Active Directory users
 to access Oracle databases.

6.1.6 How the Oracle Multitenant Option Affects Centrally Managed Users

Multitenant database users in pluggable databases (PDBs) can connect to a central Microsoft Active Directory or, if required, users in an individual PDB can connect to a different Microsoft Active Directory.

All PDBs and the root container in a multitenant database can have a shared configuration, so that the entire CDB can authenticate and authorize users against a single Active Directory server, mulitple Active Directory servers in one Windows domain, or multiple Active Directory servers in trusted Windows domains, based on the shared configuration. Alternatively, individual PDBs can authenticate and authorize users against different Active Directory servers in the same Windows domain or different (trusted or un-trusted) Windows domains, based on their individual configurations.

6.1.7 Centrally Managed Users with Database Links

CMU supports both fixed user database links and connected user database links, but not current user database links.



There is no special requirement for CMU-Active Directory users to use the fixed user database links. CMU-Active Directory users using password, Kerberos, or PKI authentication can use fixed user database links as regular database users do. Kerberos authentication works the same with Oracle Database strong authentication with database links. For more information, see My Oracle Support note 1370327.1.

For CMU-Active Directory users to use connected user database links, only password authentication is supported, and both source and target databases must be configured with CMU-Active Directory to allow the same Active Directory user to log in both databases using password authentication.

6.2 Configuring the Oracle Database-Microsoft Active Directory Integration

Before you can use Microsoft Active Directory to authenticate and authorize users, you must configure the connection from the Oracle database to Active Directory.

- About Configuring the Oracle Database-Microsoft Active Directory Connection
 Before you configure this connection, you must have Microsoft Active Directory installed
 and configured.
- Connecting to Microsoft Active Directory
 You can configure a Microsoft Active Directory connection during the Oracle database
 creation or with an existing Oracle database.

6.2.1 About Configuring the Oracle Database-Microsoft Active Directory Connection

Before you configure this connection, you must have Microsoft Active Directory installed and configured.

You must create an Oracle service directory user in Active Directory, configure the Oracle Database connection to Active Directory, and then depending on the authentication type, configure the database and Active Directory for password, Kerberos, or public key infrastructure (PKI) authentication. Before you map Database users and global roles to Active Directory users and groups, you must ensure that the Active Directory users and groups have been created. You will map the database users and global roles to Active Directory users and groups by using the CREATE USER, CREATE ROLE, ALTER USER, ALTER ROLE SQL statements with the GLOBALLY clause. An Active Directory system administrator must also set up new Active Directory groups with Active Directory users to meet your requirements.

The Active Directory system administrator is responsible for setting Active Directory connections with or without SASL bind. The Oracle Database will automatically try the Active Directory connection first with SASL bind and if it fails, it will try it without SASL bind but still secured with TLS. This means that regardless of how the Microsoft Active Directory administrator may have the SASL settings configured on Active Directory, the Oracle database will connect even if the SASL bind is unsuccessful.

6.2.2 Connecting to Microsoft Active Directory

You can configure a Microsoft Active Directory connection during the Oracle database creation or with an existing Oracle database.



 Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions

The Oracle service directory user account is for the interaction between Oracle Database and the LDAP directory service.

 Step 2: For Password Authentication, Install the Password Filter and Extend the Microsoft Active Directory Schema

You can use the Oracle <code>opwdintg.exe</code> executable on the Active Directory server to install the password filter and extend the Active Directory schema.

- Step 3: If Necessary, Install the Oracle Database Software
 If you have not done so yet, then use Oracle Universal Installer (OUI) to install the Oracle software.
- Step 4: Create the dsi.ora or Idap.ora File
 The dsi.ora and ldap.ora files specify connections for centrally managed users for Active Directory.
- Step 5: Request an Active Directory Certificate for a Secure Connection
 After you have configured the dsi.ora or ldap.ora file, you are ready to prepare Microsoft
 Active Directory and Oracle Database certificates for a secure connection.
- Step 6: Create the Wallet for a Secure Connection
 After you have copied the Active Directory certificate, you are ready to add it to the Oracle wallet.
- Step 7: Configure the Microsoft Active Directory Connection
 Next, you are ready to connect the database to Active Directory using the settings you have so far.
- Step 8: Verify the Oracle Wallet
 The orapki utility can verify that the wallet for this database was created successfully.
- Step 9: Test the Integration
 To test the integration, you must set the ORACLE_HOME, ORACLE_BASE, and ORACLE_SID environment variables and then verify the LDAP parameter settings.

6.2.2.1 Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions

The Oracle service directory user account is for the interaction between Oracle Database and the LDAP directory service.

In addition to being used for the Oracle Database-to-LDAP directory service interaction, the Oracle service directory user account can be used for Kerberos.

This account is an Active Directory user account that Oracle Database uses to bind to Active Directory domain controllers and query for users and groups information from Active Directory, update login success or failure, and if Kerberos is configured, update Kerberos authentication. The minimum permissions required for this account are Read properties (of Active Directory users who will log in to a database) permission, and if database password authentication is to be used by Active Directory users, the Write lockoutTime (property of the Active Directory users) permission, and Control Access (of the orclCommonAttribute property of the Active Directory users) permission.

- Log in to a Windows domain controller of Microsoft Active Directory as an administrator who has administrative privileges to create a user account and grant permissions to the user account.
- Create the Oracle service directory user account as an Active Directory user.



Create the service user account in the directory. Depending on the Windows domains that your Active Directory users will use, you can choose where the service user account will be created. Follow these guidelines:

- If all the Active Directory users will be in one domain, then create this account in that domain. Doing so will help performance.
- If the Active Directory users will be in multiple Windows domains, then create this service user account in a domain that is trusted by all other domains.
 - The domain chosen must be trusted by all other domains.
 - The service user must be able to bind to all of these multiple Windows domains, and must be able to access the properties of Active Directory users in all of these multiple Windows domains with the granted permissions.
 - All other domains must support simple bind over TLS/SSL to allow the access of the service user from the trusted domain.
 - All other domains administrators must grant the required minimum permissions to the service user account from the trusted domain.
- 3. Grant the Oracle service directory user account in the Active Directory the following permissions on the properties of the Active Directory users who need to access Oracle databases:
 - Read properties (of Active Directory users who will log in to an Oracle database)
 - Write lockoutTime (property of Active Directory users who will use password authentication to log in to an Oracle database)
 - Control Access (of the orclCommonAttribute property of the Active Directory users who will use password authentication to log in to an Oracle database)

6.2.2.2 Step 2: For Password Authentication, Install the Password Filter and Extend the Microsoft Active Directory Schema

You can use the Oracle <code>opwdintg.exe</code> executable on the Active Directory server to install the password filter and extend the Active Directory schema.

You do not need to perform this step if your authentication method is Kerberos or SSL. The <code>opwdintg.exe</code> executable installs the Oracle password filter, extends the Active Directory schema, and creates Active Directory groups to allow Oracle Database password authentication with Active Directory. This procedure adds an <code>orclCommonAttribute</code> property to the Active Directory schema for user accounts.

Note:

You must install the Oracle password filter on **every** Windows domain controller in a domain, to ensure that Oracle password verifiers will be generated for Active Directory users in this domain if they need to use password authentication to log in Oracle database.

Note also that orclCommonAttribute stores Oracle password verifier for the Active Directory user. This attribute is also used for password authentication by other Oracle products or features such as Enterprise User Security. For security consideration, you should deny everyone except the Oracle service directory user from accessing the orclCommonAttribute property.



- Access the latest version of the opwdintg.exe (Oracle Password Integration) utility.
 - If you have a My Oracle Support account: Log in to your account at My Oracle Support and then search for Doc ID 2462012.1. Download opwdintg.exe from this location. This version is the latest version.
 - If you do not have a My Oracle Support account: Register for a My Oracle Support account so that you can download the latest version of opwdintg.exe from Doc ID 2462012.1.
- 2. Using a secure method of copying (such as sftp), copy opwdintg.exe to a temporary directory (for example, C:\temp) on each Windows domain controller.
- Connect to each Windows domain controller as the Active Directory administrator.Currently, the opwdintg.exe utility requires English for the Windows OS.
- 4. Ensure that the Windows OS language setting is English.
- 5. Run the opwdintg.exe utility on each Windows domain controller.

If you reinstall an updated password filter using a newer <code>opwdintg.exe</code>, then you must restart the domain controller.

Use one of the following methods to run the opwdintg.exe utility:

- Open the Windows Explorer and then double click the opwdintg.exe utility.
- Open a Windows command prompt and then follow these steps:
 - a. Navigate to the directory where the opwdintg.exe utility is located. For example:

```
cd c:\temp
```

b. Execute the utility from the command line by typing the following command:

```
.\opwdintg.exe
```

- **6.** Answer the following prompts:
 - Do you want to extend AD schema? [Yes/No]: Enter Yes.
 Extending the Active Directory schema requires the Windows OS language setting to be English.
 - Schema extension for this domain will be permanent. Continue? [Yes/No]:Enter Yes.

Note the following:

- You can only extend the Active Directory schema one time. If you try to extend the schema again, error messages appear, but you can ignore these errors.
- This step creates the following three verifier groups. If these groups already exist, then errors will appear, but you can ignore these errors. These verifier groups can be moved from the installed AD Users folder or outside this folder structure for user objects.
 - * ORA VFR MD5 is required when the Oracle Database WebDAV client is used.
 - * ORA VFR 11G enables the use of the Oracle Database 11G password verifier.
 - * ORA VFR 12C enables the use of the Oracle Database 12C password verifier.
- Unless you have backed up the Active Directory schema, once extended, the Active Directory schema extension cannot be reverted.

The next two prompts depend on whether the password filter has been installed already.



Found password filter installed already. Do you want to deinstall? [Yes/No]: This
prompt appears if the password filter has already been installed. In most cases, enter
No to not deinstall the filter.

If you enter Yes to deinstall the password filter, then you must re-run <code>opwdintg.exe</code> to re-install the password filter after you complete these prompts. Otherwise, after you restart the computer, the password verifiers will be no longer be generated when Active Directory users change their passwords.

- **Do you want to install Oracle password filter? [Yes/No]:** This prompt appears if the password filter has not been installed yet. Enter Yes.
- The change requires machine reboot. Do you want to reboot now? [Yes/No]: Enter Yes.

6.2.2.3 Step 3: If Necessary, Install the Oracle Database Software

If you have not done so yet, then use Oracle Universal Installer (OUI) to install the Oracle software.

You only need to install the Oracle Database software, not the full database. After you install the Oracle database software, you can configure centrally managed users with Active Directory during database creation by using Database Configuration Assistant (DBCA). You can also configure centrally managed users with Active Directory using DBCA or manually after database creation.

• Follow the instructions in the *Oracle Database Installation Guide* for your platform to install the Oracle software.

After you install the Oracle database software, then you can configure centrally managed users with Active Directory during database creation using DBCA. You can also configure centrally managed users with Active Directory using DBCA or manually after the database creation.

6.2.2.4 Step 4: Create the dsi.ora or Idap.ora File

The dsi.ora and ldap.ora files specify connections for centrally managed users for Active Directory.

- Comparison of the dsi.ora and ldap.ora Files

 How you use the dsi.ora and ldap.ora depends on how ldap.ora is used with other services.
- About Using a dsi.ora File
 You use a dsi.ora file to specify Active Directory servers for centrally managed users.
- Creating the dsi.ora File
 The dsi.ora configuration file sets the information to find the Active Directory servers for centrally managed users.
- About Using an Idap.ora File
 You can use an ldap.ora file to specify Active Directory servers for centrally managed
 users.
- Creating the Idap.ora File
 These steps assume that Idap.ora is not being used for net naming services and can be used to set up the connection with Active Directory for centrally managed users.

6.2.2.4.1 Comparison of the dsi.ora and Idap.ora Files

How you use the dsi.ora and ldap.ora depends on how ldap.ora is used with other services.

The dsi.ora file specifies connections for centrally managed users for Active Directory. The ldap.ora file can also specify the connection to the Active Directory server. However, because each individual PDB cannot have its own ldap.ora, and also ldap.ora may already be used (or may be used in the future) for other services like net naming services, Oracle recommends the use of dsi.ora for centrally managed users.

6.2.2.4.2 About Using a dsi.ora File

You use a dsi.ora file to specify Active Directory servers for centrally managed users.

You must manually create the <code>dsi.ora</code> file to identify the Active Directory servers. The <code>dsi.ora</code> file provides Active Directory connection information for all pluggable databases if it is located in the same places where the <code>ldap.ora</code> file can be placed. A <code>dsi.ora</code> file in a PDB-specific wallet location takes precedence over the main <code>dsi.ora</code> file for that PDB only.



If you are using <code>ldap.ora</code> for naming services, then do not make any changes to <code>ldap.ora</code> for the CMU with Active Directory configuration. Only use <code>dsi.ora</code> to configure CMU-Active Directory.

Placement of dsi.ora

Oracle recommends that you use directories for writable files under <code>\$ORACLE_BASE</code>, not under <code>\$ORACLE_HOME</code>. Starting with Oracle Database 18c, you can optionally set the <code>\$ORACLE_HOME</code> directory to be read-only. Hence, you should place the <code>dsi.ora</code> file in a directory that is outside of <code>\$ORACLE_HOME</code> to accommodate the <code>dsi.ora</code> configuration for future releases.

Search Order for dsi.ora

When you create the dsi.ora file, Oracle Database searches for it in the following order:

- 1. If the WALLET_LOCATION setting is included in the sqlnet.ora file, then for a non-multitenant database or the root container of a multitenant database, Oracle searches for it in the location that is specified in sqlnet.ora. For a PDB of a multitenant database, Oracle searches for it in the per-PDB wallet location that is in the WALLET LOCATION specified in sqlnet.ora/pdb guid directory.
- 2. If the WALLET_LOCATION setting is not included in the sqlnet.ora file, then Oracle Database searches for it in the default wallet location.
- 3. If Oracle Database cannot find dsi.ora in the wallet location, then Oracle Database searches for it in the following order. These are the same locations that Oracle Database searches for the ldap.ora file.
 - a. \$LDAP_ADMIN environment variable setting
 - b. \$ORACLE_HOME/ldap/admin directory
 - \$TNS ADMIN environment variable setting
 - d. \$ORACLE HOME/network/admin directory



When to Use dsi.ora

Oracle recommends that you use only <code>dsi.ora</code> to identify the Active Directory servers for centrally managed users. If both <code>dsi.ora</code> and <code>ldap.ora</code> are configured in the same database for centrally managed users for Active Directory and are both located in the same directory, then <code>dsi.ora</code> takes precedence over the <code>ldap.ora</code> file. If they are in different directories, then Oracle uses the first one that it finds in the location precedence list above to find the Active Directory server. If the directory server type in the first found <code>dsi.ora</code> or <code>ldap.ora</code> is not Active Directory, then centrally managed users will not be enabled.

Using dsi.ora in a Multitenant Environment

You can specify dsi.ora files for individual PDBs in a multitenant database. A PDB-specific dsi.ora will override the common settings in the shared dsi.ora or ldap.ora for that one PDB. Different PDBs can connect to different Active Directory servers for CMU. The dsi.ora file for an individual PDB is located in the same directory as the wallet for that PDB.

When the WALLET_LOCATION parameter in the sqlnet.ora file is set, then the dsi.ora file for an individual PDB will be in the per-PDB wallet in the

WALLET LOCATION specified in sqlnet.ora/pdb guid/directory.

When the WALLET_LOCATION parameter in the sqlnet.ora file is not set, then the default wallet location for an individual container in a multitenant database is the <code>\$ORACLE_BASE/admin/db_unique_name/pdb_guid/wallet/</code> directory. For each PDB to use the default wallet location, you must not set <code>WALLET_LOCATION</code> in <code>sqlnet.ora</code>.

To find the db unique name, connect to the CDB root and execute the following query:

```
SELECT DB_UNIQUE_NAME FROM V$DATABASE;
```

To find the pdb guid, from the CDB root, execute the following query:

SELECT PDB NAME, GUID FROM DBA PDBS;

How the WALLET_LOCATION Parameter in sqlnet.ora Affects dsi.ora

Setting or not setting the WALLET LOCATION parameter in sqlnet.ora has the following effects:

- If WALLET_LOCATION is not set in sqlnet.ora, then you can also place dsi.ora in the default wallet directory for the CDB root container, located in the \$ORACLE_BASE/admin/db_unique_name/wallet directory. However, this will only connect the CDB root container to the Active Directory, not the entire CDB database.
- If WALLET_LOCATION is set in sqlnet.ora, then you can place the dsi.ora in that wallet location, and this will also only connect the CDB root container to the Active Directory, not the entire CDB database.

Changing the Contents of dsi.ora

If you change the contents of dsi.ora after the database has been started, then you must either restart the database instance or re-execute the following DDL to make the updated content in dsi.ora effective:

```
ALTER SYSTEM SET LDAP DIRECTORY ACCESS = 'PASSWORD';
```

In a multitenant environment, you should set the LDAP_DIRECTORY_ACCESS parameter in each PDB, not in the CDB root.



6.2.2.4.3 Creating the dsi.ora File

The dsi.ora configuration file sets the information to find the Active Directory servers for centrally managed users.

To use the dsi.ora configuration file:

- 1. Log in to the host where the Oracle database is located.
- 2. Choose a directory where to use the dsi.ora file, based on the search order for the dsi.ora file. (See Related Topics.) If this directory does not exist, then create the directory. Then go to this directory to create the dsi.ora file.
- 3. Add the following parameters to the dsi.ora file:
 - DSI_DIRECTORY_SERVERS, which sets the Active Directory server host and port number, and alternate directory servers. The directory server name must be a fully qualified name. You can also have multiple Active Directory servers here if you want to use multiple Windows domains. For example:

```
DSI_DIRECTORY_SERVERS = (AD-server.production.examplecorp.com:389:636,
sparky.production.examplecorp.com:389:636)
```

Active Directory domain servers in a high availability and failover configuration can be configured with CMU. You can configure high availability and failover Active Directory domain servers by one of the following methods:

- Using a load balancer in front of the Active Directory domain servers
- Listing each Active Directory domain server by host name or IP address in a list
- Using a domain name that returns a different Active Directory domain server

Using a load balancer is the preferred choice, especially if you already use one for the Active Directory domain servers. The load balancer enables you to manage and add or subtract Active Directory domain servers behind the load balancer without having to make any changes to the <code>dsi.ora</code> file. Specifying a list of Active Directory domain servers is quicker and less expensive, but it ties you to the Active Directory domain servers so changes (new or dropped servers) must be reflected in <code>dsi.ora</code>. Using a domain name offers some high availability and failover, but it is not an ideal solution. The DNS will need to return different servers instead of the same server every time. CMU will try the first returned server from a domain name look-up and if that fails, then the authentication will fail. However, using domain names gives you some ability to use different Active Directory domain servers without having to specify the list of servers in <code>dsi.ora</code>.

DSI_DEFAULT_ADMIN_CONTEXT, which sets the search base where the Active Directory users and groups are located. This parameter is optional. By default, Oracle locates Active Directory users and groups in Active Directory's default naming context. Oracle recommends that you do not set this parameter. Set this parameter only if you want to limit the search scope for Active Directory users and groups. For example:

```
DSI_DEFAULT_ADMIN_CONTEXT =
"OU=sales,DC=production,DC=examplecorp,DC=com"
```



DSI_DIRECTORY_SERVER_TYPE, which determines the Active Directory server access. You must set it to AD for Active Directory. Enter this value in upper case.

```
DSI_DIRECTORY_SERVER_TYPE = AD
```

Related Topics

About Using a dsi.ora File
 You use a dsi.ora file to specify Active Directory servers for centrally managed users.

6.2.2.4.4 About Using an Idap.ora File

You can use an ldap.ora file to specify Active Directory servers for centrally managed users.

If you are already using an ldap.ora file for another purpose such as net naming services, then you must use the dsi.ora file to configure centrally managed users to connect with Active Directory for user authentication and authorization. Even if Active Directory is already being used for net naming services, then you must create and use a dsi.ora file to identify the Active Directory servers for centrally managed users. Even if the database currently is not using ldap.ora for another service, Oracle recommends using dsi.ora in case ldap.ora will be used at a future time for net naming services.

If ldap.ora is being used for naming services, then do not make any changes to ldap.ora. Only use dsi.ora to configure CMU-Active Directory.

Benefit of Using Idap.ora

The benefit of using <code>ldap.ora</code> is that you can use the DBCA graphical interface or the DBCA silent mode to complete configuring the connection to the Active Directory servers. When using <code>dsi.ora</code>, the steps to complete configuring the connection to Active Directory must be done separately.

Placement of Idap.ora

Typically, the <code>ldap.ora</code> file is stored in the <code>\$ORACLE_HOME/network/admin</code> directory. Usually, the <code>ldap.ora</code> file cannot be in the same directory as the <code>WALLET_LOCATION</code> that is specified in the <code>sqlnet.ora</code> file, unless the <code>WALLET_LOCATION</code> is set to <code>\$ORACLE_HOME/network/admin</code>.

Search Order for Idap.ora

After you create the ldap.ora file, Oracle Database searches for it in the following order:

- 1. \$LDAP ADMIN environment variable setting
- 2. \$ORACLE HOME/ldap/admin directory
- 3. \$TNS ADMIN environment variable setting
- \$ORACLE HOME/network/admin directory

Changing the Contents of Idap.ora

If you change the contents of ldap.ora after the database has been started, then you must either restart the database instance or re-execute the following DDL to make the updated content in ldap.ora effective:

```
ALTER SYSTEM SET LDAP DIRECTORY ACCESS = 'PASSWORD';
```



In a multitenant environment, set the LDAP_DIRECTORY_ACCESS parameter in each PDB, not in the CDB root.

6.2.2.4.5 Creating the Idap.ora File

These steps assume that ldap.ora is not being used for net naming services and can be used to set up the connection with Active Directory for centrally managed users.

- Log in to the host where the Oracle database is located.
- 2. Choose a directory where to use the ldap.ora file, based on the search order for the ldap.ora file. (See Related Topics.) If this directory does not exist, then create the directory. Then go to this directory to create the ldap.ora file.
- If the ldap.ora file does not exist, then create it by using a text editor.
 If the ldap.ora file does exist, create a backup of this file, and then open ldap.ora.
- 4. Add the following parameters to the ldap.ora file:
 - DIRECTORY_SERVERS, which sets the Active Directory server host and port number, and alternate directory servers. You can also have multiple Active Directory servers here if you want to use multiple Windows domains. The directory server name must be a fully qualified name. For example:

```
DIRECTORY_SERVERS = (AD-server.production.examplecorp.com:389:636,
sparky.production.examplecorp.com:389:636)
```

DEFAULT_ADMIN_CONTEXT, which sets the search base where the Active Directory users
and groups are located. This parameter is optional. By default, Oracle locates Active
Directory users and groups in the Active Directory's default naming context. Oracle
recommends that you do not set this parameter. Set this parameter only if you want to
limit the search scope for Active Directory users and groups. For example:

```
DEFAULT ADMIN CONTEXT = "OU=sales, DC=production, DC=examplecorp, DC=com"
```

• DIRECTORY_SERVER_TYPE, which determines the LDAP server access. You must set it to AD for Active Directory. Enter this value in upper case.

```
DIRECTORY SERVER TYPE = AD
```

Related Topics

About Using an Idap.ora File
 You can use an Idap.ora file to specify Active Directory servers for centrally managed
 users.

6.2.2.5 Step 5: Request an Active Directory Certificate for a Secure Connection

After you have configured the dsi.ora or ldap.ora file, you are ready to prepare Microsoft Active Directory and Oracle Database certificates for a secure connection.

Request the Active Directory certificate from an Active Directory administrator.

Related Topics

Management of Certificate Revocation Lists (CRLs) with orapki Utility
 You must manage certificate revocation lists (CRLs) with the orapki utility.



6.2.2.6 Step 6: Create the Wallet for a Secure Connection

After you have copied the Active Directory certificate, you are ready to add it to the Oracle wallet.

 Copy the certificate text file (for example, AD_CA_Root_cert.txt) from the Active Directory server to a temporary directory (for example, /tmp) on the local host.

If wallet location is not specified in the sqlnet.ora file, then the database will search the following locations in this order for the wallet. The directory location may need to be created.

For a non-multitenant database, or for the CDB root container of a multitenant database:

- a. \$ORACLE BASE/admin/db unique name/wallet/
- b. \$ORACLE HOME/admin/db unique name/wallet/

For a PDB in a multitenant database:

- a. \$ORACLE BASE/admin/db unique name/pdb guid/wallet/
- b. \$ORACLE HOME/admin/db unique name/pdb guid/wallet/

Oracle recommends that for each individual container in a multitenant database, you place the wallet files in the default wallet location under <code>\$ORACLE_BASE</code>, that is, in the <code>\$ORACLE_BASE/admin/db unique name/pdb guid/wallet/directory</code>.

To find the db unique name, connect to the CDB root and execute the following query:

```
SELECT DB UNIQUE NAME FROM V$DATABASE;
```

To find the pdb guid, from the CDB root, execute the following query:

```
SELECT PDB NAME, GUID FROM DBA PDBS;
```

If you are using sqlnet.ora to specify the wallet location, then the wallet location specified is for a non-multitenant database, or the root container of a multitenant database. For each PDB of the multitenant database, its wallet is located at

WALLET_LOCATION_specified_in_sqlnet.ora/pdb_guid. You can also place an individual PDB dsi.ora in WALLET LOCATION specified in sqlnet.ora/pdb guid.

2. Create a new wallet.

The following command creates an auto-login wallet in the specified path.

```
orapki wallet create -wallet path\_of\_wallet -auto\_login Enter password: password
```

Create an entry in wallet with the user name of the Oracle service directory user account for performing searches in Active Directory (created in the first step).

For example:

```
\verb|mkstore -wrl|| path\_of\_wallet - \verb|createEntry ORACLE.SECURITY.USERNAME oracle| \\
```

Create an entry in wallet with the DN of the Oracle service directory user account.

For example:

```
mkstore -wrl path_of_wallet -createEntry ORACLE.SECURITY.DN
cn=oracle,cn=users,dc=production,dc=examplecorp,dc=com
```

In this example, the DN indicates that the DNS domain is production.examplecorp.com. The Windows domain name is just production.



Create an entry in wallet with the user password credential of the Oracle service directory user account.

For example:

```
mkstore -wrl path of wallet -createEntry ORACLE.SECURITY.PASSWORD password
```

6. Add the certificate to the wallet. Use the Active Directory certificate that you received from the Active Directory administrator.

For example:

```
orapki wallet add -wallet path of wallet -cert /tmp/AD CA Root cert.txt -trusted cert
```

If WALLET_LOCATION is specified in sqlnet.ora, then you must add Active Directory certificates to the PDB specific wallet location (that is,

WALLET_LOCATION_specified_in_sqlnet.ora/pdb_guid, for each individual PDB). You can also add the Active Directory certificate to the

WALLET_LOCATION_specified_in_sqlnet.ora. However, it will only be effective for the root container, not for the entire CDB.

7. Verify the credentials.

For example:

```
orapki wallet display -wallet path_of_wallet
```

The output should be similar to the following:

```
Requested Certificates:
User Certificates:
Oracle Secret Store entries:
ORACLE.SECURITY.DN
ORACLE.SECURITY.PASSWORD
ORACLE.SECURITY.USERNAME
Trusted Certificates:
Subject: CN=ADSVR,DC=production,DC=examplecorp,DC=com
```

Changes to the wallet take effect immediately and do not require a database restart.

6.2.2.7 Step 7: Configure the Microsoft Active Directory Connection

Next, you are ready to connect the database to Active Directory using the settings you have so far.

- About Configuring the Microsoft Active Directory Connection
 To configure the Microsoft Active Directory connection, you can set the parameters in the database or use DBCA.
- Configuring the Access Manually Using Database System Parameters
 You can configure the Active Directory services connection manually by using LDAP-specific Oracle Database system parameters.
- Configuring the Access Using the Database Configuration Assistant GUI
 Oracle Database Configuration Assistant (DBCA) completes the LDAP connection
 configuration and automatically creates the wallet and stores the Active Directory
 certificate for use. DBCA only works when ldap.ora is configured for CMU-Active
 Directory.
- Configuring the Access Using Database Configuration Assistant Silent Mode
 Assuming ldap.ora (not dsi.ora) has been created in the correct location and configured
 properly, DBCA silent mode can create a new database or alter an existing database for
 the Microsoft Active Directory-Oracle Database integration.

6.2.2.7.1 About Configuring the Microsoft Active Directory Connection

To configure the Microsoft Active Directory connection, you can set the parameters in the database or use DBCA.

DBCA only recognizes the ldap.ora that is configured for centrally managed users, and only creates the wallet in the recommended default location. To use the default wallet locations, you must **not** set WALLET_LOCATION in sqlnet.ora.



Oracle recommends using dsi.ora for CMU-Active Directory.

Related Topics

Configuring the Access Manually Using Database System Parameters
 You can configure the Active Directory services connection manually by using LDAP-specific Oracle Database system parameters.

6.2.2.7.2 Configuring the Access Manually Using Database System Parameters

You can configure the Active Directory services connection manually by using LDAP-specific Oracle Database system parameters.

- 1. Ensure that you have created the dsi.ora file or the ldap.ora file, and that you have created the wallet.
- 2. Log in to the database instance as a user who has the ALTER SYSTEM system privilege.

For example, in a non-multitenant database:

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

In a multitenant environment, log in to the appropriate PDB.

```
sqlplus sec_admin@pdb_name
Enter password: password
```

To find the available PDBs in a CDB, log in to the CDB root container and then query the $\mathtt{PDB_NAME}$ column of the $\mathtt{DBA_PDBS}$ data dictionary view. To check the current container, run the show con name command.

3. Modify the LDAP_DIRECTORY_ACCESS parameter, which determines the type of LDAP directory access.

If you are using a multitenant environment, then set LDAP_DIRECTORY_ACCESS in each PDB, not in the CDB root. Setting this parameter in the CDB root will apply it only to the root, not to the PDBs.

Valid values are PASSWORD and NONE (to disable the connection). PASSWORD requires an Active Directory server certificate and when you create the wallet, you must include the credentials for the Active Directory service user account for Oracle.

For example:

ALTER SYSTEM SET LDAP_DIRECTORY_ACCESS = 'PASSWORD';

You can also set this parameter in the spfile or in the init.ora file (if the init.ora file is used). Afterward, restart the database.

4. Set the LDAP_DIRECTORY_SYSAUTH parameter to YES, so that administrative users from Active Directory can log in to Oracle Database with the SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, or SYSRAC administrative privilege.

Set LDAP_DIRECTORY_SYSAUTH in each PDB, not in the CDB root. Setting this parameter in the CDB root will apply it only to the root, not to the PDBs.

If you set this parameter to NO, then centrally managed users from Active Directory cannot log in to Oracle database with these privileges.

```
ALTER SYSTEM SET LDAP_DIRECTORY_SYSAUTH = YES SCOPE=SPFILE ;
```

You can also set this parameter in the spfile or in the init.ora file (if the init.ora file is used). Afterward, restart the database.

- Restart the database instance or re-open the PDB.
 - If you are in a non-multitenant environment, then restart the database:

```
SHUTDOWN IMMEDIATE STARTUP
```

If you are in a multitenant environment, then close and re-open the PDB:

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

After you restart the database or re-open the PDB, you can log in with the SYSDBA administrative privilege and check the LDAP parameters settings as follows:

```
show parameter ldap
```

6.2.2.7.3 Configuring the Access Using the Database Configuration Assistant GUI

Oracle Database Configuration Assistant (DBCA) completes the LDAP connection configuration and automatically creates the wallet and stores the Active Directory certificate for use. DBCA only works when ldap.ora is configured for CMU-Active Directory.

These instructions assume that you have already installed the Oracle software and that you are using an <code>ldap.ora</code> file (not <code>dsi.ora</code>) to identify the Active Directory servers for the centrally managed users. If you have not installed the database software yet, then you can install the software using Oracle Universal Installer (OUI). After that, use DBCA to create the database, and at the same time you can configure the connection for Active Directory centrally managed users.

- 1. Log in to the host where the Oracle database software is installed as a user who has administrative privileges.
- 2. Start DBCA.

By default, the DBCA utility is located in the \$ORACLE HOME/bin directory.

For example:

```
cd $ORACLE_HOME/bin
./dbca
```

3. Select the Network Configuration option (or when you get to the Network Configuration option when creating the database).

The Specify Network Configuration Details window appears. If the Directory Service Integration area is not visible, then the ldap.ora file was not configured correctly. Check

the ldap.ora configuration that you did earlier, and after you have corrected the file, rerun DBCA.

- 4. In the Directory Service Integration area, do the following:
 - In the Service username field, enter the name of the Oracle service directory user account.
 - In the Password field, enter the password of the Oracle service directory user account.
 - In the Service user DN field, enter the DN for the Oracle service directory user account. The DN can be retrieved directly from the Active Directory server or from an Active Directory system administrator.
 - For Access Type, select the type of authentication from the list (for example, PASSWORD). (This setting sets the LDAP_DIRECTORY_ACCESS parameter.) If necessary, select the Allow admin privileges authentication checkbox, which allows Active Directory users to authenticate and use database schemas with administrative privileges (for example, SYSDBA, SYSOPER, SYSBACKUP, and so on). Otherwise, centrally managed users from Active Directory cannot log in to the database with administrative privileges. (This setting corresponds to the LDAP_DIRECTORY_SYSAUTH parameter.)
 - Provide the path to the Active Directory certificate in the Certificate file location field.
 In a multitentant environment, DBCA recognizes and sets up Active Directory connections for the database instance connection. You must manually configure PDB connections if you want to connect a different Active Directory server to a PDB.
 - In the Wallet password and Confirm password fields, enter and confirm the
 password for the Oracle wallet that will store the certificate and credential of the Oracle
 service directory user account. Afterward, DBCA automatically validates the service
 directory user account, creates the wallet, stores the user credential, and imports the
 certificate.
- 5. Click **Next** until you reach the Finish page.
- Click Finish.

Related Topics

- Step 4: Create the dsi.ora or Idap.ora File
 The dsi.ora and ldap.ora files specify connections for centrally managed users for Active
 Directory.
- Configuring the Access Using Database Configuration Assistant Silent Mode
 Assuming ldap.ora (not dsi.ora) has been created in the correct location and configured
 properly, DBCA silent mode can create a new database or alter an existing database for
 the Microsoft Active Directory-Oracle Database integration.

6.2.2.7.4 Configuring the Access Using Database Configuration Assistant Silent Mode

Assuming ldap.ora (not dsi.ora) has been created in the correct location and configured properly, DBCA silent mode can create a new database or alter an existing database for the Microsoft Active Directory-Oracle Database integration.

- 1. Log in to the host that will have the Oracle database to be used for the integration.
- 2. Make sure ldap.ora is created with the correct content in a correct location.
- 3. Make sure that the WALLET LOCATION parameter is not specified in the sqlnet.ora file.
- 4. Run Database Configuration Assistant (DBCA) in silent mode.
 For example, to create a single instance non-multitenant database:



```
cd $ORACLE_HOME/bin

./dbca -silent -createDatabase -gdbName inst1.production.examplecorp.com
-templateName General_Purpose.dbc -totalMemory 1000
-registerWithDirService true
-dirServiceUser oracle
-dirServiceUserName cn=oracle,cn=users,dc=production,dc=examplecorp,dc=com
-dirServicePassword service_user_password
-ldapDirectoryAccessType PASSWORD
-useSysAuthForLDAPAccess true
-dirServiceCertificatePath /tmp/AD_CA_Root_cert.txt
-walletPassword wallet_password
-sysPassword sys_password
-systemPassword system password
```

To configure the root container of a CDB or a non-multitenant database:

```
cd $ORACLE_HOME/bin

./dbca -silent -configureDatabase -sourceDB db_name
-registerWithDirService true
-dirServiceUser oracle
-dirServiceUserName cn=oracle, cn=users, dc=production, dc=examplecorp, dc=com
-dirServicePassword service_user_password
-ldapDirectoryAccessType PASSWORD
-useSYSAuthForLDAPAccess true
-dirServiceCertificatePath /tmp/AD_CA_Root_cert.txt
-walletPassword wallet password
```

To configure a pluggable database in a CDB:

```
cd $ORACLE_HOME/bin

./dbca -silent -configurePluggableDatabase -pdbName pdb_name -sourceDB db_name
-registerWithDirService true
-dirServiceUser oracle
-dirServiceUserName cn=oracle, cn=users, dc=production, dc=examplecorp, dc=com
-dirServicePassword service_user_password
-dirServiceCertificatePath /tmp/AD_CA_Root_cert.txt
-walletPassword wallet password
```

Related Topics

About Using an Idap.ora File

You can use an ldap.ora file to specify Active Directory servers for centrally managed users.

6.2.2.8 Step 8: Verify the Oracle Wallet

The orapki utility can verify that the wallet for this database was created successfully.

- 1. Log in to the host where a database is used in the integration.
- 2. Go to the directory that contains the wallet.

If WALLET_LOCATION is not set in sqlnet.ora, then the default wallet locations are the following:

In a non-multitenant environment, the wallet directory is in the <code>\$ORACLE_BASE/admin/db unique name/wallet directory</code>.

In a multitenant environment, it is in one of the following locations:



- For the CDB root, the wallet is in the \$ORACLE_BASE/admin/db_unique_name/wallet/directory.
- For a PDB, the wallet is in the \$ORACLE_BASE/admin/db_unique_name/pdb_guid/ wallet/ directory.
- 3. At the command line, enter the following commands:

ls -ltr wallet location (to check that the wallet directory contains wallet files)

For example:

```
$ ls -ltr $ORACLE_BASE/admin/db_unique_name/pdb_guid/wallet/
total 12
-rw----- 1 creator_user creator_group 1597 Nov 27 22:47 cwallet.sso
-rw----- 1 creator_user creator_group 1552 Nov 27 22:47 ewallet.p12
-rw-rw-r-- 1 creator_user creator_group 86 Nov 27 22:48 dsi.ora
```

orapki wallet display -wallet wallet_location (to find the Oracle Secret Store entries)

The output should contain the following entries:

```
Requested Certificates:
User Certificates:
Oracle Secret Store entries:
ORACLE.SECURITY.DN
ORACLE.SECURITY.PASSWORD
ORACLE.SECURITY.USERNAME
Trusted Certificates:
Subject: CN=ADSVR,DC=production,DC=examplecorp,DC=com
```

6.2.2.9 Step 9: Test the Integration

To test the integration, you must set the <code>ORACLE_HOME</code>, <code>ORACLE_BASE</code>, and <code>ORACLE_SID</code> environment variables and then verify the LDAP parameter settings.

- 1. Log in to the host where a database is used for the integration.
- 2. Set the ORACLE HOME, ORACLE BASE, and ORACLE SID environment variables.

For example:

```
export ORACLE_HOME=/app/product/18.1/dbhome_1
export ORACLE_BASE=/app
export ORACLE_SID=sales_db
```

3. Log in to the database instance as a user who has the SYSDBA administrative privilege.

For example:

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

In a multitenant environment, log in to the appropriate PDB. For example:

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

To find the available PDBs in a CDB, log in to the CDB root container and then query the $\tt PDB_NAME$ column of the $\tt DBA_PDBS$ data dictionary view. To check the current container, run the show con name command.

Check the LDAP parameter settings:

show parameter ldap

The output should be similar to the following:

NAME	TYPE	VALUE
ldap_directory_access	string	PASSWORD
ldap directory sysauth	string	YES

6.3 Configuring Authentication for Centrally Managed Users

You can configure password authentication, Kerberos authentication, or public key infrastructure (PKI) authentication.

- Configuring Password Authentication for Centrally Managed Users
 Configuring password authentication for centrally managed users entails the use of a
 password filter with Active Directory to generate and store Oracle Database password
 verifiers on Active Directory.
- Configuring Kerberos Authentication for Centrally Managed Users
 If you plan to use Kerberos authentication, then you must configure Kerberos in the Oracle database that will be integrated with Microsoft Active Directory.
- Configuring Authentication Using PKI Certificates for Centrally Managed Users
 If you plan to use PKI certificates for the authentication of centrally managed users, then
 you must configure Transport Layer Security in the Oracle database that will be integrated
 with Microsoft Active Directory.

6.3.1 Configuring Password Authentication for Centrally Managed Users

Configuring password authentication for centrally managed users entails the use of a password filter with Active Directory to generate and store Oracle Database password verifiers on Active Directory.

- About Configuring Password Authentication for Centrally Managed Users

 To configure password authentication, you must deploy a password filter, extend the Active Directory schema by adding one user attribute, and create groups for generating different versions of password verifiers on Active Directory.
- Configuring Password Authentication for a Centrally Managed User
 You must perform password authentication configuration on Active Directory servers, and
 also on Oracle databases if it is required that Active Directory users will log in to Oracle
 databases with administrative privileges.
- Logging in to an Oracle Database Using Password Authentication
 For password authentication, centrally managed users have choices of how to log in to the database.

6.3.1.1 About Configuring Password Authentication for Centrally Managed Users

To configure password authentication, you must deploy a password filter, extend the Active Directory schema by adding one user attribute, and create groups for generating different versions of password verifiers on Active Directory.

For Active Directory users to log in Oracle database with administrative privileges, you must also set a password file with Oracle database.

For password authentication, because Oracle Database does not pass Active Directory users' passwords through the ldapbind command to authenticate with Active Directory, you must

install an Oracle filter and extend the Active Directory schema. The Oracle filter that you install in Active Directory creates Oracle-specific password verifiers when Active Directory users update their passwords. The Oracle filter does not generate all required Oracle password verifiers when it is first installed; the Oracle filter only generates the Oracle password verifier for a user when the user changes his or her Active Directory password.

To maintain backward compatibility (if your site requires it), the Oracle filter can generate password verifiers to work with Oracle Database clients for releases 11g, 12c, and 18c. The Oracle password filter uses Active Directory groups named ORA VFR MD5 (for WebDAV), ORA VFR 11G (for release 11q) and ORA VFR 12C (for releases 12c and 18c) to determine which Oracle Database password verifiers to generate. These groups must be created in Active Directory for the Oracle password verifiers to be generated for group member users. These are separate groups that dictate which specific verifiers should be generated for the Active Directory users. For example, if ten directory users need to log in to a newly created Oracle Database release 18c database that only communicated with Oracle Database release 18c and 12c clients, then an Active Directory group ORA VFR 12C will have ten Active Directory users as members. The Oracle filter will only generate 12c verifiers for these ten Active Directory users when they change passwords with Active Directory (18c verifiers are the same as 12c verifiers). If an Active Directory user no long needs to log in to Oracle databases, in order to clear the Oracle password verifiers generated for the Active Directory user, remove the user from any ORA VFR groups, and reset the password (or require password change) for this user. You can also manually clear the orclcommonAttribute attribute for this user. Oracle password verifiers will no longer be generated after the user has been removed from ORA VFR groups.

6.3.1.2 Configuring Password Authentication for a Centrally Managed User

You must perform password authentication configuration on Active Directory servers, and also on Oracle databases if it is required that Active Directory users will log in to Oracle databases with administrative privileges.

- 1. Deploy the Oracle Database password filter and extend the Active Directory schema.
 - The utility tool for performing this task, <code>opwdintg.exe</code>, is located in <code>SORACLE_HOME/bin</code>. This utility installs the password filter in Active Directory, extends the Active Directory schema to hold the Oracle password verifiers, and creates the Active Directory password verifier groups. The password filter will enable the Microsoft Active Directory user accounts to be authenticated by the Oracle database when connected to clients using <code>Webday</code>, <code>11G</code>, and <code>12C</code> password verifiers.
 - a. To deploy the <code>opwdintg.exe</code> executable, copy this file to the Active Directory server and then have the Active Directory administrator run the <code>opwdintg.exe</code> utility tool.
 - **b.** Log in to Microsoft Active Directory as a user who has privileges to create and manage user groups.
 - c. Check for the following password verifier user groups: ORA_VFR_MD5, ORA_VFR_11G, and ORA_VFR_12C. If these groups do not exist, then rerun the opwdintg.exe utility tool.
 - **d.** Add the Microsoft Active Directory users who will use Oracle Database to these groups, following these guidelines:
 - If either the client or the server only permits Oracle Database release 12c authentication, then add the user to the ORA_VFR_12C group. (Oracle Database release 18c uses the same verifier as Oracle Database release 12c.)
 - If both the client and the server only permit authentication lower than Oracle Database release 12c (that is, they have Oracle Database releases 11g, or 12.1.0.1 clients), then add the user to the ORA VFR 11G group.

If a user must authenticate through an Oracle Database WebDAV client, then the user must be a member of the ORA VFR MD5 group.

This configuration enables fine-grained control over the generation of the Oracle Database password verifiers. Only the required verifiers for the required users are generated. For example, if Microsoft Active Directory user pfitch is added to the ORA_VFR_12C and ORA_VFR_11G groups, then both the 12C and 11G verifiers will be generated for pfitch. This ensures that when applicable, the most secure and strongest verifier is chosen, while in other cases, the 11G verifier is chosen for the Oracle Database release 11g clients.

2. Update the database password file to version 12.2.

If it is required that Active Directory users will log in to Oracle databases with administrative privileges, then update the database password file to version 12.2.

- **a.** As a user with administrative privileges, log in to the host where the database that is to be used for the Microsoft Active Directory connection resides.
- b. Go to the \$ORACLE HOME/dbs directory.
- c. Run the ORAPWD utility to set the format to 12.2.

For example:

```
orapwd FILE='/app/oracle/product/18.1/db 1/dbs/orapwdb181' FORMAT=12.2
```

This setting ensures that you can grant the various administrative privileges such as SYSOPOER and SYSBACKUP to the global user.

- d. Log in to the database instance as a user who has the ALTER SYSTEM privilege.
- e. Make sure that the LDAP_DIRECTORY_SYSAUTH parameter is set to YES in the spfile or the init.ora file.
- f. Set the REMOTE_LOGIN_PASSWORDFILE parameter to EXCLUSIVE in the spfile or the init.ora file.
- g. Restart the database instance.

```
SHUTDOWN IMMEDIATE STARTUP
```

Related Topics

 Step 2: For Password Authentication, Install the Password Filter and Extend the Microsoft Active Directory Schema

You can use the Oracle <code>opwdintg.exe</code> executable on the Active Directory server to install the password filter and extend the Active Directory schema.

6.3.1.3 Logging in to an Oracle Database Using Password Authentication

For password authentication, centrally managed users have choices of how to log in to the database.

To log in to a database that is configured to connect to Active Directory, an Active Directory user can use the following logon user name syntax if he or she is using password authentication:

```
sqlplus /nolog
connect "Windows_domain\Active_Directory_user_name"@tnsname_of_database
Password: password
```



The following connection assumes the Windows domain name is production:

```
connect "production\pfitch"@inst1
```

If the Active Directory user is in the same Active Directory domain as the Oracle Service Directory User Account configured in the database wallet, then an Active Directory user can use this user name (samAccountName) directly to log on to the database:

```
sqlplus samAccountName@tnsname_of_database
Enter password: password
```

For example:

```
connect pfitch@instl
Enter password: password
```

Alternatively, the user can use their Active Directory Windows user logon name with the DNS domain name.

For example:

connect "pfitch@production.examplecorp.com"@inst1

6.3.2 Configuring Kerberos Authentication for Centrally Managed Users

If you plan to use Kerberos authentication, then you must configure Kerberos in the Oracle database that will be integrated with Microsoft Active Directory.

CMU-Active Directory only supports the Microsoft Active Directory Kerberos server. Other non-Active Directory Kerberos servers are not supported with CMU-Active Directory.



You do not create database users identified externally as an Active Directory user's Kerberos UPN. Instead, you use global users that are mapped to Active Directory users or groups.

Related Topics

- Mapping a Directory Group to a Shared Database Global User
 Most users of the database will be mapped to a shared global database user (schema) through membership in a directory group.
- Exclusively Mapping a Directory User to a Database Global User
 You can map a Microsoft Active Directory user exclusively to an Oracle Database global user.
- Enabling Kerberos Authentication
 To enable Kerberos authentication for Oracle Database, you must first install it, and then follow a set of configuration steps.



6.3.3 Configuring Authentication Using PKI Certificates for Centrally Managed Users

If you plan to use PKI certificates for the authentication of centrally managed users, then you must configure Transport Layer Security in the Oracle database that will be integrated with Microsoft Active Directory.

While Kerberos authentication with CMU requires use of the Microsoft Active Directory-Active Directory Kerberos server, PKI authentication can use third-party CA services, not just the one with Microsoft Active Directory-Active Directory.



You use an Active Directory user certificate when you configure Transport Layer Security Authentication. However, you do not create database users identified externally as the DN of the Active Directory user certificate. Instead, you use global users that are mapped to Active Directory users or groups.

Related Topics

- Mapping a Directory Group to a Shared Database Global User
 Most users of the database will be mapped to a shared global database user (schema) through membership in a directory group.
- Exclusively Mapping a Directory User to a Database Global User
 You can map a Microsoft Active Directory user exclusively to an Oracle Database global user.
- Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.
- Public Key Infrastructure in an Oracle Environment
 A public key infrastructure (PKI) is a substrate of network components that provide a security underpinning, based on trust assertions, for an entire organization.

6.4 Configuring Authorization for Centrally Managed Users

With centrally managed users, you can manage the authorization for Active Directory users to access Oracle databases.

Users can be added, modified, or dropped from an organization by using Active Directory without your having to add, modify, or drop the user from every database in your organization.

- About Configuring Authorization for Centrally Managed Users
 You can manage user authorization for a database within Active Directory.
- Mapping a Directory Group to a Shared Database Global User
 Most users of the database will be mapped to a shared global database user (schema) through membership in a directory group.
- Mapping a Directory Group to a Global Role
 Database global roles mapped to directory groups give member users additional privileges
 and roles above what they have been granted through their login schemas.



- Exclusively Mapping a Directory User to a Database Global User
 You can map a Microsoft Active Directory user exclusively to an Oracle Database global user.
- Altering or Migrating a User Mapping Definition
 You can update an Active Directory user to a Database global user mapping by using the ALTER USER statement.
- Configuring Administrative Users
 Administrative users can work as they have in the past, but with CMUs, they can be controlled with centralized authentication and authorization if they are using shared schemas.
- Verifying the Centrally Managed User Logon Information
 After you configure and authorize a centrally managed user, you can verify the user logon information by executing a set of SQL queries on the Oracle database side.

6.4.1 About Configuring Authorization for Centrally Managed Users

You can manage user authorization for a database within Active Directory.

Most Oracle Database users will be mapped to a shared database schema (user). This minimizes the work that must be done in each Oracle database when directory users are hired, change jobs within the company, or leave the company. A directory user will be assigned to an Active Directory group that is mapped to an Oracle database global user (schema). When the user logs into the database, the database will guery Active Directory to find the groups the user is a member of. If your deployment is using shared schemas, then one of the groups will map to a shared database schema and the user will be assigned to that database schema. The user will have the roles and privileges that granted to the database schema. Because multiple users will be assigned to the same shared database schema, only the minimal set of roles and privileges should be granted to the shared schema. In some cases, no privileges and roles should be granted to the shared schema. Users will be assigned the appropriate set of roles and schemas through database global roles. Global roles are mapped to Active Directory groups. This way, different users can have different roles and privileges even if they are mapped to the same database shared schema. A newly hired user will be assigned to an Active Directory group mapped to a shared schema and then to one or more additional groups mapped to global roles to gain the additional roles and privileges required to complete their tasks. The combination of shared schemas and global roles allows for centralized authorization management with minimal changes to the database operationally. The database must be initially provisioned with the set of shared schemas and global roles mapped to the appropriate Active Directory groups, but then user authorization management can happen within Active Directory.

An Active Directory user can also be exclusively mapped to a database global user. This requires a new user in the database that is mapped directly to the Active Directory user. New users and departing users will require updates to each database they are members of.

Active Directory users requiring administrative privileges such as SYSOPER and SYSBACKUP cannot be granted these through global roles. Administrative privileges can only be granted to a schema and not a role. But even in these cases with administrative privileges, shared schemas can be used to provide ease of user authorization management. Using a shared schema with the SYSOPER privilege will allow new users to be easily added to the Active Directory group mapped to the schema with SYSOPER without having to create a new user schema in the database. Even if only one user is assigned to the shared schema, it can still be managed centrally.

When using global roles to grant privileges and roles to the user, remember that the maximum number of enabled roles in a session is 150.

The following types of global user mappings are supported for authorization:

- Map shared global users, in which directory users are assigned to a shared database schema (user) through the mapping of a directory group to the shared schema. The directory users that are members of the group can connect to the database through this shared schema. Use of shared schemas allows for centralized management of user authorization in Active Directory.
- Exclusive global user mappings, in which a dedicated database user is exclusively mapped to a directory user. Not as common as the shared database schema, this user is created for direct database access by using either SQL*Plus or the schema user for two-tier or three-tier applications. Oracle recommends that you grant database privileges to these users through global roles, which facilitates authorization management. However, these users can also have direct privilege grants in the Oracle database, although this is not recommended. This is because two-tier and three-tier applications can use the global user as the database schema, so the global user has the full database privileges on the schema objects as the owner.

It is common for a directory user to be a member of multiple groups. However, only one of these groups should be mapped to a shared schema.

6.4.2 Mapping a Directory Group to a Shared Database Global User

Most users of the database will be mapped to a shared global database user (schema) through membership in a directory group.

The Active Directory group must be created before the database global user can be mapped to it. You can add Active Directory users to the group at any time before the user needs to log in to the database. On the database side, you must have the CREATE USER and ALTER USER privileges to perform these mappings. This configuration can be used for users who have the password authentication, Kerberos authentication, and public key infrastructure (PKI) authentication methods.

You can assign users who share the same database schema for an application into an Active Directory group. A shared Oracle Database global user (that is, a shared schema) is mapped to an Active Directory group. This way, any Active Directory user of this group can log in to the database through that shared global user account. Although the database global user account is shared by group members, the Active Directory user's authenticated identity (Windows domain and his or her samAccountName), and enterprise identity (DN) are tracked and audited inside the database.

- 1. Log in to the database instance as a user who has been granted the CREATE USER Or ALTER USER system privilege.
- 2. Execute the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the DN of an Active Directory group.

For example, to map a directory group named widget_sales_group in the sales organization unit of the production.examplecorp.com domain to a shared database global user named WIDGET SALES:

```
CREATE USER widget_sales IDENTIFIED GLOBALLY AS 'CN=widget sales group,OU=sales,DC=production,DC=examplecorp,DC=com';
```

All members of the widget_sales_group will be assigned to the widget_sales shared schema when they log in to the database.



6.4.3 Mapping a Directory Group to a Global Role

Database global roles mapped to directory groups give member users additional privileges and roles above what they have been granted through their login schemas.

- 1. Log in to the database instance as a user who has been granted the CREATE ROLE or ALTER ROLE system privilege.
- 2. Execute the CREATE ROLE or ALTER ROLE statement with the IDENTIFIED GLOBALLY AS clause specifying the DN of an Active Directory group.

For example, to map a directory user group named $widget_sales_group$ in the sales organization unit of the production.examplecorp.com domain to a database global role WIDGET SALES ROLE:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS 'CN=widget sales group,OU=sales,DC=production,DC=examplecorp,DC=com';
```

In a multitenant environment, to create a common role called C##WIDGET SALES ROLE:

```
CREATE ROLE c##widget_sales_role IDENTIFIED GLOBALLY AS
'CN=widget_sales_group,OU=sales,DC=production,DC=examplecorp,DC=com'
CONTAINER = ALL;
```

All members of the widget_sales_group will be authorized with the database role widget sales role when they log in to the database.

6.4.4 Exclusively Mapping a Directory User to a Database Global User

You can map a Microsoft Active Directory user exclusively to an Oracle Database global user.

You perform the configuration on the Oracle Database side only, not the Active Directory side. You must have the CREATE USER and ALTER USER privileges to perform these mappings. This configuration can be used for users who have the password authentication, Kerberos authentication, and public key infrastructure (PKI) authentication methods.

- 1. Log in to the database instance as a user who has been granted the CREATE USER or ALTER USER system privilege.
- 2. Execute the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the DN of an Active Directory user.

For example, to map an existing Active Directory user named Peter Fitch (whose samAccountName is pfitch) in the sales organization unit of the production.examplecorp.com domain to a database global user named PETER_FITCH:

```
CREATE USER peter_fitch IDENTIFIED GLOBALLY AS 'CN=Peter Fitch,OU=sales,DC=production,DC=examplecorp,DC=com';
```

6.4.5 Altering or Migrating a User Mapping Definition

You can update an Active Directory user to a Database global user mapping by using the ALTER USER statement.

You can update users whose accounts were created using any of the CREATE USER statement clauses: IDENTIFIED BY password, IDENTIFIED EXTERNALLY, or IDENTIFIED GLOBALLY. This is useful when migrating users to using CMU. For example, a database user that is externally authenticated to Kerberos will be identified by their user principal name (UPN). To migrate the

user to use CMU with Kerberos authentication, you would need to run the ALTER USER statement to declare a global user and identify the user with their Active Directory distinguished name (DN).

- Log in to the database instance as a user who has been granted the ALTER USER system privilege.
- 2. Execute the ALTER USER statement with the IDENTIFIED GLOBALLY AS clause.

For example:

```
ALTER USER peter_fitch IDENTIFIED GLOBALLY AS 'CN=Peter Fitch,OU=sales,DC=production,DC=examplecorp,DC=com';
```

6.4.6 Configuring Administrative Users

Administrative users can work as they have in the past, but with CMUs, they can be controlled with centralized authentication and authorization if they are using shared schemas.

- Configuring Database Administrative Users with Shared Access Accounts
 Using shared accounts simplifies the management of database administrators for multiple
 databases as they join, move, and leave the organization.
- Configuring Database Administrative Users Using Exclusive Mapping Database administrators can also be mapped to exclusive schemas in databases.

6.4.6.1 Configuring Database Administrative Users with Shared Access Accounts

Using shared accounts simplifies the management of database administrators for multiple databases as they join, move, and leave the organization.

You can assign new database administrators to shared accounts in multiple databases using Active Directory groups without having to create new Oracle database accounts.

1. Ensure that the password file for the current database instance is in the 12.2 format.

```
orapwd file=pwd_file FORMAT=12.2
Enter password for SYS: password
```

- 2. In Active Directory, create an Active Directory group (for example, for a database administrator backup users group called ad dba backup users).
- 3. In Oracle Database, create a global user (shared schema) (for example, db_dba_backup_global_user) and map this user to the Active Directory ad dba backup users group.
- Grant the SYSBACKUP administrative privilege to the global user db_dba_backup_global_user.

At this stage, any Active Directory user who is added to the ad_dba_backup_users Active Directory group will be assigned to the new database shared schema with the SYSBACKUP administrative privilege.

6.4.6.2 Configuring Database Administrative Users Using Exclusive Mapping

Database administrators can also be mapped to exclusive schemas in databases.

1. Ensure that the password file for the current database instance is in the 12.2 format.

```
orapwd file=pwd_file FORMAT=12.2
Enter password for SYS: password
```



- 2. Log in to the database instance as a user who can create users and grant administrative privileges to other users.
- Create a database global user.

For example:

```
CREATE USER peter_fitch IDENTIFIED GLOBALLY AS
'CN=Peter Fitch,OU=sales,DC=production,DC=examplecorp,DC=com';
```

4. Grant this user the administrative privilege.

For example, to grant a user the SYSKM administrative privilege:

```
GRANT SYSKM TO peter fitch;
```

Due to the amount of work to maintain accounts and the mapping in both the database and Active Directory, a more centralized approach would be to use shared schemas for these administrative accounts as well, even if only one Active Directory user is assigned to the shared database account in some cases.

6.4.7 Verifying the Centrally Managed User Logon Information

After you configure and authorize a centrally managed user, you can verify the user logon information by executing a set of SQL queries on the Oracle database side.

1. Log in to the database as a centrally managed user from Active Directory that you have just configured and authorized.

For example, to log in to the database instance inst1 as the enterprise user pfitch, who is on the Windows domain production:

```
sqlplus /nolog
connect "production\pfitch"@inst1
Enter password: password
```

2. Verify the mapped global user.

The mapped global user is the database user account that has the centrally managed user authorization. User PETER_FITCH is considered a global user with exclusive mapping for the Active Directory user pfitch, while user WIDGET_SALES is considered a global user with shared mapping for Active Directory group widget_sales_group of which pfitch is a member. A global user account has its own schema.

```
SHOW USER;
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

```
USER is "PETER_FITCH"

Or

USER is "WIDGET SALES"
```

3. Find the roles that have been granted to the centrally managed user.

```
SELECT ROLE FROM SESSION ROLES ORDER BY ROLE;
```

Output similar to the following appears:



- 4. Execute the following queries to check the SYS_CONTEXT namespace values for the current schema being used in this database session, current user name, session user name, authentication method, authenticated identity, enterprise identity, identification type, and LDAP server type.
 - Verify the current schema that is being used in this database session. A database schema is an object container that identifies the objects it contains. The current schema is the default container for objects name resolution in this database session.

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

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

 Verify the current user. In this case, the current user is the same as the current schema.

```
SELECT SYS CONTEXT('USERENV', 'CURRENT USER') FROM DUAL;
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

Verify the session user.

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

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

Verify the authentication method.

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



Output similar to the following appears:

Verify the authenticated identity for the enterprise user. The Active Directory authenticated user identity is captured and audited when this user logs on to the database.

```
SELECT SYS CONTEXT('USERENV', 'AUTHENTICATED IDENTITY') FROM DUAL;
```

Output similar to the following appears:

Verify the centrally managed user's enterprise identity.

```
SELECT SYS CONTEXT ('USERENV', 'ENTERPRISE IDENTITY') FROM DUAL;
```

Output similar to the following appears:

Verify the identification type.

```
SELECT SYS CONTEXT('USERENV', 'IDENTIFICATION TYPE') FROM DUAL
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

Or

Verify the LDAP server type.

```
SELECT SYS CONTEXT('USERENV', 'LDAP SERVER TYPE') FROM DUAL;
```

Output similar to the following appears. In this case, the LDAP server type is Active Directory.

```
SYS_CONTEXT('USERENV','LDAP_SERVER_TYPE')
-----AD
```

Related Topics

Logging in to an Oracle Database Using Password Authentication
 For password authentication, centrally managed users have choices of how to log in to the database.

6.5 Troubleshooting Centrally Managed Users

Oracle provides error messages that help you troubleshoot common errors that may arise when a Microsoft Active Directory user tries to log in to an Oracle database.

ORA-28276 Connection Errors

The ORA-28276: Invalid ORACLE password attribute error can result from an improperly set orclCommonAttribute attribute.

ORA-01017 Connection Errors

The ORA-01017: invalid username/password logon denied error can be generated due to the differences in how special characters are allowed in Oracle Database and in Microsoft Active Directory.

ORA-28274 Connection Errors

The ORA-28274: No ORACLE password attribute corresponding to user nickname exists error is generated due to problems with the Active Directory schema or the Oracle service directory.

ORA-28300 Connection Errors

The ORA-28030: No permission to read user entry in LDAP directory service error is generated due to permissions problems with the Oracle service directory.

Using Trace Files to Diagnose CMU Connection Errors
 The trace setting gdsi tracks centrally managed users (CMU) connection errors.

6.5.1 ORA-28276 Connection Errors

The ORA-28276: Invalid ORACLE password attribute error can result from an improperly set orclCommonAttribute attribute.

For example:

```
SQL> connect "myad\dev"@orcl_db
Enter password: password

ERROR:
ORA-28276: Invalid ORACLE password attribute.
```

This error occurs when the <code>orclCommonAttribute</code> attribute has not been correctly populated with user password. For example:

```
$ ldapsearch -h <AD_Server> -p 389 -D
"cn=oracleservice,cn=users,dc=myad,dc=example,dc=com" -w **** -U 2 -W
"file:wallet_path"
-P password -b "dc=myad,dc=example,dc=com" -s sub "(sAMAccountName=def*)"
dn orclCommonAttributeCN=def,CN=Users,DC=myad,DC=example,DC=com
orclCommonAttribute=
```

To remedy this problem:

1. Run the opwdintg.exe to install the password filter on every Windows domain controller in the domain for Active Directory.

- Restart each Windows domain controller server. Each Windows domain controller must be restarted after you install the password filter. Otherwise, the password filter will not work on the Windows domain controller.
- 3. Assign the Active Directory users to the appropriate ORA_VFR group.
- Reset the user password on Active Directory.
- 5. Run ldapsearch to check that the password has been generated.

6.5.2 ORA-01017 Connection Errors

The ORA-01017: invalid username/password logon denied error can be generated due to the differences in how special characters are allowed in Oracle Database and in Microsoft Active Directory.

User names and passwords that centrally managed users (CMU) create follow different creation rules than the rules for Oracle Database user names and passwords. To remedy the problem of ORA-01017 errors, enclose the Active Directory user's user name and password in double quotation marks. For example, for an Active Directory user whose user name is peter fitch and whose password is <code>ILoveMySalads@_home!</code>, and who is in the same domain as the Oracle service user, the following login works:

```
CONNECT "peter fitch"/"ILoveMySalads@ home!"@orcl
```

If the Active Directory user is in a different domain than the Oracle service user, then the Windows domain (EXAMPLE in this case) must be included in the user name:

```
CONNECT "EXAMPLE\peter fitch"/"ILoveMySalads@_home!"@orcl
CONNECT "EXAMPLE\peter fitch"@orcl
Enter password: password
```

Note that for the password entered at the <code>Enter password</code> prompt, there are 22 characters in all: 20 characters for the <code>ILoveMySalads@_home!</code> password, plus two characters for the two double quotation marks.

6.5.3 ORA-28274 Connection Errors

The ORA-28274: No ORACLE password attribute corresponding to user nickname exists error is generated due to problems with the Active Directory schema or the Oracle service directory.

The Active Directory schema may not have been extended or it was populated poorly. Alternatively, the Oracle service directory user does not have required permissions to access the orclCommonAttribute attribute of the user who tried to log in to Oracle database.

To remedy this problem:

Solution 1:

- 1. Run the opwdintg.exe to install the password filter on every Windows domain controller in the domain for Active Directory.
- Restart each Windows domain controller server. Each Windows domain controller must be restarted after you install the password filter. Otherwise, the password filter will not work on the Windows domain controller.

- 3. Assign the Active Directory users to the appropriate ORA VFR group.
- 4. Reset the user password on Active Directory.
- 5. Run ldapsearch to check that the password has been generated.

Solution 2:

- 1. Grant the Oracle service directory user account the Read Properties and Write lockoutTime, which are permissions to access the properties of the Active Directory user who tries to log in to the database.
- Set permissions for Control Access on the orclCommonAttribute of the Active Directory users.

Related Topics

 Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions

The Oracle service directory user account is for the interaction between Oracle Database and the LDAP directory service.

6.5.4 ORA-28300 Connection Errors

The ORA-28030: No permission to read user entry in LDAP directory service error is generated due to permissions problems with the Oracle service directory.

You can track this error using the CMU trace. For example:

```
2023-03-27 19:51:55.0 - KZLG_ERR: failed to modify user status Insufficient access 2023-03-27 17:57:27.0 - KZLG ERR: LDAPERR=50, OER=28300
```

To remedy this problem, In addition), and also the permission

- 1. Grant the Oracle service directory user account the Read Properties and Write lockoutTime, which are permissions to access the properties of the Active Directory user who tries to log in to the database.
- Set permissions for Control Access on the orclCommonAttribute of the Active Directory users.

Related Topics

 Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions

The Oracle service directory user account is for the interaction between Oracle Database and the LDAP directory service.

Using Trace Files to Diagnose CMU Connection Errors
 The trace setting gdsi tracks centrally managed users (CMU) connection errors.

6.5.5 Using Trace Files to Diagnose CMU Connection Errors

The trace setting gdsi tracks centrally managed users (CMU) connection errors.



As a user who has the ALTER SYSTEM privilege and the SYSDBA administrative privilege, you can enable this trace event as follows:

```
ALTER SYSTEM SET EVENTS='TRACE[GDSI] DISK LOW';
```

After the Active Directory user tries to log in, and if the login fails, go to the directory that contains the trace files and grep these files for the connection errors.

```
grep -i kzlg *.trc
```

Then you can collect and review the trace file that contains the detailed information.

To disable tracing, you can enter the following command:

```
ALTER SYSTEM SET EVENTS='TRACE[GDSI] OFF';
```

6.6 Integration of Oracle Database with Microsoft Active Directory Account Policies

As part of the Oracle Database-Microsoft Active Directory integration, Oracle Database enforces the Active Directory account policies when Active Directory users log into the Oracle database.

Active Directory account policy settings cover the password policy, account lockout policy, and Kerberos policy. Oracle Database enforces all of the account policies for centrally managed users from Active Directory. For example, Oracle prevents Active Directory users with account status, such as password expired, password must change, account locked out, or account disabled from logging in to the database. If you are using Kerberos authentication, then Oracle prevents Active Directory users with expired Kerberos tickets from logging in the database. If you are using password authentication, then an Active Directory user account will be locked out for a specified period of time on Active Directory after the user makes a specified number of failed attempts consecutively when trying to log in to the Oracle database using incorrect passwords. With enforcing the account lockout policy, Oracle effectively prevents password guessing attacks against Active Directory user accounts.



Oracle supports only the Active Directory default domain policy, but not any fine-grained password policies. For example, if a password expiration is set in the default domain policy but the fine-grained password policy has a shorter expiration, then only the password expiration in default domain policy is honored with Active Directory users who access the Oracle database by using CMU with Active Directory.

6.7 Configuring Centrally Managed Users with Oracle Autonomous Database

You can deploy centrally managed users (CMU) on Oracle Autonomous Database.



For instructions on deploying CMU on Oracle Autonomous Database, see "Use Microsoft Active Directory with Autonomous Database" in *Using Oracle Autonomous Database Serverless*.

6.8 Troubleshooting Centrally Managed Users

Oracle provides error messages that help you troubleshoot common errors that may arise when a Microsoft Active Directory user tries to log in to an Oracle database.

ORA-28276 Connection Errors

The ORA-28276: Invalid ORACLE password attribute error can result from an improperly set orclCommonAttribute attribute.

ORA-01017 Connection Errors

The ORA-01017: invalid username/password logon denied error can be generated due to the differences in how special characters are allowed in Oracle Database and in Microsoft Active Directory.

ORA-28274 Connection Errors

The ORA-28274: No ORACLE password attribute corresponding to user nickname exists error is generated due to problems with the Active Directory schema or the Oracle service directory.

ORA-28300 Connection Errors

The ORA-28030: No permission to read user entry in LDAP directory service error is generated due to permissions problems with the Oracle service directory.

Using Trace Files to Diagnose CMU Connection Errors
 The trace setting gdsi tracks centrally managed users (CMU) connection errors.

6.8.1 ORA-28276 Connection Errors

The ORA-28276: Invalid ORACLE password attribute error can result from an improperly set orclCommonAttribute attribute.

For example:

```
SQL> connect "myad\dev"@orcl_db
Enter password: password

ERROR:
ORA-28276: Invalid ORACLE password attribute.
```

This error occurs when the orclCommonAttribute attribute has not been correctly populated with user password. For example:

```
$ ldapsearch -h <AD_Server> -p 389 -D
"cn=oracleservice,cn=users,dc=myad,dc=example,dc=com" -w **** -U 2 -W
"file:wallet_path"
-P password -b "dc=myad,dc=example,dc=com" -s sub "(sAMAccountName=def*)"
dn orclCommonAttributeCN=def,CN=Users,DC=myad,DC=example,DC=com
orclCommonAttribute=
```

To remedy this problem:

- Run the opwdintg.exe to install the password filter on every Windows domain controller in the domain for Active Directory.
- Restart each Windows domain controller server. Each Windows domain controller must be restarted after you install the password filter. Otherwise, the password filter will not work on the Windows domain controller.
- 3. Assign the Active Directory users to the appropriate ORA VFR group.
- 4. Reset the user password on Active Directory.
- 5. Run Idapsearch to check that the password has been generated.

6.8.2 ORA-01017 Connection Errors

The ORA-01017: invalid username/password logon denied error can be generated due to the differences in how special characters are allowed in Oracle Database and in Microsoft Active Directory.

User names and passwords that centrally managed users (CMU) create follow different creation rules than the rules for Oracle Database user names and passwords. To remedy the problem of ORA-01017 errors, enclose the Active Directory user's user name and password in double quotation marks. For example, for an Active Directory user whose user name is peter fitch and whose password is IloveMySalads@_home!, and who is in the same domain as the Oracle service user, the following login works:

```
CONNECT "peter fitch"/"ILoveMySalads@ home!"@orcl
```

If the Active Directory user is in a different domain than the Oracle service user, then the Windows domain (EXAMPLE in this case) must be included in the user name:

```
CONNECT "EXAMPLE\peter fitch"/"ILoveMySalads@_home!"@orcl
CONNECT "EXAMPLE\peter fitch"@orcl
Enter password: password
```

Note that for the password entered at the <code>Enter password</code> prompt, there are 22 characters in all: 20 characters for the <code>ILoveMySalads@_home!</code> password, plus two characters for the two double quotation marks.

6.8.3 ORA-28274 Connection Errors

The ORA-28274: No ORACLE password attribute corresponding to user nickname exists error is generated due to problems with the Active Directory schema or the Oracle service directory.

The Active Directory schema may not have been extended or it was populated poorly. Alternatively, the Oracle service directory user does not have required permissions to access the orclCommonAttribute attribute of the user who tried to log in to Oracle database.

To remedy this problem:

Solution 1:

1. Run the opwdintg.exe to install the password filter on every Windows domain controller in the domain for Active Directory.

- Restart each Windows domain controller server. Each Windows domain controller must be restarted after you install the password filter. Otherwise, the password filter will not work on the Windows domain controller.
- 3. Assign the Active Directory users to the appropriate ORA VFR group.
- Reset the user password on Active Directory.
- 5. Run ldapsearch to check that the password has been generated.

Solution 2:

- 1. Grant the Oracle service directory user account the Read Properties and Write lockoutTime, which are permissions to access the properties of the Active Directory user who tries to log in to the database.
- 2. Set permissions for Control Access on the orclCommonAttribute of the Active Directory users.

Related Topics

 Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions

The Oracle service directory user account is for the interaction between Oracle Database and the LDAP directory service.

6.8.4 ORA-28300 Connection Errors

The ORA-28030: No permission to read user entry in LDAP directory service error is generated due to permissions problems with the Oracle service directory.

You can track this error using the CMU trace. For example:

```
2023-03-27 19:51:55.0 - KZLG_ERR: failed to modify user status Insufficient access 2023-03-27 17:57:27.0 - KZLG_ERR: LDAPERR=50, OER=28300
```

To remedy this problem, In addition), and also the permission

- 1. Grant the Oracle service directory user account the Read Properties and Write lockoutTime, which are permissions to access the properties of the Active Directory user who tries to log in to the database.
- 2. Set permissions for Control Access on the orclCommonAttribute of the Active Directory users.

Related Topics

 Step 1: Create an Oracle Service Directory User Account on Microsoft Active Directory and Grant Permissions

The Oracle service directory user account is for the interaction between Oracle Database and the LDAP directory service.

Using Trace Files to Diagnose CMU Connection Errors
 The trace setting gdsi tracks centrally managed users (CMU) connection errors.

6.8.5 Using Trace Files to Diagnose CMU Connection Errors

The trace setting gdsi tracks centrally managed users (CMU) connection errors.



As a user who has the ALTER SYSTEM privilege and the SYSDBA administrative privilege, you can enable this trace event as follows:

```
ALTER SYSTEM SET EVENTS='TRACE[GDSI] DISK LOW';
```

After the Active Directory user tries to log in, and if the login fails, go to the directory that contains the trace files and grep these files for the connection errors.

```
grep -i kzlg *.trc
```

Then you can collect and review the trace file that contains the detailed information.

To disable tracing, you can enter the following command:

```
ALTER SYSTEM SET EVENTS='TRACE[GDSI] OFF';
```



7

Authenticating and Authorizing IAM Users for Oracle DBaaS Databases

Identity and Access Management (IAM) users can be configured to connect to an Oracle Database as a service (Oracle DBaaS) instance.

- Introduction to Authenticating and Authorizing IAM Users for Oracle DBaaS
 Before you begin authenticating and authorizing IAM users for an Oracle DBaaS instance,
 you should understand the overall process.
- Configuring Oracle DBaaS for IAM
 To configure Oracle DBaaS to work with IAM, an Oracle DBaaS database administrator must first enable the IAM integration and then authorize IAM users and roles for Oracle DBaaS.
- Configuring IAM for Oracle DBaaS
 To configure IAM to work with the Oracle DBaaS instance, an IAM administrator may need to create an IAM policy and have users create an IAM database password.
- Accessing the Database Using an Instance Principal or a Resource Principal
 An Oracle Cloud Infrastructure (OCI) application or function can connect to the database
 instance using its own instance or resource principal.
- Configuring the Database Client Connection
 Configuring the IAM client connection controls the authentication of IAM users to the
 Oracle DBaaS instance.
- Accessing a Database Cross-Tenancy Using an IAM Integration
 Users and groups in one tenancy can access DBaaS database instances in another
 tenancy if policies in both tenancies allow this.
- Database Links in an Oracle DBaaS-to-IAM Integration
 The use of database links when accessing the Oracle DBaaS database using IAM credentials is supported.
- Troubleshooting IAM Connections
 The ORA-01017: invalid username/password; logon denied error can be caused by several different issues throughout the Oracle DBaaS integration with Identity and Access Management (IAM).

7.1 Introduction to Authenticating and Authorizing IAM Users for Oracle DBaaS

Before you begin authenticating and authorizing IAM users for an Oracle DBaaS instance, you should understand the overall process.

About Authenticating and Authorizing IAM Users for Oracle DBaaS
 Users for the Oracle DBaaS instance can be centrally managed in Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM).

Architecture of the IAM Integration with Oracle DBaaS

The architecture for the IAM integration with an Oracle DBaaS instance depends on whether the IAM user is using an Oracle Cloud Infrastructure (OCI) IAM database password verifier or an OCI IAM token to authenticate or connect to the DBaaS instance.

IAM Users and Groups to Map with Oracle DBaaS

IAM users must be mapped to a schema, either an exclusive mapping of a database schema to an IAM user or to a database shared schema that is mapped to an IAM group the user is a member of.

7.1.1 About Authenticating and Authorizing IAM Users for Oracle DBaaS

Users for the Oracle DBaaS instance can be centrally managed in Oracle Cloud Infrastructure (OCI) Identity and Access Management (IAM).

You can perform this integration in the following Oracle Database environments:

- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Autonomous Database on Exadata Cloud@Customer
- Oracle Exadata Database Service on Dedicated Infrastructure
- Oracle Exadata Database Service on Cloud@Customer
- Oracle Base Database Service

The instructions for configuring IAM use the term "Oracle DBaaS" to encompass these environments.

Note:

Oracle Database supports the Oracle DBaaS integration for OCI IAM with identity domains as well as the legacy IAM, which does not include identity domains. Both default and non-default domain users and groups are supported when using IAM with identity domains.

Oracle Database only supports Oracle DBaaS integration for OCI IAM with local IAM users when they use legacy IAM tenancies. Federated users are supported when using IAM with identity domains.

The DBaaS integration with OCI IAM does not support users with administrative privileges (SYSDBA, SYSOPER, SYSBACKUP, SYSDG, SYSKM, and SYSRAC).

An Oracle Database administrator works with an OCI IAM administrator to manage the authentication and authorization of OCI IAM users who need to connect to the Oracle DBaaS instance. The types of Oracle DBaaS instance that IAM users can connect to are Oracle Autonomous Database Serverless, Oracle Autonomous Database on Dedicated Exadata Infrastructure, and Oracle Base Database Service.

This type of connection enables the IAM user to access the Oracle DBaaS. These users typically log in with a user name and password (for example, using SQL*Plus). Alternatively, a user can log in with IAM Single-Sign On (SSO) credentials with a token when accessing the DBaaS instance. The choice to use IAM password authentication or the IAM SSO token authentication depends on the use case and user preference.



Legacy applications using existing supported database clients can migrate seamlessly to using an IAM user name and password. They can also use the IAM database gradual password rollover feature to set a second database password in IAM and update the application passwords without downtime.

Tools and applications that are updated to support IAM tokens can authenticate users directly with IAM and pass the database access token to the DBaaS instance. Existing database tools such as SQL*Plus can use the IAM database password to authenticate with the database directly using existing password login protocol or the database client can request a database token (db-token) from OCI IAM using the IAM user name and IAM database password and send the db-token to the database for IAM user access. The database client can only request a db-token in exchange for the IAM user name and IAM database password. All other IAM credentials (API-key, instance principal, resource principal, security token, delegation token) will require the db-token to be requested by the application or helper client like OCI CLI. A database access token (db-token) is a scoped proof-possession (POP) token and comes with a public key. Before the db-token is sent to the database, the database client signs the db-token with the private key that is associated with token's public key. It provides "proof" that the sender of the token is the rightful holder of the token. The scope can optionally be included as part of the request for the db-token to reduce the scope of what the db-token can be used for. The default scope for the db-token is the entire tenancy but compartment and individual databases can also be defined as the scope. See the get description in OCI CLI Command Reference for more information.

IAM users and OCI applications can request a database token from IAM by using one of the following methods:

- Using an existing, valid security (session) token
- Using an IAM recognized API-key
- Using a delegation token within an OCI cloud shell
- Using an OCI instance principal for an application on OCI compute instance
- Using an OCI resource principal for an application with a resource principal
- Using an IAM user name and IAM database password (can only be requested by database client)

The general process of enabling an IAM user to connect to an Oracle DBaaS instance is as follows:

- The IAM administrator creates and manages the IAM user accounts and groups, adding IAM users to appropriate IAM groups based on their tasks.
- 2. On the Oracle DBaaS instance, the database administrator enables the connection between the Oracle DBaaS and the IAM endpoint. If the database is Autonomous Database on Dedicated Exadata Infrastructure, then the IAM connection for new PDBs is automatically enabled. Check the Oracle DBaaS documentation for details.
- 3. On the Oracle DBaaS server, the database administrator enables the authorization of the IAM users by performing the following types of mappings:
 - Mapping an IAM group to a shared Oracle Database global user account
 - Mapping an IAM group to an Oracle Database global role
 - Exclusively mapping the IAM user to an Oracle Database global user

The IAM user must be mapped to one schema, either exclusively or to a shared schema. They can optionally be members in an IAM group that is mapped to one or more global roles.



- 4. The following use cases are some common scenarios to connect to the Oracle DBaaS with centralized IAM authentication and authorization:
 - Connecting using SQL*Plus to the Oracle DBaaS using an IAM user name and IAM database password.
 - Using SQL*Plus to connect using an IAM SSO token.
 - Using SQLcl to connect to the Oracle DBaaS using the IAM password or IAM token.
 - Using SQL*Plus within the Oracle Cloud Infrastructure (OCI) Cloud Shell to connect to the Oracle DBaaS using the IAM password or IAM SSO token. Authenticating and authorization with IAM will take additional time as opposed to authenticating to a local database user account (non-global).

Related Topics

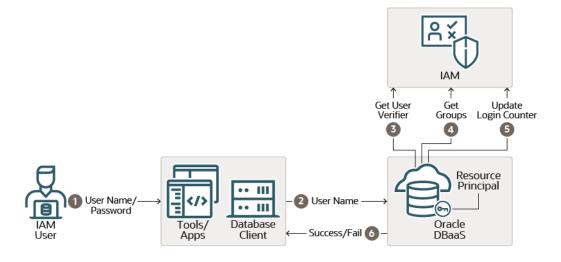
- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Autonomous Database on Exadata Cloud@Customer
- Oracle Exadata Database Service on Dedicated Infrastructure
- Oracle Exadata Database Service on Cloud@Customer
- Oracle Base Database Service
- Enabling External Authentication for Oracle DBaaS
 The method of enabling an IAM connection with Oracle DBaaS depends on the platform of Oracle DBaaS that you are using.

7.1.2 Architecture of the IAM Integration with Oracle DBaaS

The architecture for the IAM integration with an Oracle DBaaS instance depends on whether the IAM user is using an Oracle Cloud Infrastructure (OCI) IAM database password verifier or an OCI IAM token to authenticate or connect to the DBaaS instance.

The following diagram illustrates how using an Oracle Cloud Infrastructure (OCI) IAM database password verifier to authenticate with the Oracle DBaaS works:

Figure 7-1 IAM User Authenticating to Oracle DBaaS with an OCI IAM Database Password Verifier



- 1. The IAM user logs in to a tool or application client that is associated with the Oracle Database client. This user logs in with their IAM user name and IAM database password, which begins the authentication process. The user can use any database client that is at least Oracle Database release12.1.0.2. Earlier versions of the database client do not support the 12c database verifier.
- The IAM user connection request is sent through the database client.
- 3. After the IAM user name is sent to the Oracle DBaaS instance, the database requests the user's Oracle Cloud Infrastructure (OCI) IAM database password verifier from IAM. (The IAM user profile stores the IAM database password verifier.) This verifier is a hashed version of the password, not clear text. If the password verifier from IAM matches the password verifier generated by the database client, then the user is authenticated. The Oracle DBaaS instance uses a resource principal to communicate with IAM. The resource principal is the Oracle DBaaS identity that is recognized by IAM and used by the database to securely communicate with IAM.
- 4. When the authentication succeeds, the Oracle DBaaS instance retrieves the IAM user groups. If the IAM user is mapped to an Oracle Database schema and the user has not been locked out of their OCI account, then the IAM user successfully accesses the database. The user is also granted any global roles that are mapped to a group the user is a member of.
- 5. The Oracle Cloud Infrastructure (OCI) login counter tracks logins for both the OCI console and OCI database passwords. A successful database login using the IAM database password will reset this counter.
- 6. Based on the outcome of the preceding steps, the IAM user database access attempt either succeeds or fails.

The following diagram illustrates the start of actions that take place when an IAM user or an Oracle Cloud Infrastructure (OCI) application accesses the Oracle DBaaS instance using an OCI IAM token:



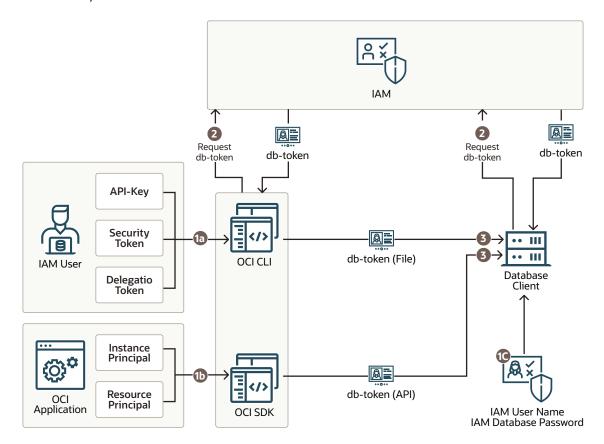


Figure 7-2 IAM User or OCI Application Authenticating to an Oracle DBaaS with an OCI IAM Token, Part 1

- Access to the database requires one of the following:
 - 1a: From an IAM user, the user must have an API-key stored in their local system or have a security token from signing into OCI recently. An API-key, security token, delegation token, instance principal, can be used with the OCI CLI. If a current and valid security token is not available, then the user can be prompted to authenticate with OCI IAM. (See User Credentials for information about the available user credentials.) In an OCI cloud shell environment, a delegation token will be available.
 - **1b:** For an OCI application, the application must have be configured to have an instance principal or a resource principal. All key types (API-key, security token, delegation token, instance principal, and resource principal) can be used with the OCI SDK.
 - 1c: You can configure the database client to request a db-token from IAM by using the IAM user name and IAM database password. Only the database client can use this type of token to access the database. The database client cannot request a db-token using any other credential.
- 2. The application, OCI CLI, or the database client makes a call to IAM requesting the db-token using one of the principal credentials. Only the db-token can be used to access the Oracle DBaaS. Requesting a db-token can be done by an application written with the Oracle Cloud Infrastructure (OCI) public SDK to connect with OCI IAM. (See Software Development Kits and Command Line Interface.) If an application cannot be changed to connect directly with OCI IAM using the OCI public SDK, then a helper tool such as the OCI command line interface (OCI CLI) can be used to retrieve the db-token for the user.



The database client can also be configured to request a db-token using the IAM user name and IAM database password.

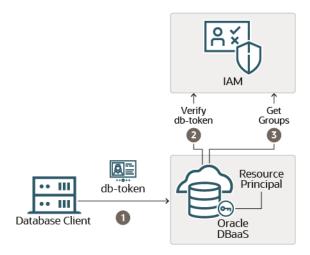
3. An application or tool that has been updated to work with IAM can then pass the db-token directly to the database client through the client API as an attribute. If an application cannot be updated to get the db-token directly, then a helper tool such as OCI CLI can put the db-token into the default or specified location in the local directory. The TOKEN_AUTH=OCI_TOKEN setting in the connect string or the sqlnet.ora file enables the database client to retrieve the db-token from the default or specified file location. A user can request a token at the OCI CLI by running the oci iam db-token get command and specifying their profile, which stores their user account credentials. For example:

oci iam db-token get --profile PeterFitch

The directory location for the <code>db-token</code> and the corresponding private key should only have enough permission for the OCI CLI to write the files to the location and the database client to retrieve these files (for example, just read and write by the process user). Because the token and key allow access to the database, they should be protected within the file system.

The following diagram illustrates the continuation of the OCI IAM token authentication process:

Figure 7-3 IAM User or OCI Application Authenticating to an Oracle DBaaS with an OCI IAM Token, Part 2



- 1. The db-token is signed and sent to the Oracle DBaaS instance. TLS must be enabled on the database client-server link as well as DN matching. (When you use the Autonomous Database wallet files to connect to the Autonomous Database instance, TLS and DNS matching is already set for you.) DN matching is on by default with the JDBC driver, but will need to be configured for the OCI-C database client (and instant client). A db-token that the database client retrieves by using an IAM user name and IAM database password does not come with a private key and is not be signed by the database client.
- 2. The Oracle DBaaS instance will request the IAM public key, if a valid copy is not already available locally. This key will be used to validate that the db-token was sent by IAM. The Oracle DBaaS instance uses a resource principal to communicate with IAM.
- 3. After this authorization step completes successfully, the Oracle DBaaS instance will request the IAM user's groups from IAM. This action will map the user to a global schema and also to map the user to any global roles that the user is a member of. After the IAM

user has successfully completed these steps, the user has access to the Oracle DBaaS instance.

IAM SSO token-based authentication requires that you download the latest Oracle Database 19c (19.16) clients.

Related Topics

Using Oracle Autonomous Database Serverless

7.1.3 IAM Users and Groups to Map with Oracle DBaaS

IAM users must be mapped to a schema, either an exclusive mapping of a database schema to an IAM user or to a database shared schema that is mapped to an IAM group the user is a member of.

An IAM user must be mapped to a database schema to successfully complete the login and authorization steps. An IAM user can be directly mapped to a database schema if the IAM user needs to maintain their own schema objects (exclusive mapping). More commonly, an IAM user is a member of an IAM group that is mapped to a database schema (shared schema mapping). Shared schema mapping allows multiple IAM users to share the same schema so a new database schema is not required to be created every time a new user joins the organization. This operational efficiency allows database administrators to focus on database application maintenance, performance, and tuning tasks instead of configuring new users, updating privileges and roles, and removing accounts.

Database administrators for a group of databases can be members of an IAM group (for example, sales application developers for a sales application are in an IAM group called sales_app_dev_group). In this scenario, all the related databases can map the shared schema to the sales_app_dev_group group. Database global roles cannot be granted to a schema; they can only be mapped to an IAM group. Global roles can differentiate IAM user privileges when multiple IAM users are mapped to the same shared schema.

Remember that an IAM user **must** be mapped exclusively to a database schema or to a shared schema so that the IAM user can access the Oracle DBaaS instance.

7.2 Configuring Oracle DBaaS for IAM

To configure Oracle DBaaS to work with IAM, an Oracle DBaaS database administrator must first enable the IAM integration and then authorize IAM users and roles for Oracle DBaaS.

- Enabling External Authentication for Oracle DBaaS
 The method of enabling an IAM connection with Oracle DBaaS depends on the platform of Oracle DBaaS that you are using.
- Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications
 An Oracle DBaaS database administrator can map IAM users and Oracle Cloud
 Infrastructure (OCI) applications to the Oracle Database global schemas and global roles.
- Configuring IAM Proxy Authentication
 Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.

7.2.1 Enabling External Authentication for Oracle DBaaS

The method of enabling an IAM connection with Oracle DBaaS depends on the platform of Oracle DBaaS that you are using.

- Oracle Autonomous Database on Dedicated Exadata Infrastructure: The IAM
 connection is automatically configured to work with this platform. See Using Oracle
 Autonomous Database on Dedicated Exadata Infrastructure.
- Oracle Autonomous Database Serverless: The IAM connection must be enabled to work with this platform. See Using Oracle Autonomous Database Serverless.
- Oracle Base Database Service: See Use Identity and Access Management Authentication with Base Database Service.
- Oracle Exadata Database Service on Dedicated Infrastructure: See Connect Identity and Access Management (IAM) Users to Oracle Exadata Database Service on Dedicated Infrastructure.

Databases Other Than Oracle Autonomous Database Serverless

- Refer to the documentation for your Oracle DBaaS platform for prerequisites and other information you may need.
- 2. For non-Oracle Autonomous Database instances, set the <code>IDENTITY_PROVIDER_CONFIG</code> parameter.

```
ALTER SYSTEM SET IDENTITY PROVIDER TYPE=OCI IAM SCOPE=BOTH;
```

If IDENTITY_PROVIDER_CONFIG had been set to a different value, then run the following statement:

```
ALTER SYSTEM RESET IDENTITY PROVIDER CONFIG SCOPE=BOTH;
```

The IDENTITY_PROVIDER_CONFIG parameter may have been set to a different value because a different identity provider, such as Microsoft Azure, had been used.

7.2.2 Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications

An Oracle DBaaS database administrator can map IAM users and Oracle Cloud Infrastructure (OCI) applications to the Oracle Database global schemas and global roles.

- About Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications
 - You create the mappings for IAM users and Oracle Cloud Infrastructure (OCI) applications to database users (schemas) in the Oracle DBaaS.
- Mapping an IAM Group to a Shared Oracle Database Global User
 Oracle Database global users that are mapped to IAM groups and IAM dynamic groups
 give IAM users and OCI applications a schema when they log in along with the privileges
 and roles granted to that schema.
- Mapping an IAM Group to an Oracle Database Global Role
 Oracle Database global roles that are mapped to IAM groups and dynamic groups give
 member users and applications additional privileges and roles above what they have been
 granted through their login schemas.
- Exclusively Mapping an IAM User to an Oracle Database Global User You can map an IAM user exclusively to an Oracle Database global user.



- Altering or Migrating an IAM User Mapping Definition
 - You can update an IAM user to a database global user mapping by using the ALTER USER statement.
- Mapping Instance and Resource Principals
 - Applications can use instance principals and resource principals to retrieve database tokens and establish a connection to an Oracle DBaaS instance.
- Verifying the IAM User Logon Information
 - After you configure and authorize an IAM user for the Oracle DBaaS instance, you can verify the user logon information by executing a set of SQL queries on the Oracle database side

7.2.2.1 About Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications

You create the mappings for IAM users and Oracle Cloud Infrastructure (OCI) applications to database users (schemas) in the Oracle DBaaS.

There is a difference with authorization between IAM database password authentication and using IAM token based authentication. IAM database password verifier authorization is only based on mappings of database schemas and global roles to IAM users and group. With IAM token based authentication, IAM policies are an additional authorization for IAM users to access their tenancy databases. An IAM user must be authorized through an IAM policy and be authorized through a mapping to a database global schema (exclusive or shared).

For both token and password verifier database access, you create the mappings for IAM users and OCI applications to the Oracle DBaaS instance. The IAM user accounts themselves are managed in IAM. The user accounts and user groups can be in either the default domain or in a custom, non-default domain.

When the IAM user accesses the Oracle DBaaS instance with a token, the database will perform an authorization check against IAM policies to ensure the user is allowed to access the database. If the IAM user is allowed to access the database by IAM policy, then the database will query IAM for the user groups. When using password verifier authentication, the database will guery IAM for user groups once the IAM user successfully completes authentication. The database queries the IAM endpoint to find the groups of which the user is a member. If your deployment is using shared schemas, then one of the IAM groups will map to a shared database schema and the IAM user will be assigned to that database schema. The IAM user will have the roles and privileges that are granted to the database schema. Because multiple IAM users can be assigned to the same shared database schema, only the minimal set of roles and privileges should be granted to the shared schema. In some cases, no privileges and roles should be granted to the shared schema. Users will be assigned the appropriate set of roles and schemas through database global roles. Global roles are mapped to IAM groups. This way, different users can have different roles and privileges even if they are mapped to the same database shared schema. A newly hired user will be assigned to an IAM group mapped to a shared schema and then to one or more additional groups mapped to global roles to gain the additional roles and privileges required to complete their tasks. The combination of shared schemas and global roles allows for centralized authorization management with minimal changes to the database operationally. The database must be initially provisioned with the set of shared schemas and global roles mapped to the appropriate IAM groups, but then user authorization management can happen within IAM.

Ensure that the IAM user is only mapped to one schema, either through exclusive mapping to a database schema or as a member of one IAM group that is mapped to a shared database schema. If more than one schema is mapped for an IAM user, then the database will take

exclusive mapping as precedence over any group mapping to a shared schema. If more than one group is mapped for a user, then the database will select the oldest mapping.

When using global roles to grant privileges and roles to the user, remember that the maximum number of enabled roles in a session is 150.

If you drop and recreate IAM users and groups using the same names, then the mappings from the database to IAM using the same names will continue to work. However, recreating an IAM user will require the IAM user to do one or more of the following: create the IAM database password, re-upload the API public key, update the OCI configuration file, and then re-examine the IAM policy for database authentication and authorization with IAM. If the IAM policy specifies a group that can use or manage the database-connections and autonomous-database-family resource types, then the user will need to be added to that group to allow IAM authentication and authorization.

Accessing the database with tokens requires the user to be authorized by IAM policy and by database mapping. Accessing the database with the IAM database password verifier requires authorization through database mapping. If no database schema mapping exists for the IAM user, the IAM user is prevented from accessing the database even if they have a valid token or password.

IAM users get their authorizations to perform various tasks based on the roles that they have been granted. The following scenarios are possible:

- IAM group mapped to a shared Oracle Database global user: With the shared database global user account, an IAM user is assigned to a shared database schema (user) through the mapping of an IAM group to the shared schema. The IAM users that are members of the group can connect to the database through this shared schema. Use of shared schemas allows for centralized management of user authorization in IAM.
- IAM group mapped to an Oracle Database global role: The privileges that have been
 granted to the shared Oracle Database global role become available to the users who
 have added to the IAM group.
- Local IAM user exclusively mapped to an Oracle Database global user: With an exclusive global user mapping, a dedicated database user is exclusively mapped to a local IAM user. Not as common as the shared database schema, this user is created for when the user requires their own schema objects. Oracle recommends that you grant database privileges to these users through global roles, which facilitates authorization management. These users can also have direct privilege and role grants to their exclusive schema. In IAM with Identity Domains, users and groups are supported in the default domain as well as custom non-default domains. When you specify users and groups in the default domain, then no domain prefix is required. When you specify users and groups in a non-default domain, then the domain must be prefixed.

7.2.2.2 Mapping an IAM Group to a Shared Oracle Database Global User

Oracle Database global users that are mapped to IAM groups and IAM dynamic groups give IAM users and OCI applications a schema when they log in along with the privileges and roles granted to that schema.

- Log in to the Oracle DBaaS instance as a user who has the CREATE USER or ALTER USER system privilege.
- 2. Run the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the IAM group name (which can be a dynamic group).



For example, to create a new database global user account named <code>shared_sales_schema</code> and map it to an existing IAM group named <code>WidgetSalesGroup</code>:

```
CREATE USER shared_sales_schema IDENTIFIED GLOBALLY AS
'IAM GROUP NAME=WidgetSalesGroup';
```

The following example shows how to accomplish this for a non-default domain:

```
CREATE USER shared_sales_schema IDENTIFIED GLOBALLY AS
'IAM GROUP NAME=sales domain/WidgetSalesGroup';
```

7.2.2.3 Mapping an IAM Group to an Oracle Database Global Role

Oracle Database global roles that are mapped to IAM groups and dynamic groups give member users and applications additional privileges and roles above what they have been granted through their login schemas.

Global roles cannot be granted to a database schema (user), they can only be mapped to a group and be assigned to an IAM user when accessing the database.

- 1. Log in to the Oracle DBaaS instance as a user who has been granted the CREATE ROLE or ALTER ROLE system privilege
- 2. Run the CREATE ROLE or ALTER ROLE statement with the IDENTIFIED GLOBALLY AS clause specifying the name of the IAM group (which can be a dynamic group).

For example, to create a new database global role named widget_mgr_role and map it to an existing IAM group named WidgetManagerGroup, using the default domain:

```
CREATE ROLE widget_mgr_role IDENTIFIED GLOBALLY AS
'IAM GROUP NAME=WidgetManagerGroup';
```

The following example shows how to create the role by specifying a non-default domain, sales domain:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS
'IAM_GROUP_NAME=sales_domain/WidgetManagerGroup';
```

All members of the WidgetManagerGroup in the sales_domain domain will be authorized with the database global role widget_sales_role when they log in to the database.

7.2.2.4 Exclusively Mapping an IAM User to an Oracle Database Global User

You can map an IAM user exclusively to an Oracle Database global user.

- 1. Log in to the Oracle DBaaS instance as a user who has been granted the CREATE USER or ALTER USER system privilege.
- 2. Run the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the IAM database user name.

By default, the IAM database user name is the same as the IAM user name, including the domain name. You can also create a unique IAM database user name for ease of authentication to the database. In your OCI IAM user profile, you can create a unique IAM database user name for ease of authentication to the database. This can be set when you

create and manage your IAM database password in your IAM profile. Adding or changing the IAM database user name will invalidate the IAM user to schema mapping, so the database schema will need to be remapped to the new IAM database user name.

For example, to create a new database global user named <code>peter_fitch</code> and map this user to an existing IAM user named with an IAM database user name of <code>peterfitch</code>, using the default domain:

```
CREATE USER peter_fitch IDENTIFIED GLOBALLY AS 'IAM PRINCIPAL NAME=peterfitch';
```

The following example shows how to create the user by specifying a non-default domain, sales domain:

```
CREATE USER peter_fitch2 IDENTIFIED GLOBALLY AS
'IAM PRINCIPAL NAME=sales domain/peterfitch';
```

7.2.2.5 Altering or Migrating an IAM User Mapping Definition

You can update an IAM user to a database global user mapping by using the ALTER USER statement.

You can update database schemas that were mapped to an IAM user, and whose accounts were created using any of the CREATE USER statement clauses: IDENTIFIED BY password, IDENTIFIED EXTERNALLY, or IDENTIFIED GLOBALLY. This is useful when migrating existing schemas to using IAM. If you delete and recreate an IAM user or an IAM group using the exact same name as the previous IAM user or group, then the existing mapping from the database that uses that IAM user or IAM group name will continue to work.

- Log in to the Oracle DBaas instance as a user who has been granted the ALTER USER system privilege.
- 2. Run the Alter user statement with the identified globally as clause.

For example, suppose you want to change the existing schema shared_sales_schema to a different IAM group:

```
ALTER USER shared_sales_schema IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=BiggerWidgetSalesGroup';
```

The following example shows how to modify the schema by specifying a non-default domain, sales domain:

```
ALTER USER shared_sales_schema IDENTIFIED GLOBALLY AS 'IAM_GROUP_NAME=sales_domain/BiggerWidgetSalesGroup';
```

7.2.2.6 Mapping Instance and Resource Principals

Applications can use instance principals and resource principals to retrieve database tokens and establish a connection to an Oracle DBaaS instance.

You can exclusively map instance principals and resource principals to a global schema (database user) or you can map them by using dynamic groups to a shared schema.

You can only use instance principal and resource principal OCIDs to map them exclusively or to a shared schema. Instance principal and resource principal dynamic groups can also be mapped to global roles.

Examples are as follows:

Exclusive schema mapping using an instance principal (ip_user) and a resource principal (rp user):

```
CREATE USER ip_user IDENTIFIED GLOBALLY AS
'IAM_PRINCIPAL_OCID=ocid1.instance.region1.sea.abcdef123456';

CREATE USER rp_user IDENTIFIED GLOBALLY AS
'IAM PRINCIPAL OCID=ocid1.dbsystem.oc1.sea.abcdef123456';
```

Shared schema mapping using dynamic group:

```
CREATE USER iam dg IDENTIFIED GLOBALLY AS 'IAM GROUP NAME=DB Principals';
```

· Mapping to a global role;

```
CREATE ROLE app_role IDENTIFIED GLOBALLY AS
'IAM GROUP NAME=application principals';
```

Related Topics

- Managing Dynamic Groups
- Calling Services from an Instance
- Accessing Other Oracle Cloud Infrastructure Resources from Running Functions

7.2.2.7 Verifying the IAM User Logon Information

After you configure and authorize an IAM user for the Oracle DBaaS instance, you can verify the user logon information by executing a set of SQL queries on the Oracle database side.

 Log in to the Oracle DBaaS instance as an IAM user that you have just configured and authorized.

For example, to log in to the database instance <code>inst1</code> as the database global user <code>peterfitch</code>, who is using the default domain in IAM:

```
sqlplus /nolog
CONNECT "peterfitch"@inst1
Enter password: password
```

This example shows how to log in if user peterfitch is in a non-default domain, sales domain:

```
sqlplus /nolog
CONNECT "sales_domain/peterfitch"@inst1
Enter password: password
```

2. Verify the mapped global user.

The mapped global user is the database user account that has the IAM user authorization. User PETER_FITCH_SCHEMA is considered a global user with exclusive mapping for the IAM user peterfitch, while user WIDGET_SALES is considered a global user with shared mapping for IAM group widget sales group of which peterfitch is a member.

```
SHOW USER;
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

```
USER is "PETER_FITCH_SCHEMA"

Or

USER is "WIDGET SALES"
```

3. Find the roles that have been granted to the centrally managed user.

```
SELECT ROLE FROM SESSION ROLES ORDER BY ROLE;
```

Output similar to the following appears:

- 4. Run the following queries to check the SYS_CONTEXT namespace values for the current schema being used in this database session, current user name, session user name, authentication method, authenticated identity, enterprise identity, identification type, and server type.
 - Verify the current schema that is being used in this database session. A database schema is an object container that identifies the objects it contains. The current schema is the default container for objects name resolution in this database session.

```
SELECT SYS CONTEXT ('USERENV', 'CURRENT SCHEMA') FROM DUAL;
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

 Verify the current user. In this case, the current user is the same as the current schema.

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

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

WIDGET SALES

Verify the session user.

```
SELECT SYS CONTEXT('USERENV', 'SESSION USER') FROM DUAL;
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

Or

Verify the authentication method.

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

Output similar to the following appears:

If the user is authenticating with a token, then the output is TOKEN GLOBAL.

 Verify the authenticated identity for the enterprise user. The IAM authenticated user identity is captured and audited when this user logs on to the database.

```
SELECT SYS CONTEXT ('USERENV', 'AUTHENTICATED IDENTITY') FROM DUAL;
```

Output similar to the following appears:

```
SYS_CONTEXT('USERENV','AUTHENTICATED_IDENTITY')
-----sales domain/peterfitch
```

If a user nickname has been set for the enterprise user, then verify this nickname.

```
SELECT SYS CONTEXT('USERENV', 'USER NICKNAME') FROM DUAL;
```

Output similar to the following appears:

Verify the centrally managed user's enterprise identity.

```
SELECT SYS CONTEXT('USERENV', 'ENTERPRISE IDENTITY') FROM DUAL;
```

Enterprise Identity will show the OCI Identity (OCID) of the IAM user or OCI application. Output similar to the following appears:

Verify the identification type.

```
SELECT SYS CONTEXT('USERENV', 'IDENTIFICATION TYPE') FROM DUAL
```

Output similar to the following appears, depending on if it is an exclusive mapping or a shared mapping:

```
SYS_CONTEXT('USERENV','IDENTIFICATION_TYPE')

GLOBAL EXCLUSIVE

Or

SYS_CONTEXT('USERENV','IDENTIFICATION_TYPE')

GLOBAL SHARED
```

Verify the server type.

```
SELECT SYS CONTEXT('USERENV', 'LDAP SERVER TYPE') FROM DUAL;
```

Output similar to the following appears. In this case, the LDAP server type is IAM.

7.2.3 Configuring IAM Proxy Authentication

Proxy authentication allows an IAM user to proxy to a database schema for tasks such as application maintenance.

- About Configuring IAM Proxy Authentication
 IAM users can connect to Oracle DBaaS by using proxy authentication.
- Configuring Proxy Authentication for the IAM User
 To configure proxy authentication for an IAM user, the IAM user must already have a
 mapping to a global schema (exclusive or shared mapping). A separate database schema
 for the IAM user to proxy to must also be available.
- Validating the IAM User Proxy Authentication
 You can validate the IAM user proxy configuration for both password and token
 authentication methods.

7.2.3.1 About Configuring IAM Proxy Authentication

IAM users can connect to Oracle DBaaS by using proxy authentication.

Proxy authentication is typically used to authenticate the real user and then authorize them to use a database schema with the schema privileges and roles in order to manage an application. Alternatives such as sharing the application schema password are considered insecure and unable to audit which actual user performed an action.

A use case can be in an environment in which a named IAM user who is an application database administrator can authenticate by using their credentials and then proxy to a database schema user (for example, hrapp). This authentication enables the IAM administrator to use the hrapp privileges and roles as user hrapp in order to perform application maintenance, yet still use their IAM credentials for authentication. An application database administrator can sign in to the database and then proxy to an application schema to manage this schema.

You can configure proxy authentication for both the password authentication and token authentication methods.

7.2.3.2 Configuring Proxy Authentication for the IAM User

To configure proxy authentication for an IAM user, the IAM user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the IAM user to proxy to must also be available.

After you ensure that you have this type of user, alter the database user to allow the IAM user to proxy to it.

- 1. Log in to the Autonomous Database instance as a user who has the ALTER USER system privileges.
- 2. Grant permission for the IAM user to proxy to the local database user account.

An IAM user cannot be referenced in the command so the proxy must be created between the database global user (mapped to the IAM user) and the target database user.

In the following example, hrapp is the database schema to proxy to, and peterfitch schema is the database global user exclusively mapped to user peterfitch.

```
ALTER USER hrapp GRANT CONNECT THROUGH peterfitch schema;
```

At this stage, the IAM user can log in to the database instance using the proxy. For example, to connect using a password verifier:

```
CONNECT peterfitch[hrapp]@connect_string
Enter password: password
```

To connect using a token:

```
CONNECT [hrapp]/@connect_string
```

7.2.3.3 Validating the IAM User Proxy Authentication

You can validate the IAM user proxy configuration for both password and token authentication methods.

- 1. Log in to the Autonomous Database instance as a user who has the CREATE USER and ALTER USER system privileges.
- 2. Connect as the IAM user and run the SHOW USER and SELECT SYS CONTEXT commands.

For example, suppose you want to check the proxy authentication of the IAM user peterfitch when they proxy to database user hrapp. Run the following queries after you proxy to the database using an IAM user. Depending on how you authenticate and access the database, you will get different values for these queries.

For password authentication, assuming the IAM user is in the default domain:

```
CONNECT peterfitch[hrapp]/password\!@connect_string
SHOW USER;
--The output should be "USER is HRAPP"
SELECT SYS_CONTEXT('USERENV','AUTHENTICATION_METHOD') FROM DUAL;
--The output should be "PASSWORD_GLOBAL_PROXY"
SELECT SYS CONTEXT('USERENV','PROXY USER') FROM DUAL;
```



```
--The output should be "PETERFITCH_SCHEMA"

SELECT SYS_CONTEXT('USERENV','CURRENT_USER') FROM DUAL;

--The output should be "HRAPP"
```

• For token authentication, for a user who is in a non-default domain, sales_domain:

```
CONNECT [hrapp]/@connect_string
SHOW USER;
--The output should be USER is "HRAPP"

SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATION_METHOD') FROM DUAL;
--The output should be "TOKEN_GLOBAL_PROXY"

SELECT SYS_CONTEXT('USERENV', 'PROXY_USER') FROM DUAL;
--The output should be "PETERFITCH_SCHEMA"

SELECT SYS_CONTEXT('USERENV', 'CURRENT_USER') FROM DUAL;
--The output should be "HRAPP"
```

7.3 Configuring IAM for Oracle DBaaS

To configure IAM to work with the Oracle DBaaS instance, an IAM administrator may need to create an IAM policy and have users create an IAM database password.

- Creating an IAM Policy to Authorize Users Authenticating with Tokens
 To configure IAM to work with the Oracle DBaaS instance, an IAM administrator must
 create an IAM policy (if using IAM tokens), create IAM groups and manage group
 membership.
- Creating an IAM Database Password
 The IAM database password, different from the Oracle Cloud Infrastructure (OCI) console password, and set by the IAM user, is required for the Oracle DBaaS password verification process.

7.3.1 Creating an IAM Policy to Authorize Users Authenticating with Tokens

To configure IAM to work with the Oracle DBaaS instance, an IAM administrator must create an IAM policy (if using IAM tokens), create IAM groups and manage group membership.

The IAM administrator should work with the database administrator to create the appropriate IAM groups for databases. Individual IAM users will need to create an IAM database password in their profile if they are using password verifiers.

You do not need to create a policy for users who are authenticating with password verifiers.

Use the allow group command to create the policy. For example:

```
allow group DBUsers to use database-connections in tenancy
```

 To create a policy that limits members of DBUsers group to access DBaaS instances in compartment testing compartment only

```
allow group DBUsers to use autonomous-database-family in compartment testing compartment
```

To create a policy that limits group access to a single database in a compartment:

```
allow group DBUsers to use autonomous-database-family in compartment
testing_compartment where target.database.id =
'ocid1.autonomousdatabase.oc1.iad.aaaabbbbcccc'
```

Note the following:

- The database-connections resource type is included in the autonomous-database-family resource type. Either resource can be used, depending on your use case.
- The minimum verb to enable access to the database is use. You can also use the manage verb to enable access to the database.
- Dynamic group names are case sensitive when they are used in this policy. You must use
 the exact case for the dynamic group name when using it with this policy.

See Oracle Cloud Infrastructure Documentation for more information about the syntax of policy statements.

7.3.2 Creating an IAM Database Password

The IAM database password, different from the Oracle Cloud Infrastructure (OCI) console password, and set by the IAM user, is required for the Oracle DBaaS password verification process.

The set of allowed characters for the OCI IAM database password is similar to the set of allowed characters for the OCI console password except that the double quotation mark character is not allowed for the OCI IAM database password. See Managing User Credentials for information about creating an IAM database password.

- Log in to the OCI console to your user page.
- 2. Access **My profile** or **User settings** (top right in the navigation toolbar) depending on the IAM version that you are using.
- 3. In your profile or settings, in the left, under Resources, click on the **Database Passwords** link.
- 4. Click the Create Database Password button.
- 5. Add a description and the password, ensuring that you apply the listed complexity rules.
- 6. Click Create Database Password to save the password.

After the password is created, its description and creation date are listed under Database Passwords.

7.4 Accessing the Database Using an Instance Principal or a Resource Principal

An Oracle Cloud Infrastructure (OCI) application or function can connect to the database instance using its own instance or resource principal.

You can map instance principals and resource principals exclusively to a database global schema or to a shared schema using a mapping to a dynamic group. When mapping instance



principals and resource principals exclusively to a database global schema, you must use the principal OCID. For example:

```
CREATE USER widget IDENTIFIED GLOBALLY
AS 'IAM_PRINCIPAL_OCID=ocid1.instance.region1.sea.1234567890abcdef';
```

When using shared schemas, you must add instance principals and resource principals to a dynamic group, and map the dynamic group to the shared schema.

Related Topics

- Managing Dynamic Groups
- Calling Services from an Instance
- Accessing Other Oracle Cloud Infrastructure Resources from Running Functions
- Accessing the Oracle Cloud Infrastructure API Using Instance Principals
- Using Oracle Autonomous Database Serverless

7.5 Configuring the Database Client Connection

Configuring the IAM client connection controls the authentication of IAM users to the Oracle DBaaS instance.

- About Connecting to an Autonomous Database Instance Using IAM
 IAM users can connect to the Autonomous Database instance by using either an IAM database password verifier or an IAM token.
- Supported Client Drivers for IAM Connections
 Oracle DBaaS supports several types of client drivers for IAM connections.
- Using Centralized Oracle Cloud Infrastructure Services for Net Naming and Secrets
 You can use the Oracle Cloud Infrastructure (OCI) object store and vault to centrally store
 net names and secrets.
- Client Connections That Use an IAM Database Password Verifier
 After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.
- Client Connections That Use a Token Requested by an IAM User Name and Database Password

You can create a client connection that uses a token requested by an IAM user name and database password.

- Client Connections That Use a Token Requested by a Client Application or Tool
 For IAM token access to the Oracle DBaaS, the client application or tool requests a
 database token from IAM for the IAM user.
- TLS Connections without Client Wallets
 The use of Transport Layer Security (TLS) connections without client wallets is supported for IAM connections.
- Common Database Client Configurations
 IAM users can connect to the Oracle DBaaS instance using client tools such as SQLcl on a laptop.

7.5.1 About Connecting to an Autonomous Database Instance Using IAM

IAM users can connect to the Autonomous Database instance by using either an IAM database password verifier or an IAM token.

Using the IAM database password verifier is similar to the Oracle Database password authentication process. However, instead of the password verifier (encrypted hash of the password) being stored in the Oracle database, the verifier is instead stored as part of the Oracle Cloud Infrastructure (OCI) IAM user profile.

The second connection method, the use of an IAM token for the database, is more modern. The use of token-based access is a better fit for Cloud resources such as Autonomous Database. The token is based on the strength that the IAM endpoint can enforce. This can be multi-factor authentication, which is stronger than the use of passwords alone. Another benefit of using tokens is that the password verifier (which is considered sensitive) is never stored or available in memory. A TCPS (TLS) connection is required when using tokens for database access.



You cannot configure native network encryption when passing an IAM token. Only Transport Layer Security (TLS) by itself is supported, not native network encryption or native network encryption with TLS.

7.5.2 Supported Client Drivers for IAM Connections

Oracle DBaaS supports several types of client drivers for IAM connections.

IAM database password verifiers work with any supported database client. Using IAM tokens requires the latest Oracle Database client 19c (at least 19.16). Some earlier clients (19c and 21c) provide a limited set of capabilities for token access. Oracle Database client 21c does not fully support the IAM token access feature. Oracle Database client 23c supports the IAM token access feature.

7.5.3 Using Centralized Oracle Cloud Infrastructure Services for Net Naming and Secrets

You can use the Oracle Cloud Infrastructure (OCI) object store and vault to centrally store net names and secrets.

This functionality is currently supported with the JDBC-thin and .NET-thin drivers.

See the following guides:

- Oracle Database Net Services Administrator's Guide
- Oracle Database Net Services Reference



7.5.4 Client Connections That Use an IAM Database Password Verifier

After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.

The IAM user enters the IAM user name and IAM database password (not the Oracle Cloud Infrastructure (OCI) console password) using any currently supported database client. The only constraint is that the database client version be either Oracle Database release 12.1.0.2 or later to use Oracle Database 12c passwords. The database client must be able to use the 12c password verifier. Using the 11g verifier encryption is not supported with IAM. No special client or tool configuration is needed for the IAM user to connect to the OCI DBaaS instance.

7.5.5 Client Connections That Use a Token Requested by an IAM User Name and Database Password

You can create a client connection that uses a token requested by an IAM user name and database password.

- About Client Connections That Use a Token Requested by an IAM User Name and Database Password
 - IAM users can connect to the Oracle DBaaS instance by using an IAM token that was retrieved using an IAM user name and IAM database password.
- Parameters to Set for Client Connections That Use a Token Requested by an IAM User Name and Database Password
 - To set these parameters, you modify either the sqlnet.ora file or the tnsnames.ora file.
- Configuring the Database Client to Retrieve a Token Using an IAM User Name and Database Password
 - You can configure the database client to retrieve the IAM database token using the provided IAM user name and IAM database password.
- Configuring a Secure External Password Store Wallet to Retrieve an IAM Token
 You can enable an IAM user name and a secure external password store (SEPS) to
 request the IAM database token.

7.5.5.1 About Client Connections That Use a Token Requested by an IAM User Name and Database Password

IAM users can connect to the Oracle DBaaS instance by using an IAM token that was retrieved using an IAM user name and IAM database password.

In both cases, the token is retrieved by using a database password, either by using SQL*Plus or through a SEPS.

In previous releases, you could only use the IAM user name and database password to get a password verifier from IAM. Getting a token with these credentials is more secure than getting a password verifier because a password verifier is considered sensitive. Using a token means that you do not need to pass or use the verifier. Applications cannot pass a token that was retrieved by the IAM user name and password through the database client API. Only the database client can retrieve this type of token. A database client can only retrieve a database token using the IAM user name and IAM database password.

You can enter the IAM username and IAM database password directly into the tool or use a SEPS wallet to hold these credentials securely.

7.5.5.2 Parameters to Set for Client Connections That Use a Token Requested by an IAM User Name and Database Password

To set these parameters, you modify either the sqlnet.ora file or the tnsnames.ora file.

Token-Specific Parameters for IAM User Name and Database Password Token Requests

PASSWORD_AUTH Parameter

Sets sets the authentication method. This configuration must use a setting of <code>OCI_TOKEN</code>. Getting a token using the user and password credentials is more secure than using a password verifier, since a password verifier is considered sensitive. This parameter is required for retrieving the IAM bearer token with an IAM user name and database password.

Syntax:

PASSWORD AUTH=authentication method

Example:

PASSWORD AUTH=OCI TOKEN

OCI_IAM_URL Parameter

Specifies the IAM URL that the database client must connect with to get the database token. This parameter is required for retrieving the IAM bearer token with an IAM user name and database password. This setting is specific to your region. See Identity and Access Management Data Plane API for the appropriate URL for your region. Then append /v1/actions/generateScopedAccessBearerToken to the regional URL.

Syntax:

 $\label{local_com_v1_action_regional_endpoint.com_v1_actions/generateScopedAccessBearerToken} OCI_IAM_URL=authentication_regional_endpoint.com/v1/actions/generateScopedAccessBearerToken$

Example:

The following example uses the Phoenix URL (https://auth.us-phoenix-1.oraclecloud.com):

https://auth.us-phoenix-1.oraclecloud.com/v1/actions/generateScopedAccessBearerToken

OCI_TENANCY Parameter

Specifies the OCID of the user's tenancy. You can find this setting under the user's icon at the top right of the OCI console. This parameter is required for retrieving the IAM bearer token with an IAM user name and database password.

Syntax:

OCI_TENANCY=tenancy_OCI..OCID



Example:

OCI TENANCY=ocid1.tenancy.region1..12345

OCI COMPARTMENT Parameter

Specifies defines the scope of the database token request. Note that there are two periods after <code>region_name</code>. The token will only be usable for databases in the specified compartment. If you omit this value, then the entire tenancy is the scope of the request. This parameter is optional, except if <code>OCI_DATABASE</code> is set.

Syntax:

OCI COMPARTMENT=compartment OCID

Example:

Note that there are two periods after region1.

OCI COMPARTMENT=ocid1.compartment.region1..12345

OCI DATABASE Parameter

Specifies the OCID of the database to access. This parameter limits the token to the database only. This parameter is optional.

Syntax:

OCI DATABASE=database OCID

Example:

OCI_DATABASE=ocid1.autonomousdatabase.oc1.iad.12345

DN-Specific Parameters for IAM User Name and Database Password Token Requests

SSL_SERVER_CERT_DN Parameter

Specifies the distinguished name (DN) of the database server for this client. (Note that this parameter is not specific to the bearer tokens.)

Syntax:

SSL SERVER CERT DN=DN

Example:

SSL_SERVER_CERT_DN="C=US,O=ExampleCorporation,CN=sslserver2"

SSL_SERVER_DN_MATCH Parameter

Enforces server-side validation through DN matching. Set this parameter to TRUE.

Syntax:

SSL_SERVER_DN_MATCH=TRUE | FALSE



Example:

```
SSL SERVER DN MATCH=TRUE
```

sqlnet.ora Example

```
PASSWORD_AUTH=OCI_TOKEN

OCI_IAM_URL=https://auth.region1.example.com/v1/actions/
generateScopedAccessBearerToken

OCI_TENANCY=ocid1.tenancy..12345

OCI_COMPARTMENT=ocid1.compartment.region1..12345

OCI_DATABASE=ocid1.autonomousdatabase.oc1.iad.12345

SSL_SERVER_CERT_DN="C=US,O=ExampleCorporation,CN=sslserver2",
SSL_SERVER_DN_MATCH=TRUE
```

tnsnames.ora Example

In this specification:

- (PROTOCOL=tcps) sets the protocol to TCPS. You must use TCPS as the protocol or the
 connection will fail. TCPS must be enabled when passing tokens from the database client
 to the server.
- SECURITY is where you set the authentication and DN parameters.

7.5.5.3 Configuring the Database Client to Retrieve a Token Using an IAM User Name and Database Password

You can configure the database client to retrieve the IAM database token using the provided IAM user name and IAM database password.

- 1. Log in to the Oracle DBaaS client.
- 2. Set the appropriate parameters to retrieve a token that will be requested by an IAM user name and database password.
- 3. In the sqlnet.ora file, set the WALLET_LOCATION parameter to the location of the client. The root certificates will reside in this directory.

For example:

Related Topics

 Parameters to Set for Client Connections That Use a Token Requested by an IAM User Name and Database Password

To set these parameters, you modify either the sqlnet.ora file or the tnsnames.ora file.

7.5.5.4 Configuring a Secure External Password Store Wallet to Retrieve an IAM Token

You can enable an IAM user name and a secure external password store (SEPS) to request the IAM database token.

- 1. Log in to the Oracle DBaaS client.
- 2. Configure this client to use the secure external password store.
- Set the appropriate parameters to retrieve a token that will be requested by an IAM user name and database password.

Related Topics

- Configuring a Client to Use the Secure External Password Store
 You can configure a client to use the secure external password store feature by using the
 mkstore command-line utility.
- Parameters to Set for Client Connections That Use a Token Requested by an IAM User Name and Database Password

To set these parameters, you modify either the sqlnet.ora file or the tnsnames.ora file.

7.5.6 Client Connections That Use a Token Requested by a Client Application or Tool

For IAM token access to the Oracle DBaaS, the client application or tool requests a database token from IAM for the IAM user.

The client application will pass the database token directly to the database client through the database client API.

If the application or tool has not been updated to request an IAM token, then the IAM user can use Oracle Cloud Infrastructure (OCI) command line interface (CLI) to request and store the database token. You can request a database access token (db-token) using the following credentials:

- Security tokens (with IAM authentication), delegation tokens (in the OCI cloud shell) and API-keys, which are credentials that represent the IAM user to enable the authentication
- Instance principal tokens, which enable instances to be authorized actors (or principals) to perform actions on service resources after authenticating

 Resource principal token, which is a credential that enables the application to authenticate itself to other Oracle Cloud Infrastructure services

When the IAM users logs into the client with a slash / login and the OCI_IAM parameter is configured (sqlnet.ora, tnsnames.ora, or as part of a connect string), then the database client retrieves the database token from a file. If the IAM user submits a user name and password, the connection will use the IAM database verifier access described for client connections that use IAM database password verifiers. The instructions in this guide show how to use the OCI CLI as a helper for the database token. If the application or tool has been updated to work with IAM, then follow the instructions for the application or tool. Some common use cases include the following: SQLPlus on-premises, SQLcl on-premises, SQL*Plus in Cloud Shell, or applications that use SEP wallets.

Related Topics

Client Connections That Use an IAM Database Password Verifier
 After you have configured the authorization needed for the IAM user, this user can log in using existing client application, such as SQL*Plus or SQLcl without additional configuration.

7.5.7 TLS Connections without Client Wallets

The use of Transport Layer Security (TLS) connections without client wallets is supported for IAM connections.

Before you configure this type of connection, ensure that the Oracle DBaaS environment meets the requirements.

Related Topics

Transport Layer Security Connection without a Client Wallet
 A Transport Layer Security (TLS) connection that uses a common root certificate for the database server does not require a client wallet.

7.5.8 Common Database Client Configurations

IAM users can connect to the Oracle DBaaS instance using client tools such as SQLcI on a laptop.

- Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password You can configure SQL*Plus to use an IAM database password.
- Configuring a Client Connection for SQL*Plus That Uses an IAM Token You can configure a client connection for SQL*Plus that uses an IAM token.

7.5.8.1 Configuring a Client Connection for SQL*Plus That Uses an IAM Database Password

You can configure SQL*Plus to use an IAM database password.

 As the IAM user, log in to the Autonomous Database instance by using the following syntax:

CONNECT user_name@db_connect_string
Enter password: password

In this specification, user_name is the IAM user name. There is a limit of 128 bytes for the combined domain name/user name.

The following example shows how IAM user peter_fitch can log in to an Autonomous Database instance.

```
sqlplus /nolog
connect peter_fitch@db_connect_string
Enter password: password
```

Some special characters will require double quotation marks around <code>user_name</code> and <code>password</code>. For example:

```
"peter_fitch@example.com"@db_connect_string
"IAM database password"
```

7.5.8.2 Configuring a Client Connection for SQL*Plus That Uses an IAM Token

You can configure a client connection for SQL*Plus that uses an IAM token.

- 1. Ensure you have an IAM user account.
- Check with an IAM administrator and an Oracle Database administrator to ensure you have a policy allowing you to access the database in the compartment or your tenancy and that you are mapped to a global schema in the database.
- If your application or tool does not support direct IAM integration, then download, install, and configure the OCI CLI. (See OCI Command Line Interface Quickstart.) Set up an API key as part of the OCI CLI configuration and select default values.
 - a. Set up the API key access for the IAM user.
 - **b.** Retrieve the db-token. For example:
 - Retrieving a db-token with an API-key using the Oracle Cloud Infrastructure (OCI) command-line interface:

```
oci iam db-token get
```

Retrieving a db-token with a security (or session) token:

```
oci iam db-token get --auth security_token
```

If the security token has expired, a window will appear so the user can log in to OCI again. This generates the security token for the user. OCI CLI will use this refreshed token to get the <code>db-token</code>.

 Retrieving a db-token with a delegation token: When you log in to the cloud shell, the delegation token is automatically generated and placed in the /etc directory.
 To get this token, execute the following command in the cloud shell:

```
oci iam db-token get
```



Retrieving an instance token by using the OCI command-line interface:

```
oci iam db-token get --auth instance principal
```

c. The database client can also be configured to retrieve a database token using the IAM username and IAM database password.

See Client Connections That Use a Token Requested by an IAM User Name and Database Password for more information.

See Required Keys and OCIDs for more information.

4. Ensure that you are using the latest release updates for the Oracle Database client releases 19c and 21c.

This configuration only works with the Oracle Database client release 19c or 21c.

- **5.** Follow the existing process to download the wallet from the Autonomous Database and then follow the directions for configuring it for use with SQL*Plus.
 - a. Confirm that DN matching is enabled by looking for SSL_SERVER_DN_MATCH=ON in sqlnet.ora.
 - b. Configure the database client to use the IAM token by adding <code>TOKEN_AUTH=OCI_TOKEN</code> to the <code>sqlnet.ora</code> file. Because you will be using the default locations for the database token file, you do not need to include the token location.

The TOKEN_AUTH and TOKEN_LOCATION values in the tnsnames.ora connect strings take precedence over the sqlnet.ora settings for that connection. For example, for the connect string, assuming that the token is in the default location (~/.oci/db-token for Linux):

After the connect string is updated with the <code>TOKEN_AUTH</code> parameter, the IAM user can log in to the Autonomous Database instance by running the following command to start SQL*Plus. You can include the connect descriptor itself or use the name of the descriptor from the <code>tnsnames.ora</code> file.

```
connect /@exampledb high
```

Or:

```
connect /@(description=
  (retry_count=20) (retry_delay=3)
  (address=(protocol=tcps) (port=1522)
  (host=example.us-phoenix-1.oraclecloud.com))

(connect_data=(service_name=aaabbbccc_exampledb_high.example.oraclecloud.com))
  (security=(ssl server cert dn="CN=example.uscom-east-1.oraclecloud.com,"))
```

```
OU=Oracle BMCS US, O=Example Corporation,
L=Redwood City, ST=California, C=US")
(TOKEN AUTH=OCI TOKEN)))
```

The database client is already configured to get a db-token because <code>TOKEN_AUTH</code> has already been set, either through the <code>sqlnet.ora</code> file or in a connect string. The database client gets the db-token and signs it using the private key and then sends the token to the Autonomous Database. If an IAM user name and IAM database password are specified instead of slash /, then the database client will connect using the password instead of using the <code>db-token</code>.

7.6 Accessing a Database Cross-Tenancy Using an IAM Integration

Users and groups in one tenancy can access DBaaS database instances in another tenancy if policies in both tenancies allow this.

- About Cross-Tenancy Access for IAM Users to DBaaS Instances
 Cross-tenancy access to an Oracle Cloud Infrastructure (OCI) DBaaS instance is similar to
 a single tenancy scenario except that tenancy information is required for mappings and
 token requests and a policy is required in both tenancies to allow this cross tenancy
 database resource access.
- Configuring Policies
 You must create policies in both the user tenancy and the database resource tenancy to allow cross-tenancy database access.
- Mapping Database Schemas and Roles to Users and Groups in Another Tenancy
 When you perform this type of mapping, you must add the tenancy OCID to the mapping
 information so the database knows it is cross-tenancy access.
- Configuring Database Clients for Cross-Tenancy Access You can configure some database clients directly.
- Requesting Cross-Tenancy Tokens Using the OCI Command-Line Interface
 You must add the --scope parameter to the Oracle Cloud Infrastructure (OCI) command line interface command to get a db-token for a cross-tenancy request.

7.6.1 About Cross-Tenancy Access for IAM Users to DBaaS Instances

Cross-tenancy access to an Oracle Cloud Infrastructure (OCI) DBaaS instance is similar to a single tenancy scenario except that tenancy information is required for mappings and token requests and a policy is required in both tenancies to allow this cross tenancy database resource access.

The following figure illustrates the process for a cross-tenancy access to an OCI DBaaS instance.

Get db-token

User Tenancy

Domain

User/Application

Policy

Access DB

Query group

Databases

Databases

Figure 7-4 Cross-Tenancy Access to an OCI DBaaS Instance

The cross-tenancy process is as follows:

- 1. The policy is required in both tenancies to endorse and admit access cross tenancy.
- 2. The IAM principal (user or application) requests a db-token for a cross-tenancy resource.
- 3. The db-token is returned and is used to access the database in a different tenancy
- **4.** The database will make a cross-tenancy group query for the user's groups and map principal to global schema and optional global roles.

You must subscribe the user tenancy to the same regions in which the databases are located. For example, if the databases in the database tenancy are in the PHX and IAD regions, then you must subscribe the user tenancy to these regions. This is not the home region, just the additional subscribed regions in the user tenancy.

7.6.2 Configuring Policies

You must create policies in both the user tenancy and the database resource tenancy to allow cross-tenancy database access.

- Configuring the Source User Tenancy
 Two policies are required to allow cross-tenancy access in the user tenancy.
- Configuring the Target Database Resource Tenancy
 The database tenancy will need matching policies to enable access to the users from the user tenancy as well as allow its own databases to query group information in the user tenancy
- Policy Examples for Cross-Tenancy Access
 Examples include using a WHERE clause to refine the cross-tenancy configuration, and other methods of performing this type of configuration.

7.6.2.1 Configuring the Source User Tenancy

Two policies are required to allow cross-tenancy access in the user tenancy.

The first policy is to allow a user tenancy group to access a database in a different tenancy. The second policy allows a database in the database tenancy to query group information in the user tenancy.

- 1. In the OCI console, select Identity & Security.
- 2. Under Identity, select Policies.
- 3. Click Create Policy and in the Policy Builder select Show manual editor.
- 4. Use the DEFINE statement to make it easier to read the actual policies.

For example:

```
DEFINE tenancy database tenancy as ocid1.tenancy.OCID
```

5. Endorse the tenancy group domainA/xt_db_users to use database_connections in tenancy database tenancy.

This allows users of the xt_db_users group in domainA to access any database in tenancy database_tenancy.

```
ENDORSE group domainA/xt_db_users to use database-connections in tenancy database tenancy
```

6. Use the ADMIT statement to create an Admit policy to allow any database in the database tenancy to query group information for specific IAM users in the user tenancy.

```
ADMIT any-user of tenancy database_tenancy to {GROUP_MEMBERSHIP_INSPECT, AUTHENTICATION INSPECT} in tenancy
```

7.6.2.2 Configuring the Target Database Resource Tenancy

The database tenancy will need matching policies to enable access to the users from the user tenancy as well as allow its own databases to query group information in the user tenancy

- 1. In the OCI console, select Identity & Security.
- Under Identity, select Policies.
- 3. Click Create Policy and in the Policy Builder select Show manual editor.
- 4. Use DEFINE to make it easier to troubleshoot and read the policies.

```
DEFINE tenancy user_tenancy as ocid1.tenancy.OCID
DEFINE group xt db users as ocid1.group.defg
```

5. Use ADMIT to create an Admit policy in the tenancy to match the Endorse policy from the user tenancy.



The Admit policy must match the ENDORSE policy in the user tenancy so that it can enable users from the user tenancy to access databases in this tenancy.

```
ADMIT group xt_db_users of tenancy user_tenancy to use database-connections in tenancy
```

6. Create an Endorse policy, which will match the Admit policy created in the User tenancy.

The Endorse policy will enable databases in the database tenancy to query group information from the user_tenancy.

```
ENDORSE any-user to {GROUP_MEMBERSHIP_INSPECT, AUTHENTICATION_INSPECT} in tenancy user tenancy
```

While using any-user makes it easy to understand the required policies, Oracle recommends that you use stronger constraints in addition to or instead of using any-user. The any-user option will allow any principal or resource to query user groups in the user_tenancy. Ideally, you should limit this to just allowing the database resources (resource principals) to make the group queries. You can do this by adding a WHERE clause to the policies or by adding a dynamic group that limits it to the members of the dynamic group. Defining every possible way to specify dynamic groups and policies is outside the scope of this topic. You can find more information from these sources:

- Managing Dynamic Groups
- Managing Policies

7.6.2.3 Policy Examples for Cross-Tenancy Access

Examples include using a WHERE clause to refine the cross-tenancy configuration, and other methods of performing this type of configuration.

You can add a WHERE clause to limit the database resources allowed to make the cross-tenancy group query:

```
ADMIT any-user of tenancy db_tenancy to {GROUP_MEMBERSHIP_INSPECT, AUTHENTICATION_INSPECT} in tenancy where request.principal.type = 'dbsystem'
```

This Admit policy allows any Base Database Service (resource type: dbsystem) in the db_tenancy to query a user's group information from the user tenancy. Resource type names are in the table below.

A similar method can be done by putting the same resource type into a dynamic group:

```
dynamic group: db_principals
any {resource.type = 'dbsystem', resource.type = 'vmcluster', resource.type =
'cloudvmcluster'}
```

The dynamic group in the preceding example includes database instances for Oracle Base Database Service (dbsystem), Oracle Exadata Cloud@Customer (vmcluster), and Oracle Exadata Database Service (cloudvmcluster).



This example uses a dynamic group instead of any-user:

```
ADMIT dynamic group db_principals of tenancy db_tenancy to {GROUP_MEMBERSHIP_INSPECT, AUTHENTICATION_INSPECT} in tenancy
```

You can also add all resource principals in a compartment using resource.compartment.id. However, this might also allow other non-database resource principals to make the cross-tenancy group query. The following table provides a mapping of the various resource types with the DBaaS platform name:

DBaaS Platform Name	Resource Type Name
ADB-S	autonomousdatabase
ADB-D (OPC)	cloudautonomousvmcluster*
Base DBS	dbsystem
ExaCS	cloudvmcluster
ExaCC	vmcluster

^{*} Older ADBD instances may still be using the autonomous exainfrastructure resource type.

7.6.3 Mapping Database Schemas and Roles to Users and Groups in Another Tenancy

When you perform this type of mapping, you must add the tenancy OCID to the mapping information so the database knows it is cross-tenancy access.

Use a full colon to separate the tenancy OCID when you use the CREATE USER and CREATE ROLE statements in SQL*Plus.

To use the CREATE USER statement to perform the mapping:

The following examples show exclusive and shared schema mapping with principals and groups in default and non-default domains. When using default domains, you do not need to include a domain name.

```
CREATE USER schema1 IDENTIFIED GLOBALLY
AS 'IAM_PRINCIPAL_NAME=ocid1.tenancy.OCID:example_domain/
peter.fitch@oracle.com';

CREATE USER schema2 IDENTIFIED GLOBALLY
AS 'IAM_PRINCIPAL_NAME=ocid1.tenancy.OCID:peter.fitch@oracle.com';

CREATE USER qa_db_user_group IDENTIFIED GLOBALLY
AS 'IAM_GROUP_NAME=ocid1.tenancy.OCID:example_domain/xt_db_users';

CREATE USER qa_sales_user_group IDENTIFIED GLOBALLY
AS 'IAM_GROUP_NAME=ocid1.tenancy.OCID:sales_users';

CREATE USER xt_ip_user IDENTIFIED GLOBALLY
AS 'IAM_PRINCIPAL_OCID=ocid1.instance.region1.sea.OCID';
GRANT CREATE SESSION TO xt_ip_user;

CREATE USER xt iam dg IDENTIFIED GLOBALLY
```

```
AS 'IAM_GROUP_NAME=ocid1.tenancy.region1.OCID:sales_principals';
GRANT CREATE SESSION TO xt iam dg;
```

To use the CREATE ROLE statement to perform the mapping:

The following examples show global role mapping with groups in default and non-default domains. When using default domains, you do not need to include a domain name.

```
CREATE ROLE globalrole1 IDENTIFIED GLOBALLY

AS 'IAM_GROUP_NAME=ocid1.tenancy.abcdef:example_domain/xt_db_users';

CREATE ROLE globalrole2 IDENTIFIED GLOBALLY

AS 'IAM GROUP NAME=ocid1.tenancy.abcdef:sales users';
```

7.6.4 Configuring Database Clients for Cross-Tenancy Access

You can configure some database clients directly.

The database tenancy must be identified in either the connect string or in sqlnet.ora if the client is configured to directly get the access token from OCI IAM. Review client-specific documentation for specific parameter values (JDBC-thin, ODP.NET-core, managed).

7.6.5 Requesting Cross-Tenancy Tokens Using the OCI Command-Line Interface

You must add the --scope parameter to the Oracle Cloud Infrastructure (OCI) command-line interface command to get a db-token for a cross-tenancy request.

See Optional Parameters for more details about using the optional parameters of the oci get command.

You can scope it for the entire tenancy or scope it to a compartment or database in the tenancy. When scoping for cross tenancy compartment or database, you do not need to also add the tenancy information because the compartment and database OCIDs are unique across OCI.

Certain clients can request the tokens directly from MSEI. Refer to their documentation on setting the parameters to get the MSEI <code>OAuth2</code> access tokens.

7.7 Database Links in an Oracle DBaaS-to-IAM Integration

The use of database links when accessing the Oracle DBaaS database using IAM credentials is supported.

The method of configuring database links for Oracle DBaaS connections to IAM depends on the Oracle DBaaS platform. Review the topic below that corresponds to your Oracle DBaaS platform and then click on the associated link for more information.

Oracle Autonomous Database Serverless: You can use fixed user database links in
which a database user is used for the fixed database link. The database user for creating
the database link can only use password authentication with the database link. The IAM
user can authenticate to the source database using either password or token access. You
cannot configure IAM users as fixed database links, nor can you use connected or current
user database links. See Using Oracle Autonomous Database Serverless

• Oracle Autonomous Database on Dedicated Exadata Infrastructure and all non-Autonomous Database DBaaS platforms: You can use connected user and fixed user database links, but not current user database links. For connected user database links, an IAM user must be provisioned to both the source and target link databases. You can use a database password verifier or an IAM database token to connect and use connected user database links. For a fixed user database link, a user can connect to the target database using a target database user with password authentication. In addition, an IAM user can connect to the first PDB by using an IAM user name and password or an IAM token. See Using Oracle Autonomous Database on Dedicated Exadata Infrastructure

7.8 Troubleshooting IAM Connections

The ORA-01017: invalid username/password; logon denied error can be caused by several different issues throughout the Oracle DBaaS integration with Identity and Access Management (IAM).

- Areas to Check on the Client-Side for ORA-01017 Errors
 Client-side ORA-01017 errors can result from problems with IAM credentials, client configuration, or problems with the IAM profile.
- Database Client Trace Files
 You can generate two levels of trace files to troubleshoot IAM connections on client side.
- Check in the Oracle Cloud Infrastructure IAM and the Oracle Database for ORA-01017
 Errors
 ORA-01017 errors in the Oracle Database instance can arise from the way that the
- ORA-01017 Errors Caused by Improperly Configured IAM Users
 Several ORA-01017 errors can arise from improperly configured IAM users.
- ORA-12599 and ORA-03114 Errors Caused When Trying to Access a Database Using a Token

The ORA-12599: TNS: cryptographic checksum mismatch and ORA-03114: not connected to ORACLE errors indicate that the database to which you are trying to connect is protected by native network encryption.

Actions IAM Administrators Can Take to Address ORA-01017 Errors
 Several actions to remedy ORA-01017 errors can only be performed by IAM administrators.

7.8.1 Areas to Check on the Client-Side for ORA-01017 Errors

database was enabled to work with IAM.

Client-side ORA-01017 errors can result from problems with IAM credentials, client configuration, or problems with the IAM profile.

Troubleshooting the IAM Token

Check the version of the Oracle Cloud Infrastructure (OCI) CLI used for the token.
 The OCI CLI must be at least OCI version 3.4, which includes the command to get the new db-token from IAM. To check the version of OCI, run the following command:

oci --version

• Check the Oracle Database Client version. You can find the latest version by checking the Oracle Database documentation. Currently, only the following drivers are supported:



- JDBC: Version 19.13.0.0.1 and later versions of 19c JDBC clients JDBC: Version 21.5 and later versions of 21c
- Instant Client/SQL*Plus (Linux only): Version 19.13 (annotated with -2) and later versions of 19c
- Instant Client/OCI/SQL*Plus (Linux only): Version 21.5 and later versions of 21c (Not all features are supported with Instant Client/OCI version 21c. Oracle recommends that you use the latest 19c or version 23c client, if possible.)
- SQLcl: version 21.4 and later
- ODP.net: Version 19.13 and higher versions of 19c
- ODP.net: Version 21.4 and higher versions of 21c
- Oracle Database release 23c: All clients

The latest version of these drivers is needed when you use IAM tokens to access the database. All supported database clients will work when using IAM database passwords.

- Check the token location that was specified in the tnsnames.ora file. The database clients and OCI CLI use the same default location for storing and retrieving database tokens and the private key (~/.oci/db-token). You can specify a different location, but both OCI CLI and the database client must be configured to use the same directory. Ensure that the correct TOKEN_LOCATION value is specified in the connect string, in the tnsnames.ora or sqlnet.ora file. The connect string takes precedence over tnsnames.ora, which takes precedence over the value of TOKEN_LOCATION in sqlnet.ora.
- Check if the token has expired. The IAM database token is only valid for one hour. After
 the database token has expired, re-run the following OCI CLI command to request another
 token if you are using an API-key:

```
oci iam db-token get
```

- Check the TOKEN_AUTH parameter value in tnsnames.ora. Ensure that the parameter TOKEN_AUTH=OCI_TOKEN is set in either the connect string, tnsnames.ora, Or sqlnet.ora. The connect string takes precedence over tnsnames.ora, which takes precedence over sqlnet.ora for the value of TOKEN_AUTH.
- Check if there is a missing token or private key from the default user-specified token location. Ensure that both the token and the private key are in the directory that is specified by the TOKEN_LOCATION after you run the OCI CLI command oci cli db-token get. You can find the db-token and private key location by running the following command:

```
[oracle@localhost ~]$ oci iam db-token get
```

Output similar to the following appears:

```
Private key written at /home/oracle/.oci/db-token/oci_db_key.pem db-token written at: /home/oracle/.oci/db-token/token db-token is valid until 2022-01-05 15:36:51
```

If the location does not match the ${\tt TOKEN_LOCATION}$ setting, either update the OCI CLI command or update the ${\tt TOKEN_LOCATION}$ parameter.

Check your OCI IAM profile.



- Ensure that the public API-key exists in the OCI user account. The OCI CLI will default
 to using the API-key on the client to request a db-token from IAM. If the public APIkey is not in the OCI user account, then IAM will not return a database token.
- Ensure that the IAM account is not locked. If it is, then ask the IAM administrator to unlock it.
- If you are using the IAM database password, then ensure that you set the IAM database password in your IAM profile.
- If you are not using the API-key, then explicitly state that you are using the security token. Use the following command:

```
oci iam db-token get --auth security token
```

If the security token does not exist or has expired, this command will try to open the browser for you to sign into IAM (or your federated IdP). This command will fail if you do not have a browser in your environment.

Troubleshooting Both the IAM Database Password and the IAM Token

 Check client tracing on Oracle Instant Client only. Client tracing can provide some information when you use SQL*Plus with the Instant Client. You can generate Oracle Database client trace files using two different tracing levels.

Related Topics

Database Client Trace Files
 You can generate two levels of trace files to troubleshoot IAM connections on client side.

7.8.2 Database Client Trace Files

You can generate two levels of trace files to troubleshoot IAM connections on client side.

The two levels of trace files that you can generate are as follows:

- Low level tracing prints traces in case of failures:
 - If TCPS is not set up for the IAM connection, then it prints a message that the protocol has to be TCPS.
 - If SSL_SERVER_DN_MATCH is not set to TRUE, then it prints a message that the value is FALSE.
 - If an invalid TOKEN_LOCATION has been specified, then it prints a message that the token location does not exist.
 - If the db-token and private key are not present at the specified TOKEN_LOCATION or the default token location, then it prints a message.
 - If the application has passed in only db-token or private key, it prints a message for the missing attribute.
 - If the db-token has expired, then it prints a message.
- High level tracing prints traces in case of failure as mentioned above. In addition, it prints traces in case of success, as follows:
 - It prints where SSL_SERVER_DN_MATCH is present, tnsnames.ora or sqlnet.ora. It also prints the value as TRUE if set to TRUE.



- If both the db-token and private key are set by the application, then it prints a
 message.
- If TOKEN AUTH has the correct value OCI TOKEN, then it prints the value.
- If db-token is not expired, then it prints a message.

To control client tracing for IAM connections, you can use one of these methods:

- Add the following settings to the client side sqlnet.ora file:
 - EVENT 25701=14 for low level tracing
 - EVENT 25701=15 for high level tracing
- Set the environment variable EVENT 25701:
 - EVENT 25701=14 for low level tracing
 - EVENT 25701=15 for high level tracing

Client trace files are created in the following locations:

- Linux: \$ORACLE_HOME/log/diag/clients
- Windows: %ORACLE HOME%\log\diag\clients

You can use the ADR_BASE parameter in the client side sqlnet.ora to specify the directory in which tracing messages are stored. Ensure that the directory path is valid and has write permissions. Ensure that the diag_adr_enabled parameter is not set to false.

An example of setting ADR_BASE is as follows:

```
ADR BASE=/oracle/iam/trace
```

7.8.3 Check in the Oracle Cloud Infrastructure IAM and the Oracle Database for ORA-01017 Errors

ORA-01017 errors in the Oracle Database instance can arise from the way that the database was enabled to work with IAM.

Check if the IAM configuration has been enabled. The OCI server must be configured for IAM integration and one or more database schemas (database users) must be mapped to IAM users or groups. This applies to both the IAM token and IAM database password use cases. To check if the configuration has been enabled, run the following command in SQL*Plus:

```
SELECT NAME, VALUE
FROM V$PARAMETER
WHERE NAME='identity_provider_type';
```

Alternatively, you can use this command:

```
SHOW PARAMETER IDENTITY PROVIDER TYPE
```

If the returned value does not equal OCI IAM, then enable the external authentication.

Check the schemas that have been mapped to IAM. Note which IAM users and IAM
groups are used in the mapping. You can find this information by running the following
query in SQL*Plus:

```
SELECT USERNAME, EXTERNAL_NAME, CREATED FROM DBA_USERS
WHERE AUTHENTICATION TYPE='GLOBAL';
```

In the output, check that there is at least one <code>EXTERNAL_NAME</code> that starts with either <code>IAM_USER</code> or <code>IAM_GROUP</code>. Make a note of the IAM user or group name. If there are no global schemas, then you must create a new schema, or alter an existing schema, and then map it to an IAM user or IAM group that the user is a member of.

Check if the Oracle Database instance needs to be restarted. In some cases, a
database instance that existed before the IAM configuration was introduced may need to
be restarted. But before doing so, follow all other troubleshooting guidelines before trying
to restart the database.

Related Topics

Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications
 An Oracle DBaaS database administrator can map IAM users and Oracle Cloud
 Infrastructure (OCI) applications to the Oracle Database global schemas and global roles.

7.8.4 ORA-01017 Errors Caused by Improperly Configured IAM Users

Several ORA-01017 errors can arise from improperly configured IAM users.

- Ensure that the IAM user can log in to the Oracle DBaaS instance. Ask the IAM user to try logging in an IAM user but not as a federated user. Ensure that this user is not locked out of the account. (The user should contact an IAM administrator if this happens.) If the user's IAM account is locked, then this user cannot log in to the Oracle DBaaS instance. You should also check the IAM user name and IAM groups that the user is a member of. One of these (user name or group names) should match the mapped IAM user and group name that you found from the Oracle DBaaS server. If there is no mapping, then the user will be denied access to the database. If this is the case, then an IAM administrator should add the user to an IAM group that is mapped to the DBaaS instance that the user needs to access.
- Ensure that the API public key is registered in the IAM user profile. If the Oracle DBaaS instance configuration with IAM uses tokens, and if you use an API-key to retrieve the database token, then the API public key needs to be registered in the user's IAM user profile.
- Ensure that the IAM database password has been set in the IAM user profile. If the Oracle DBaaS instance configuration with IAM uses database password authentication, then ensure that an IAM database password has been set in the user IAM user profile. In addition, ensure that Database Passwords is an allowed setting in the User Capability section of the IAM user profile.

Related Topics

Configuring Authorization for IAM Users and Oracle Cloud Infrastructure Applications
 An Oracle DBaaS database administrator can map IAM users and Oracle Cloud
 Infrastructure (OCI) applications to the Oracle Database global schemas and global roles.



7.8.5 ORA-12599 and ORA-03114 Errors Caused When Trying to Access a Database Using a Token

The ORA-12599: TNS: cryptographic checksum mismatch and ORA-03114: not connected to ORACLE errors indicate that the database to which you are trying to connect is protected by native network encryption.

When tokens are being used to access an Oracle database, a Transport Layer Security (TLS) connection must be established, not network native encryption. To remedy these errors, ensure that TLS is properly configured for your database. You should test the configuration with a local database user name and password and check the following SYSCONTEXT USERENV parameters:

- NETWORK PROTOCOL
- TLS_VERSION

Related Topics

Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.

7.8.6 Actions IAM Administrators Can Take to Address ORA-01017 Errors

Several actions to remedy ORA-01017 errors can only be performed by IAM administrators.

- Check if the IAM user needs to recreate API-keys. If the IAM user was deleted and then recreated with the exact same user name, then Oracle Cloud Infrastructure (OCI) IAM will consider this as a different user with a different user OCID. In this case, the IAM user will need to recreate their user account and API-key. This action does not affect the IAM user and IAM group mappings in the database.
- If necessary, unlock the IAM user account. If the user is inactive or otherwise locked, then an IAM administrator will need to unlock the user account before database access can be allowed.
- Check the IAM policy. An IAM policy is required to allow the user to use IAM database tokens to access the database. The resource is called database-connections and it is also a member of the autonomous-database-family. You do not need to create IAM policies if the Oracle DBaaS instance uses IAM database passwords. When you configure the IAM policy, remember that the use or manage tag is required for the policy. For example:
 - Set allow all-users to use autonomous-database-family in the tenancy. This
 enables all IAM tenancy users to use IAM database tokens to access all Oracle
 DBaaS instances in the tenancy.
 - Set allow group DBUsers to use database-connections in the production_compartment compartment. This enables IAM users who are members of the DBUsers IAM group to use IAM tokens to access databases in the production compartment compartment.
- Check the mappings for IAM users and groups. The IAM user either has an exclusive mapping from a schema (that is, a database user) in the database or is a member of an IAM group that is mapped to a schema in the database. Run the following SQL*Plus query



and review its output to find the mapped IAM users and groups. Ensure that the user has one mapping to a database schema.

```
SELECT USERNAME, EXTERNAL_NAME, FROM DBA_USERS
WHERE AUTHENTICATION_TYPE='GLOBAL';
```

Related Topics

Creating an IAM Policy to Authorize Users Authenticating with Tokens
 To configure IAM to work with the Oracle DBaaS instance, an IAM administrator must
 create an IAM policy (if using IAM tokens), create IAM groups and manage group
 membership.



Authenticating and Authorizing Microsoft Azure Active Directory Users for Oracle Databases

An Oracle Database can be configured for Microsoft Azure AD users to connect using single-sign on.

- Introduction to Oracle Database Integration with Microsoft Azure AD
 Before you begin configuring Microsoft Azure AD to access an Oracle database, you must
 understand the overall process.
- Configuring the Oracle Database for Microsoft Azure AD Integration
 The Microsoft Azure AD integration with the Oracle Database instance requires the database to be registered with Azure AD.
- Mapping Oracle Database Schemas and Roles
 Azure AD users will be mapped to one database schema and optionally to one or more database roles.
- Configuring Azure AD Client Connections to the Oracle Database
 You can configure client connections to connect with the Azure AD registered database
- Configuring Microsoft Azure AD Proxy Authentication
 Proxy authentication allows an Azure AD user to proxy to a database schema for tasks such as application maintenance.
- Troubleshooting Microsoft Azure AD Connections
 You can use trace files to diagnose problems with Microsoft Azure AD connections. You
 also can easily remedy ORA-12599 and ORA-03114 errors.

8.1 Introduction to Oracle Database Integration with Microsoft Azure AD

Before you begin configuring Microsoft Azure AD to access an Oracle database, you must understand the overall process.

- About Integrating Oracle Database with Microsoft Azure AD
 Oracle Database and Microsoft Azure AD can be configured to allow users and applications to connect to the database using their Azure AD credentials.
- Architecture of Oracle Database Integration with Microsoft Azure AD
 Microsoft Azure Active Directory access tokens follow the OAuth 2.0 standard with
 extensions.
- Azure AD Users Mapping to an Oracle Database Schema and Roles
 Microsoft Azure users must be mapped to an Oracle Database schema and have the
 necessary privileges (through roles) before being able to authenticate to the Oracle
 Database instance.

- Use Cases for Connecting to an Oracle Database Using Azure AD
 Oracle Database supports several use cases for connecting to the database.
- General Process of Authenticating Microsoft Azure AD Identities with Oracle Database
 The Oracle Database administrator and the Microsoft Azure AD administrator play roles to allow Azure AD users to connect to the database using Azure AD OAuth2 access tokens.

8.1.1 About Integrating Oracle Database with Microsoft Azure AD

Oracle Database and Microsoft Azure AD can be configured to allow users and applications to connect to the database using their Azure AD credentials.

Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to access the standalone and CDB multitenant databases. This is done with an Azure AD <code>OAuth2</code> access token that the user or application first requests from Azure AD. This <code>OAuth2</code> access token contains the user identity and database access information and is then sent to the database. Refer to Refer to the Microsoft article <code>Passwordless</code> authentication options for Azure Active Directory for information about configuring multi-factor and passwordless authentication.

You can perform this integration in the following Oracle Database environments:

- On-premises Oracle Database release 19.18 and later, excluding 21c
- All Oracle Database server platforms: Linux, Windows, AIX, Solaris, and HPUX
- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Autonomous Database on Exadata Cloud@Customer
- Oracle Exadata Database Service on Dedicated Infrastructure
- Oracle Exadata Database Service on Cloud@Customer
- Oracle Base Database Service

The instructions for configuring Azure AD use the term "Oracle Database" to encompass these environments.

This type of integration enables the Azure AD user to access an Oracle Database instance. Azure AD users and applications can log in with Azure AD Single Sign On (SSO) credentials to get an Azure AD OAuth2 access token to send to the database.

The Azure AD administrator creates and registers Oracle Database with Azure AD. Within Azure AD, this is called an app registration, which is short for application registration. This is the digital information that Azure AD must know about the software that is using Azure AD. The Azure AD administrator also creates application (app) roles for the database app registration in Azure AD. App roles connect Azure users, groups, and applications to database schemas and roles. The Azure AD administrator assigns Azure AD users, groups, or applications to the app roles. These app roles are mapped to a database global schema or a global role or to both a schema and a role. An Azure AD user, group, or application that is assigned to an app role will be mapped to a database global schema, global role, or to both a schema and a role. An Oracle global schema can also be mapped exclusively to an Azure AD user. An Azure AD guest user (non-organization user) or an Azure AD service principal (application) can only be mapped to a database global schema through an Azure AD app role. An Oracle global role can only be mapped from an Azure app role and cannot be mapped from an Azure user.

Tools and applications that are updated to support Azure AD tokens can authenticate users directly with Azure AD and pass the database access token to the Oracle Database instance. You can configure existing database tools such as SQL*Plus to use an Azure AD token from a file location. In these cases, Azure AD tokens can be retrieved using tools like Microsoft



PowerShell or Azure CLI and put into a file location. An Azure AD OAuth2 database access tokens are issued with an expiration time. The Oracle Database client driver will ensure that the token is in a valid format and that it has not expired before passing it to the database. The token is scoped for the database, which means that there is information in the token about the database where the token will be used. The app roles the Azure AD principal was assigned to in the database Azure AD app registration are included as part of the access token. The directory location for the Azure AD token should only have enough permission for the user to write the token file to the location and the database client to retrieve these files (for example, just read and write by the user). Because the token allows access to the database, it should be protected within the file system.

Azure AD users can request a token from Azure AD using a number of methods to open an Azure login window to enter their Azure AD credentials.

Oracle Database accepts tokens representing the following Azure AD principals:

- Azure AD user, who is registered user in the Azure AD tenancy
- Guest user, who is registered as a quest user in the Azure AD tenancy
- Service, which is the registered application connecting to the database as itself with the client credential flow (connection pool use case)

Oracle Database supports the following Azure AD authentication flows:

- Interactive flow (also called authorization code flow) using Proof Key for Code Exchange (PKCE), most commonly used for human users (not applications) to authenticate to Azure AD in an client environment with a browser
- Client credentials, which are for database applications that connect as themselves (and not the end-user)
- On-Behalf-Of (OBO), where an application requests an access token on behalf of a logged-in user to send to the database
- Resource owner password credential (ROPC), which is not recommended for production use, but can be used in test environments where a pop-up browser user authentication would be difficult to incorporate. ROPC needs the Azure AD user name and password credential to be part of the token request call.

Related Topics

- Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Oracle Autonomous Database on Exadata Cloud@Customer
- Oracle Exadata Database Service on Dedicated Infrastructure
- Oracle Exadata Database Service on Cloud@Customer
- Oracle Base Database Service

8.1.2 Architecture of Oracle Database Integration with Microsoft Azure AD

Microsoft Azure Active Directory access tokens follow the OAuth 2.0 standard with extensions.

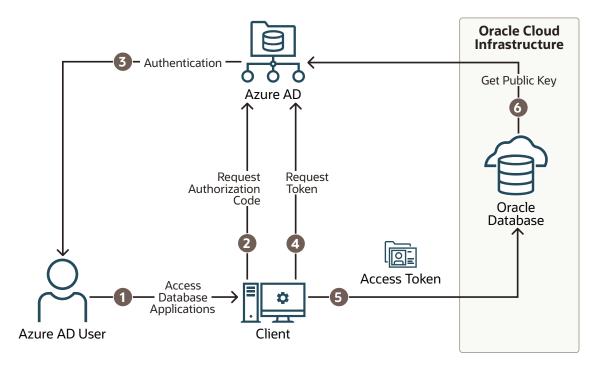
The Azure AD access token will be needed before you access the database from the database client (for example, with SQLPlus or SQLcl). The Oracle clients (for example, OCI, JDBC, and ODP) can be configured to pick up an Azure AD token from a file location or the token can be passed to the client through the database client API. An Azure user can use a script (examples available from Microsoft) to retrieve a token and put it into a file location for the database client to retrieve. Applications can use the Azure SDK to get an access token and pass the token



through the database client API. Command-line tools such as Microsoft PowerShell or the Azure command-line interface can be used to retrieve the Azure AD token if the application cannot directly get the token.

The following diagram is a generalized flow diagram for OAuth 2.0 standard, using the <code>OAuth2</code> token. See Authentication flow support in MSAL in the Microsoft Azure AD documentation for more details about each supported flow.

Figure 8-1 Azure AD User Accessing the Database with the Interactive Authorization Code Flow



The authorization code flow is an OAuth2 standard and is described in detail as part of the standards. There are two steps in the flow. The first step authenticates the user and retrieves the authorization code. The second step uses the authorization code to get the database access token.

- 1. The Azure AD user requests access to the resource, the Oracle Database instance.
- 2. The database client or application requests an authorization code from Azure AD.
- 3. Azure AD authenticates the Azure AD user and returns the authorization code.
- The helper tool or application uses the authorization code with Azure AD to exchange it for the OAuth2 token.
- 5. The database client sends the <code>OAuth2</code> access token to the Oracle database. The token includes the database app roles the user was assigned to in the Azure AD app registration for the database.
- 6. The Oracle Database instance uses the Azure AD public key to verify that the access token was created by Azure AD.

Both the database client and the database server must be registered with the **app registrations** feature in the Azure Active Directory section of the Azure portal. The database client must be registered with Azure AD app registration. Permission must also be granted to allow the database client to get an access token for the database.



8.1.3 Azure AD Users Mapping to an Oracle Database Schema and Roles

Microsoft Azure users must be mapped to an Oracle Database schema and have the necessary privileges (through roles) before being able to authenticate to the Oracle Database instance.

In Microsoft Azure, an Azure AD administrator can assign users, groups, and applications to the database app roles.

Exclusively mapping an Azure AD schema to a database schema requires the database administrator to create a database schema when the Azure AD user joins the organization or is authorized to the database. The database administrator must also modify the privileges and roles that are granted to the database schema to align them with the tasks the Azure AD user is assigned to. When the Azure AD user leaves the organization, the database administrator must drop the database schema so that an unused account is not left on the database. Using the database app roles enables the Azure AD administrator to control access and roles by assigning users to app roles that are mapped to global schemas and global roles. This way, user access to the database is managed by Azure AD administrators and database administrators do not need to create, manage, and drop schemas for every user.

An Azure AD user can be mapped to a database schema (user) either exclusively or through an app role.

- Creating an exclusive mapping between an Azure AD user and an Oracle Database schema. In this type of mapping, the database schema must be created for the Azure AD user. Database privileges and roles that are needed by the Azure AD user must be granted to the database schema. The database schema not only must be created when the Azure AD user is authorized to the database, but the granted privileges and roles must be modified as the Azure AD roles and tasks change. Finally, the database schema must be dropped when the Azure AD user leaves the organization.
- Creating a shared mapping between an Azure AD app role and an Oracle Database schema. This type of mapping, which is more common than exclusive mappings, is for Azure AD users who have been assigned directly to the app role or is a member of an Azure AD group that is assigned to the app role. The app role is mapped to an Oracle Database schema (shared schema mapping). Shared schema mapping allows multiple Azure AD users to share the same Oracle Database schema so a new database schema is not required to be created every time a new user joins the organization. This operational efficiency allows database administrators to focus on database application maintenance, performance, and tuning tasks instead of configuring new users, updating privileges and roles, and removing accounts.

In addition to database roles and privileges being granted directly to the mapped global schema, additional roles and privileges can be granted through mapped global roles. Different Azure AD users mapped to the same shared global schema may need different privileges and roles. Azure app roles can be mapped to Oracle Database global roles. Azure AD users who are assigned to the app role or are a member of an Azure AD group that is assigned to the app role will be granted the Oracle Database global role when they access the database.

The following diagram illustrates the different types of assignments and mappings that are available.



Azure AD

Identity

App Registration

App Registration

Global Schema (User)

App Roles

Global Role

Figure 8-2 Assignments and Mappings Between Azure AD and Oracle Database

These mappings are as follows:

- An Azure AD user can be mapped directly to an Oracle Database global schema (user).
- An Azure AD user, Azure AD group, or application is assigned to an app role, which is then mapped to either an Oracle Database global schema (user) or a global role.

8.1.4 Use Cases for Connecting to an Oracle Database Using Azure AD

Oracle Database supports several use cases for connecting to the database.

- OAuth2 authorization code flow: This is the most common flow for human users. The client directs the Azure AD user to Azure AD to get the authorization code. This code is used to get the database access token. See the Microsoft Azure article Microsoft identity platform and OAuth 2.0 authorization code flow.
- Resource owner password credentials (ROPC): This flow is not recommended for
 production servers. It is useful for test software that cannot work with a pop-up
 authentication window. It is used in non-graphic user interface environments when a popup window cannot be used to authenticate a user.
- Client credentials: This flow is used for applications to connect with the database. The
 application must register with Azure AD app registration and needs a client ID and client
 password. These client credentials must be used to get the database access token from
 Azure AD when the application connects to the database. The application can pass the
 token through the file system or through the database client API.
- On-behalf-of (OBO) token: An Azure application requests an OBO token for a logged in
 user. The OBO token will also be an access token for the database with the Azure AD user
 identity and assigned app roles for the database. This enables the Azure AD user to log in
 to the database as the user and not the application. Only an application can request an
 OBO token for its Azure AD user and pass it to the database client through the API.



8.1.5 General Process of Authenticating Microsoft Azure AD Identities with Oracle Database

The Oracle Database administrator and the Microsoft Azure AD administrator play roles to allow Azure AD users to connect to the database using Azure AD OAuth2 access tokens.

The general process is as follows:

- The Oracle Database administrator ensures that the Oracle Database environment meets
 the requirements for the Microsoft Azure AD integration. See Oracle Database
 Requirements for the Microsoft Azure AD Integration.
- The Azure AD administrator creates an Azure AD app registration for the database and the Oracle Database administrator enables the database to use Azure AD tokens for database access.
 - As part of the app registration process, the Azure AD administrator creates Azure app roles to be used for the mappings between the Azure users, groups, and applications to the Oracle Database schemas and roles.
- 3. The Oracle Database administrator creates and maps global schemas to either an Azure AD user (exclusive schema mapping) or to an Azure app role (shared schema mapping). The Azure AD user or application must be mapped to one schema.
- **4.** Optionally, the Oracle administrator creates and maps global Oracle Database roles to Azure app roles.
- The Azure AD end user who wants to connect with the Oracle Database instance registers the client application as an Azure AD client (similar to how the Oracle database is registered).
 - The Azure AD client will have a client identification and a client secret, unless the application client is public. If the application client is public, then only the application client identification is necessary.
- 6. The Azure AD end user (who can be a database administrator) connects using an utility such as PowerShell or the Azure command-line interface to retrieve the <code>OAuth2</code> database access token and store it in a local file directory. An application can also request an Azure AD <code>OAuth2</code> access token directly from Azure AD and pass it through a database client API. Refer to the following Oracle Database client documentation for information about passing Azure AD <code>OAuth2</code> tokens:
 - JDBC-thin clients: Oracle Database JDBC Developer's Guide
 - · Oracle Call Interface (OCI): Oracle Call Interface Programmer's Guide
 - Oracle Data Provider for .NET (ODP): Oracle Data Provider for .NET Developer's Guide for Microsoft WindowsConnecting to Oracle Database
- Once connected to the Oracle Database instance, the Azure AD end user performs database operations as needed.

8.2 Configuring the Oracle Database for Microsoft Azure AD Integration

The Microsoft Azure AD integration with the Oracle Database instance requires the database to be registered with Azure AD.



- Oracle Database Requirements for the Microsoft Azure AD Integration
 Before you can configure an Oracle Database instance with Microsoft Azure AD, you must ensure that your environment meets special requirements.
- Registering the Oracle Database Instance with a Microsoft Azure AD Tenancy
 A user with Azure AD administrator privileges uses Microsoft Azure AD to register the
 Oracle Database instance with the Microsoft Azure AD tenancy.
- Enabling Microsoft Azure AD v2 Access Tokens
 Oracle Database supports integration with the v1 and v2 Azure AD OAuth2 access token.
- Managing App Roles in Microsoft Azure AD
 In Azure AD, you can create and manage app roles that will be assigned to Azure AD users and groups and also be mapped to Oracle Database global schemas and roles.
- Enabling Azure AD External Authentication for Oracle Database
 You need to enable Microsoft Azure AD external authentication with Oracle Database.
- Disabling Azure AD External Authentication for Oracle Database
 To disable Azure AD External authentication for an Oracle Database instance, you must use the ALTER SYSTEM statement.

8.2.1 Oracle Database Requirements for the Microsoft Azure AD Integration

Before you can configure an Oracle Database instance with Microsoft Azure AD, you must ensure that your environment meets special requirements.

For an on-premises, non-cloud Oracle database, follow the steps in this document. If your Oracle database is in one of the following DBaaS platforms, then refer to the platform documentation for additional requirements.

- Using Oracle Autonomous Database Serverless
- Using Oracle Autonomous Database on Dedicated Exadata Infrastructure
- Use Azure Active Directory Authentication with Base Database Service
- Use Azure Active Directory Authentication with Exadata Database on Dedicated Infrastructure

Note the following:

- The Oracle Database server must be able to request the Azure AD public key. Depending on the enterprise network connectivity setup, you may need to configure a proxy setting.
- Users and applications that need to request an Azure AD token must also be able to have network connectivity to Azure AD. You may need to configure a proxy setting for the connection.
- You must configure Transport Layer Security (TLS) between the Oracle Database client and the Oracle Database server so that the token can be transported securely. This TLS connection can be either one-way or mutual.
- You can create the TLS server certificate to be self-signed or be signed by a well known certificate authority. The advantage of using a certificate that is signed by a well known Certificate Authority (CA) is that the database client can use the system default certificate store to validate the Oracle Database server certificate instead of having to create and maintain a local wallet with the root certificate. Note that this applies to Linux and Windows clients only.



Related Topics

Transport Layer Security Connection without a Client Wallet
 A Transport Layer Security (TLS) connection that uses a common root certificate for the
 database server does not require a client wallet.

8.2.2 Registering the Oracle Database Instance with a Microsoft Azure AD Tenancy

A user with Azure AD administrator privileges uses Microsoft Azure AD to register the Oracle Database instance with the Microsoft Azure AD tenancy.

- 1. Log in to the Azure portal as an administrator who has Microsoft Azure AD privileges to register applications.
- In the Azure Active directory admin center page, from the left navigation bar, select Azure Active Directory.
- 3. In the MS App registrations page, select **App registrations** from the left navigation bar.
- 4. Select New registration.

Register an application

The Register an application window appears.

* Name
The user-facing display name for this application (this can be changed later).

ExampleOatabase

**Supported account types

Who can use this application or access this API?

Accounts in this organizational directory only (az207oracle only - Single tenant)

Accounts in any organizational directory (Any Azure AD directory - Multitenant)

Accounts in any organizational directory (Any Azure AD directory - Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)

Personal Microsoft accounts only

Help me choose...

Redirect URI (optional)

Well return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

Select a platform

| e.g. https://example.com/auth

| Register an app you're working on here. Integrate gallery apps and other apps from outside your organization by adding from Enterprise applications.

- 5. In the Register an application page, enter the following Oracle Database instance registration information:
 - In the **Name** field, enter a name for the Oracle Database instance connection (for example, *Example Database*).
 - Under Supported account types, select the account type that matches your use case.



- Accounts in this organizational directory only (tenant_name only Single tenant)
- Accounts in any organizational directory (Any Azure AD directory -Multitenant)
- Accounts in any organizational directory (Any Azure AD directory -Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
- Personal Microsoft accounts only
- 6. Bypass the Redirect URI (Optional) settings. You do not need to create a redirect URI because Azure AD does not need one for the database server.
- 7. Click Register.

After you click **Register**, Azure AD displays the app registration's Overview pane, which will show the Application (client) ID under Essentials. This value is a unique identifier for the application in the Microsoft identity platform. Note the term Application refers to the Oracle Database instance.

8. Register a scope for the database app registration.

A scope is a permission to access the database. Each database will need a scope so that clients can establish a trust with the database by requesting permission to use the database scope. This allows the database client to get access tokens for the database.

- a. In the left navigation bar, select Expose an API.
- b. Under Set the App ID URI, in the **Application ID URI** field, enter the app ID URI for the database connection using the following format, and then click **Save**:

```
your tenancy url/application (client) id
```

In this specification:

- your_tenancy_url must include https as the prefix and the fully qualified domain name of your Azure AD tenancy.
- application_(client)_id is the ID that was generated when you registered the
 Oracle Database instance with Azure AD. It is displayed in the Overview pane of
 the app registration.

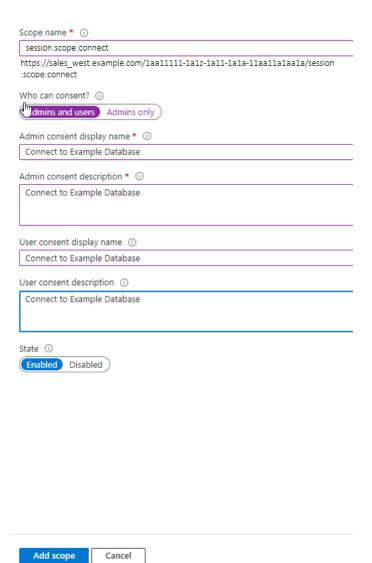
For example:

```
https://sales west.example.com/laa11111-1a1z-1a11-1a1a-11aa11a1aa1a
```

c. Select **Add a scope** and then enter the following settings:



Add a scope



Scope name specifies a name for the scope. Enter the following name:

session:scope:connect

This name can be any text. However, a scope name must be provided. You will need to use this scope name later when you give consent to the database client application to access the database.

- Who can consent specifies the necessary permissions. Select Admins and users, or for higher restrictions, Admins only.
- Admin consent display name describes the scope's purpose (for example, Connect to Oracle), which only administrators can see.
- Admin consent display name describes the scope's purpose (for example, Connect to Example Database), which only administrators can see.



- User consent display name is a short description of the purpose of the scope (for example, Connect to Example Database), which users can see if you specify Admins and users in Who can consent.
- User consent description is a more detailed description of the purpose of the scope (for example, Connect to Example Database), which users can see if you specify Admins and users in Who can consent.
- State enables or disables the connection. Select Enabled.

After you complete these steps, you are ready to add one or more Azure app roles, and then perform the mappings of Oracle schemas and roles.

Related Topics

Quickstart: Register an application with the Microsoft identity platform

8.2.3 Enabling Microsoft Azure AD v2 Access Tokens

Oracle Database supports integration with the v1 and v2 Azure AD OAuth2 access token.

The Azure AD v2 access token, supports a wider range of access scenarios than the v1 token, including authentication for both organizational accounts (Azure AD) and personal Microsoft accounts (MSA). You can use this token with applications that are registered in the Azure portal using the **App registrations** experience.

When you use the Azure AD v2 <code>OAuth2</code> access token, the credential flow continues to work as it did before without any changes. However, the <code>upn:</code> claim must be added when you use v2 tokens with the interactive flow.

- 1. Check the version of the Azure AD access token that you are using.
- Log in to the Microsoft Azure portal.
- 3. Search for and select Azure Active Directory.
- 4. Under Manage, select App registrations.
- Choose the application for which you want to configure optional claims based on your scenario and desired outcome.
- 6. Under Manage, select Token configuration.
- 7. Click Add optional claim and select upn.

When you use v2 tokens, the aud: claim only reflects the APP ID value. You do not need to set the https:domain prefix to the APP ID URI when v2 tokens are being used. This simplifies the configuration for the database because the default APP ID URI can be used.

Related Topics

Checking the Azure AD Access Token Version
 You can check the version of the Microsoft Azure AD access token that your site uses by using the JSON Web Tokens web site.

8.2.4 Managing App Roles in Microsoft Azure AD

In Azure AD, you can create and manage app roles that will be assigned to Azure AD users and groups and also be mapped to Oracle Database global schemas and roles.



Creating a Microsoft Azure AD App Role

Azure AD users, groups, and applications that need to connect to the database will be assigned to the database app roles.

Assigning Users and Groups to the Microsoft Azure AD App Role
Before Microsoft Azure AD users can have access to the Oracle database, they must first
be assigned to the app roles that will be mapped to Oracle Database schema users or
roles.

Assigning an Application to an App Role

An application that must connect to the database using the client credential flow must to be assigned to an app role.

8.2.4.1 Creating a Microsoft Azure AD App Role

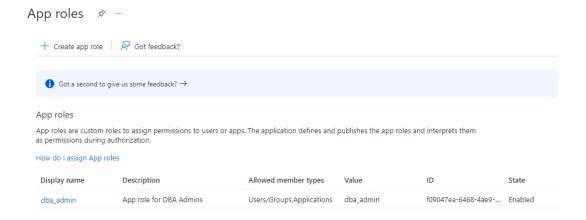
Azure AD users, groups, and applications that need to connect to the database will be assigned to the database app roles.

See the Microsoft Azure article Create and assign a custom role in Azure Active Directory for detailed steps on how to create an app role. The following steps describe how to create the app role for use with an Oracle database.

- 1. Log in to Azure AD as an administrator who has privileges for creating app roles.
- Access the Oracle Database app registration that you created.
 - **a.** Use the **Directory + subscription** filter to locate the Azure Active Directory tenant that contains the Oracle Database app registration.
 - b. Select Azure Active Directory.
 - c. Under Manage, select App registrations, and then select the Oracle Database instance that you registered earlier.
- Under Manage, select App roles.
- 4. In the App roles page, select Create app role.
- 5. In the Create app role page, enter the following information:
 - **Display name** is the displayed name of the role (for example, HR App Schema). You can include spaces in this name.
 - **Value** is the actual name of the role (for example, HR_APP). Ensure that this setting matches exactly the string that is referenced in the database mapping to a schema or role. Do not include spaces in this name.
 - Description provides a description of the purpose of this role.
 - Do you want to enable this app role? enables you to activate the role.
- 6. Click Apply.

The app role appears in the App roles pane.





8.2.4.2 Assigning Users and Groups to the Microsoft Azure AD App Role

Before Microsoft Azure AD users can have access to the Oracle database, they must first be assigned to the app roles that will be mapped to Oracle Database schema users or roles.

See the Microsoft Azure article Add app roles to your application and receive them in the token for detailed steps assigning users and groups to an app role. The following steps explain how to do this for an Oracle database.

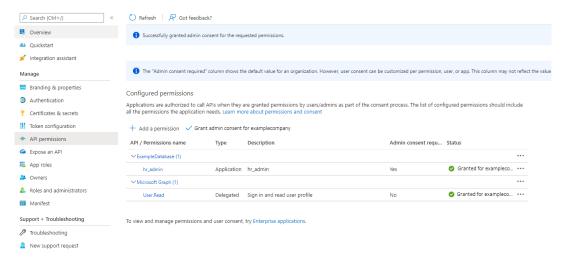
- 1. Log in to Azure AD as an administrator who has privileges for assigning Azure AD users and groups to app roles.
- 2. In enterprise applications, find the name of the Oracle Database app registration that you created. This is automatically created when you create an app registration.
 - **a.** Use the **Directory + subscription** filter to locate the Azure Active Directory tenant that contains the Oracle connection.
 - b. Select Azure Active Directory.
 - c. Under Manage, select Enterprise applications, and then select the Oracle Database app registration name that you registered earlier.
- Under Getting Started, select Assign users and groups.
- Select Add user/group.
- 5. In the Add assignment window, select **Users and groups** to display a list of users and security groups.
- From this list, select the users and groups that you want to assign to the app role, and then click Select.
- 7. In the Add assignment window, select **Select a role** to display a list of the app roles that you have created.
- 8. Select the app role and then select **Select**.
- 9. Click Assign.

8.2.4.3 Assigning an Application to an App Role

An application that must connect to the database using the client credential flow must to be assigned to an app role.



- 1. Log in to Azure AD as an administrator who has privileges for assigning Azure AD users and groups to app roles.
- 2. Access the app registration for the application.
- Under Manage, select API permissions.
- In the Configured permissions area, select + Add a permission.
- 5. In the Request API permission pane, select the My APIs tab.
- **6.** Select the Oracle Database app that you want to give permission for this application to access. Then select the **Application permissions** option.
- 7. Select the database app roles to assign to the application and then click the Add Permission box at the bottom of the screen to assign the app roles and close the dialog box. Ensure that the app roles that you just assigned appear under Configured permissions.



8. Select **Grant admin consent for** *tenancy* to grant consent for the tenancy users, then select **Yes** in the confirmation dialog box.

Related Topics

Configure the admin consent workflow

8.2.5 Enabling Azure AD External Authentication for Oracle Database

You need to enable Microsoft Azure AD external authentication with Oracle Database.

For additional information about Azure AD authentication for your platform, see the documentation links below.

- 1. Log in to the Oracle Database instance as a user who has been granted the ALTER SYSTEM system privilege.
- 2. Set the IDENTITY PROVIDER TYPE parameter as follows:

```
ALTER SYSTEM SET IDENTITY PROVIDER TYPE=AZURE AD SCOPE=BOTH;
```

3. Ensure that you set the IDENTITY PROVIDER TYPE parameter correctly.

```
SELECT NAME, VALUE FROM V$PARAMETER WHERE NAME='identity provider type';
```



The following output should appear:

```
NAME VALUE
----identity_provider_type AZURE_AD
```

4. Set the IDENTITY PROVIDER CONFIG parameter by using the following syntax:

For example:

```
ALTER SYSTEM SET IDENTITY_PROVIDER_CONFIG =
'{
    "application_id_uri" : "https://www.example.com/11aa1a11-
aaaa-1111-1111-1111aa11111",
    "tenant_id" : "111a1111-a11a-111a-1a1a-11111111111",
    "app_id" : "11aa1a11-aaaa-1111-1111-1111aa11111"
}' SCOPE=BOTH;
```

See the following platform-specific documentation for information about enabling Oracle Database for Azure AD external authentication, in addition to the information detailed in this document for on-premises (non-cloud) Oracle databases.

- Using Oracle Autonomous Database Serverless
- Oracle Autonomous Database on Dedicated Exadata Infrastructure

8.2.6 Disabling Azure AD External Authentication for Oracle Database

To disable Azure AD External authentication for an Oracle Database instance, you must use the ALTER SYSTEM statement.

In addition to Oracle Database, this procedure can be used for Oracle Autonomous Database on Dedicated Exadata Infrastructure and Oracle Exadata Cloud Service (Oracle ExaCS). If you want to disable Azure AD external authentication with these products, see their product documentation.

To disable Azure AD from Oracle Autonomous Database Serverless, see *Using Oracle Autonomous Database Serverless*. The following procedure applies to all other platforms:

- 1. Log in to the Oracle Database instance as a user who has been granted the ALTER SYSTEM system privilege.
- 2. Set the identity provider parameters as follows:

```
ALTER SYSTEM RESET IDENTITY_PROVIDER_CONFIG SCOPE=BOTH; ALTER SYSTEM RESET IDENTITY_PROVIDER_TYPE SCOPE=BOTH;
```



8.3 Mapping Oracle Database Schemas and Roles

Azure AD users will be mapped to one database schema and optionally to one or more database roles.

- Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User
 You can exclusively map an Oracle Database schema to a Microsoft Azure AD user.
- Mapping a Shared Oracle Schema to an App Role
 In this mapping, an Oracle schema is mapped to an app role. Therefore, anyone who has that app role would get the same shared schema.
- Mapping an Oracle Database Global Role to an App Role
 Oracle Database global roles that are mapped to Azure app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

8.3.1 Exclusively Mapping an Oracle Database Schema to a Microsoft Azure AD User

You can exclusively map an Oracle Database schema to a Microsoft Azure AD user.

- 1. Log in to the Oracle Database instance as a user who has been granted the CREATE USER or ALTER USER system privilege.
- 2. Run the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the Azure AD user name.

For example, to create a new database schema user named peter_fitch and map this user to an existing Azure AD user named peter.fitch@example.com:

```
CREATE USER peter_fitch IDENTIFIED GLOBALLY AS 'AZURE USER=peter.fitch@example.com';
```

3. Grant the CREATE SESSION privilege to the user.

```
GRANT CREATE SESSION TO peter_fitch;
```

8.3.2 Mapping a Shared Oracle Schema to an App Role

In this mapping, an Oracle schema is mapped to an app role. Therefore, anyone who has that app role would get the same shared schema.

- Log in to the Oracle Database instance as a user who has the CREATE USER or ALTER USER system privilege.
- 2. Run the CREATE USER or ALTER USER statement with the IDENTIFIED GLOBALLY AS clause specifying the Azure application role name.

For example, to create a new database global user account (schema) named <code>dba_azure</code> and map it to an existing Azure AD application role named <code>AZURE DBA</code>:

```
CREATE USER dba_azure IDENTIFIED GLOBALLY AS 'AZURE_ROLE=AZURE_DBA';
```



8.3.3 Mapping an Oracle Database Global Role to an App Role

Oracle Database global roles that are mapped to Azure app roles give Azure users and applications additional privileges and roles above those that they have been granted through their login schemas.

- Log in to the Oracle Database instance as a user who has been granted the CREATE ROLE or ALTER ROLE system privilege
- 2. Run the CREATE ROLE or ALTER ROLE statement with the IDENTIFIED GLOBALLY AS clause specifying the name of the Azure AD application role.

For example, to create a new database global role named widget_sales_role and map it to an existing Azure AD application role named WidgetManagerGroup:

```
CREATE ROLE widget_sales_role IDENTIFIED GLOBALLY AS 'AZURE ROLE=WidgetManagerGroup';
```

8.4 Configuring Azure AD Client Connections to the Oracle Database

You can configure client connections to connect with the Azure AD registered database

- About Configuring Client Connections to Azure ADs
 There are numerous ways that you can configure a client to connect with an Oracle Database instance using Azure AD tokens.
- Supported Client Drivers for Azure AD Connections
 Oracle Database supports several types of client drivers for Azure AD connections.
- Using Centralized Oracle Cloud Infrastructure Services for Net Naming and Secrets
 You can use the Oracle Cloud Infrastructure (OCI) object store and vault to centrally store
 net names and secrets.
- Operational Flow for SQL*Plus Client Connection in PowerShell to Oracle Database
 The connection between the Azure user, Azure AD, and the Oracle database relies on the
 passing of the OAuth2 token throughout these components.
- Registering a Client with Azure AD Application Registration
 This type of registration is similar to registering Oracle Database with Azure AD appregistration.
- Examples of Retrieving Azure AD OAuth2 Tokens
 These examples show different ways that you can retrieve Azure AD OAuth2 tokens.
- Configuring SQL*Plus for Azure AD Access Tokens
 You must configure SQL*Plus to retrieve the Azure AD database access token from a location and use it when the / slash login is used.
- Creating a Network Proxy for the Database to Connect with the Internet
 This network proxy will enable the Oracle database to reach the Azure AD endpoint.
- Enabling Clients to Directly Retrieve Azure Tokens
 You can set parameters to enable clients to directly retrieve Azure tokens on their own.



8.4.1 About Configuring Client Connections to Azure ADs

There are numerous ways that you can configure a client to connect with an Oracle Database instance using Azure AD tokens.

You should choose the client connection method that works best with your environment. This guide provides examples of connecting SQL*Plus with different methods of getting an Azure AD OAuth2 access token. All Oracle Database release 19c clients can accept a token that is passed as a file. The JDBC-thin, Instant Client, and ODP.net drivers also accept the token through the database client API from an application. Oracle Database tools such as SQL*Plus cannot retrieve the tokens directly, so tools such as PowerShell or Azure CLI must be used to retrieve the Azure AD OAuth2 access token. To retrieve an Azure AD token, the client must be registered through the Azure AD app registration process. Registering the client is similar to registering the Oracle Database server with Azure AD using app registration. Both the database and client must be registered with Azure AD.

The database must be registered so the client can get permission to get an access token for the database. The client must be registered so that Azure AD can recognize a trusted client is asking for an access token.

See the following Microsoft Azure articles for more information about connecting clients to Azure AD:

- Quickstart: Configure a client application to access a web API
- Choose the right Azure command-line tool
- Get Azure AD tokens by using the Microsoft Authentication Library
- Install the Azure CLI on Linux

8.4.2 Supported Client Drivers for Azure AD Connections

Oracle Database supports several types of client drivers for Azure AD connections.

- JDBC-thin: Oracle Database 19.16 (July 2022), Oracle Database 21.8 (October 2022)
- OCI (C driver): Oracle Database 19.16 (July 2022)
- Oracle Instant Client based on OCI
- Oracle Data Provider (core): Oracle Database 19.16, Oracle Database 21.7
- Oracle Data Provider (unmanaged): based on OCI
- Oracle Data Provider (managed): Oracle Database 19.16, Oracle Database 21.7
- All other drivers built on OCI adopts the OCI compatibility

8.4.3 Using Centralized Oracle Cloud Infrastructure Services for Net Naming and Secrets

You can use the Oracle Cloud Infrastructure (OCI) object store and vault to centrally store net names and secrets.

This functionality is currently supported with the JDBC-thin and .NET-thin drivers.

See the following guides:

Oracle Database Net Services Administrator's Guide



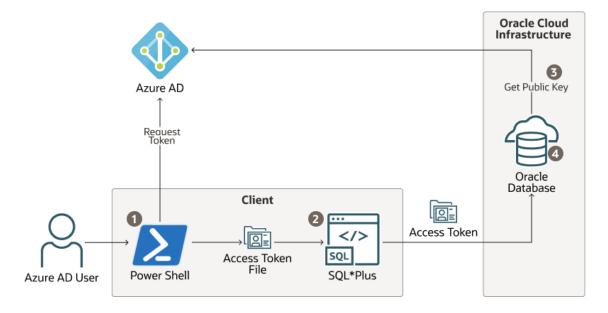
Oracle Database Net Services Reference

8.4.4 Operational Flow for SQL*Plus Client Connection in PowerShell to Oracle Database

The connection between the Azure user, Azure AD, and the Oracle database relies on the passing of the <code>OAuth2</code> token throughout these components.

This example shows the use of the Resource Owner Password Credential (ROPC) flow with a public client. See the Microsoft Azure article Microsoft identity platform and OAuth 2.0 Resource Owner Password Credentials for detailed information about ROPC.

Figure 8-3 ROPC Operational Flow with a Public Client



- 1. The Azure user requests an Azure AD access token for the database in PowerShell and the returned token is written into a file called token at a file location.
- 2. The Azure user connects to the database using / slash login. Either the sqlnet.ora or tnsnames.ora connection string tells the instant client that an Azure AD OAuth2 token is needed and to retrieve it from a specified file location. The access token is sent to the database.
- 3. The database verifies that the access token came from Azure AD (using the Azure AD public key) and then checks the token for additional claims.
- 4. The database finds the schema mapping (exclusive or shared) and creates the session. The database will also grant any global roles that the Azure user is also assigned to through an app role.

8.4.5 Registering a Client with Azure AD Application Registration

This type of registration is similar to registering Oracle Database with Azure AD appregistration.



- Confidential and Public Client Registration
 - You can register the database client with Azure as either confidential or public depending on your use case.
- Registering a Database Client App with Azure AD
 Creating the client app registration is similar to creating the Oracle Database instance with the Microsoft Azure AD tenancy.

8.4.5.1 Confidential and Public Client Registration

You can register the database client with Azure as either confidential or public depending on your use case.

See the Microsoft Azure article Authentication flows and application scenarios for detailed information about authentication flows and application scenarios.

Registering a confidential client app requires that the client have a secret, in addition to the client ID. The confidential client app uses both the client ID and the secret when it makes Azure AD requests. However, in an enterprise, it is not practical for every SQL*Plus and SQLcl user to create a separate app registration with its own secret. In addition, a secret is no longer a secret when you start to share it within an organization. It is far better to just create a public client app. A public client app does not have a secret; it only has a client ID. All database tool users can use the public client ID when they connect to Azure AD to get an access token. The Azure AD user still needs to authenticate to Azure AD with their own user credential.

8.4.5.2 Registering a Database Client App with Azure AD

Creating the client app registration is similar to creating the Oracle Database instance with the Microsoft Azure AD tenancy.

- 1. Log in to the Azure portal as an administrator who has Microsoft Azure AD privileges to register applications.
- 2. In the Azure Active directory admin center page, from the left navigation bar, select **Azure Active Directory**.
- 3. In the MS App registrations page, select **App registrations** from the left navigation bar.
- Select New registration.
- 5. In the Register an application page, enter the following Oracle Database instance registration information:
 - In the Name field, enter a name for the client app (for example, DatabaseClientApplication)..
 - Under Supported account types, select the account type that matches your use case.
 - Accounts in this organizational directory only (tenant_name only Single tenant)
 - Accounts in any organizational directory (Any Azure AD directory -Multitenant)
 - Accounts in any organizational directory (Any Azure AD directory -Multitenant) and personal Microsoft accounts (e.g. Skype, Xbox)
 - Personal Microsoft accounts only
- 6. Under Redirect URI (optional), configure the redirect URI for the client app.



Redirect URI (optional)

We'll return the authentication response to this URI after successfully authenticating the user. Providing this now is optional and it can be changed later, but a value is required for most authentication scenarios.

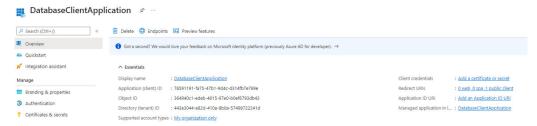


- Select Public client/native (mobile & desktop), Web, or Single-page application (SPA). Choose Public client if this client app will be used by multiple users such as database administrators who need to use SQL*Plus to access the Oracle Database instance.
- Add a redirect URI of http://localhost, unless you have another address to use.
 This redirect URI is needed for the authorization flow.

Click Register.

At this stage, the database client has been registered with Azure AD. Next, you must add the new client to the list of authorized client apps for the Oracle Database instance.

- 8. To add the new client to this list of client apps, do the following:
 - **a.** Make a note of the new client's Application (client) ID. This ID is in the Overview page for the app.



- b. On the App registrations page, open the app registration page for the database server by selecting it from the menu.
- c. On the left side, select Expose an API.
- d. Scroll down on the main page until you see Authorized client applications.
- e. Select + to add a client application.
- f. Copy the new client's Application (client) ID to the Client Id field.



g. Click Add application.

Related Topics

Quickstart: Register an application with the Microsoft identity platform



8.4.6 Examples of Retrieving Azure AD OAuth2 Tokens

These examples show different ways that you can retrieve Azure AD OAuth2 tokens.

- Example: Using PowerShell to Get a Token Using Resource Owner Password Credentials
 This example shows how to use PowerShell to get an Azure AD access token by using
 Resource Owner Password Credentials (ROPC).
- Example: Using Python with Microsoft Authentication Library Using an Authorization Flow Because this example with the Microsoft Authentication Library (MSAL) is in Python, it can be run on a variety of platforms such as PowerShell and Linux.
- Example: Using Curl with a Resource Owner Password Credential Flow
 This example shows how to use the curl command against the Azure AD API using a
 Resource Owner Password Credential (ROPC) flow with a public Azure AD client.
- Example: Azure CLI Using Authorization Flow
 This example shows how to use the Azure CLI to retrieve an access token and then write the token to a file.

8.4.6.1 Example: Using PowerShell to Get a Token Using Resource Owner Password Credentials

This example shows how to use PowerShell to get an Azure AD access token by using Resource Owner Password Credentials (ROPC).

You can retrieve the <code>OAuth2</code> access token by making a REST call from PowerShell. This configuration requires several values that were generated or that you specified when you registered the Oracle Database instance with Azure AD.

- 1. If necessary install the Azure Active Directory PowerShell module.
 - Follow the instructions in the Microsoft article Install the Azure Az PowerShell module to download and install Azure PowerShell. It takes about 20 minutes or longer to perform the installation. You may want to set debug options for Azure PowerShell so that you can see how the installation is progressing.
- After the Azure PowerShell installation is complete, log in to PowerShell and then set the following variables in the order shown.
 - a. \$TenantDomain = "user tenancy domain name"

This value is the tenancy domain name. For example:

```
$TenantDomain = "example.com"
```

b. \$AppClientId ="application client id"

This value sets the application client ID for the database client, not the database server. This is the **Application (client) ID** value in the app registration's Overview pane. For example:

```
$AppClientId ="111a1a1a-aa1a-1a1a-11aa-1a11111111aa"
```

c. \$Username = "user name", which



This value is the name of the Azure user who wants to access the Oracle Database instance. For example:

```
$Username = "peter.fitch@example.com"
```

- d. Entering the user password in PowerShell scripts depends on your corporate or personal security standards. Use your own method to capture the password securely or use this example that hides the password from the command history and command line window. You should delete the password variables after they have been used. Enter these in the order shown:
 - i. \$securePassword = Read-Host " Enter Password" -AsSecureString
 - ii. \$Password =
 [System.Runtime.InteropServices.Marshal]::PtrToStringAuto([System.Runtime.InteropServices.Marshal]::SecureStringToBSTR(\$securePassword))
- e. \$Scope = "database app id uri/scope"

This value sets the app ID URI for the database and the scope (permission) for the database, separated by a / slash. These values can be found in the database app registration Expose an API page. In the following example, https://example.com/111aa1aa-1111-1111-a1a1-1a11a111111a is the app ID URI and session:scope:connect is the scope.

```
$Scope = "https://example.com/111aa1aa-1111-1111-a1a1-1a11a111111a/
session:scope:connect"
```

This is the request body of the upcoming REST call.

g. \$OAuthResponse = Invoke-RestMethod -Method Post -Uri https://
login.microsoftonline.com/\$TenantDomain/oauth2/v2.0/token Body \$requestBody

This gets the OAuth2 access token for the user.

- h. You can remove the password from the variables if you do not need them.
 - \$securePassword = \$null, for secure password strings
 - \$Password = \$null, for clear text password strings
- i. \$AccessToken = \$OAuthResponse.access_token | Out-File -FilePath .\token -Encoding ASCII, which writes the OAuth2 token to the current file location using ASCII encoding
- 3. Optionally, because the Azure AD <code>OAuth2</code> access token is a JSON Web Token (JWT) formatted token, you can view the cleartext of the encoded content by copying and pasting the token content into the website:

https://jwt.io/

Note the following:

- The default PowerShell UTF16 file encoding cannot be used for the token. Use ASCII encoding as an alternative.
- Tokens may not work cross-platform (for example, Windows to Linux or Linux to Windows), depending on encoding changes to the file when it is moved.

At this stage, the <code>OAuth2</code> access token has been retrieved and stored as a file. The next step is to enable the SQL*Plus client to use the store access token and send it to the database.

8.4.6.2 Example: Using Python with Microsoft Authentication Library Using an Authorization Flow

Because this example with the Microsoft Authentication Library (MSAL) is in Python, it can be run on a variety of platforms such as PowerShell and Linux.

When multi-factor authentication is enabled for the user, an <code>OAuth2</code> authorization flow is necessary for a user to add the second authentication. Because the authorization flow requires two round trips to Azure AD, it is best handled using the MSAL. See the Microsoft article <code>Get Azure AD</code> tokens by using the Microsoft Authentication Library for how to use a python script with MSAL. These instructions are for the Databricks service, but the scope is changed to the database App ID URI and scope instead of the Databricks scope.

- Bypass the steps to set up the client app registration, since you have already accomplished that step except make sure you add a Redirect URI (http://localhost) for your client app registration.
- 2. Go directly to Get Azure AD tokens by using the MSAL Python library.

You will need the Directory (tenant) ID, Client ID for the public app client, and the database App ID URI and scope. You will see a code section for **scopes** with directions to not modify this variable. Because this python code was written for Databricks scope, you will need to change this scope variable to the scope of your database. For example:

```
scopes = ['https://example.com/1111aa1a-a1aa-1a11-11aa-1a1a11aa1111/
session:connect']
```

3. Modify the code to write the token to a file location.

Use the following example code and append it to the print statements at the end. Note the extra lines to back up and restore the original stdout.

```
stdout_backup = sys.stdout
with open('token', 'w') as token_file:
    sys.stdout = token_file
    print(acquire_tokens_result['access_token'])
sys.stdout = stdout backup
```

8.4.6.3 Example: Using Curl with a Resource Owner Password Credential Flow

This example shows how to use the curl command against the Azure AD API using a Resource Owner Password Credential (ROPC) flow with a public Azure AD client.

The cleartext password is part of this command so this is not so much for end-users as it is for applications. This would need to be protected.

Enter the following curl command:

```
curl -X POST -H 'Content-Type: application/x-www-form-urlencoded' https://
login.microsoftonline.com/az207oracleoutlook.onmicrosoft.com/oauth2/v2.0/
token
-d 'client_id=571c3f0a-aa3c-4f0a-93ed-4f75748955ea' -d 'scope=https://
example.com/383fe7ee-1433-4844-a2d5-5b80d811256d/session:scope:connect'
```

```
-d 'username=peter.fitch@example.com' -d 'password=password' -d
'grant type=password'
```

The response is a JSON file with token type, scope, expiration, and then the actual token. This file will need to be parsed so only the access token is written and stored in a file.

8.4.6.4 Example: Azure CLI Using Authorization Flow

This example shows how to use the Azure CLI to retrieve an access token and then write the token to a file.

See the Microsoft Azure article Install the Azure CLI on Linux for information about installing the Azure CLI.

1. Log in to your Azure tenancy.

```
$ az login
```

2. Get an access token and assign it to the token variable using the following syntax:

```
token=$(az account get-access-token --resource=database_app_id_uri --query
accessToken --output tsv)
```

For example:

```
token=$(az account get-access-token --resource=https://example.com/1111aa1a-1a1a-1a1a11aa1111 --query accessToken --output tsv)
```

If you get an error saying that the Azure CLI client app ID does not have permission to access the database resource, then copy the Azure CLI client app ID from the error message and add it to the list of authorized client applications for the database resource. (Go to the database app registration in Azure AD, click **Expose an API** and then **Add a client application**).

3. Write the token to a file.

```
$ echo "$token" >> token
```

8.4.7 Configuring SQL*Plus for Azure AD Access Tokens

You must configure SQL*Plus to retrieve the Azure AD database access token from a location and use it when the / slash login is used.

Only the latest SQL*Plus and Instant Client will work with Azure AD <code>OAuth2</code> tokens. There is no default location for the Azure AD token, so you must specify this location.

- Ensure that you have an Azure AD user account.
- Check with an Azure AD administrator or Oracle Database administrator for one of the following:
 - An application client ID that you can use to get Azure AD tokens. If you have Azure AD
 privileges to do so, then create your own client app registration, similar to registering
 the Oracle Database instance with an Azure AD tenancy.
 - You are mapped to a global schema in the database.



Ensure that you are using the latest release updates for the Oracle Database client releases 19c.

This configuration only works with the Oracle Database client release 19c.

- **4.** Follow the existing process to download the wallet from the Oracle Database instance and then follow the directions for configuring it for use with SQL*Plus.
- 5. On the client, set the following parameters in the sqlnet.ora file:
 - Check for the parameter SSL_SERVER_DN_MATCH = ON to ensure that DN matching is enabled.
 - Set the TOKEN_AUTH parameter to enable the client to use the Azure AD token. Include the TOKEN LOCATION parameter to point to the token location. For example:

```
TOKEN_AUTH=OAUTH
TOKEN LOCATION="token location"
```

Note that there is no default location. If the token is named token, then you only need to specify the file directory (for example, /test/oracle/aad-token). If the token name is different from token (for example, azure.token), then you must include this name in the path (for example, /test/oracle/aad-token/azure.token).

You can specify the TOKEN_AUTH and TOKEN_LOCATION parameters in the thermal as in sqlnet.ora. The TOKEN_AUTH and TOKEN_LOCATION values in the thermal connect strings take precedence over the sqlnet.ora settings for that connection. For example:

After the connect string is updated with the <code>TOKEN_AUTH</code> and <code>TOKEN_LOCATION</code> parameters, the Azure user can log in to the Oracle Database instance by running the following command to start SQL*Plus. You can include the connect descriptor itself or use the name of the descriptor from the <code>tnsnames.ora</code> file.

```
connect /@exampledb high
```

Or the user can use the connect string. For example:



```
L=Redwood City, ST=California, C=US") (TOKEN_AUTH=OAUTH) (TOKEN LOCATION="/test/oracle/aad-token")
```

The database client is already configured to get an Azure <code>OAuth2</code> token because <code>TOKEN_AUTH</code> has already been set, either through the <code>sqlnet.ora</code> file or in a connect string. The database client gets the <code>OAuth2</code> token and then sends the token to the Oracle Database instance.

Related Topics

Registering the Oracle Database Instance with a Microsoft Azure AD Tenancy
 A user with Azure AD administrator privileges uses Microsoft Azure AD to register the
 Oracle Database instance with the Microsoft Azure AD tenancy.

8.4.8 Creating a Network Proxy for the Database to Connect with the Internet

This network proxy will enable the Oracle database to reach the Azure AD endpoint.

- About Creating a Network Proxy for the Database to Connect with the Internet
 The Oracle database must connect to Azure AD endpoints and it may require network
 configuration and default trust store access.
- Testing the Accessibility of the Azure Endpoint
 You must ensure that your Oracle Database instance can access the Azure AD endpoint.
- Creating the Network Proxy for the Default Oracle Database Environment
 To create the network proxy, you must set environment variables and then restart the listener.
- Creating the Network Proxy for an Oracle Real Application Clusters Environment
 To create the network proxy, you must set an environment variable and then restart the
 database.
- Creating the Network Proxy in the Windows Registry Editor
 To create the network proxy in a Windows environment, you must update the Registry Editor (regedit).

8.4.8.1 About Creating a Network Proxy for the Database to Connect with the Internet

The Oracle database must connect to Azure AD endpoints and it may require network configuration and default trust store access.

You can configure the database when HTTP network proxy is in place in an enterprise, for a default Oracle Database environment and for an Oracle Real Applications Clusters environment. The database establishes a Transport Layer Security (TLS) link to Azure AD, so it also needs access to the default trust store on the database server. To enable this, ensure that the database server has access to the system default certificate store.

Related Topics

•

8.4.8.2 Testing the Accessibility of the Azure Endpoint

You must ensure that your Oracle Database instance can access the Azure AD endpoint.

For an Oracle database to accept Azure AD <code>OAuth2</code> tokens, the database must request the public key from the Azure AD endpoint.

 Run the following test to determine if the database can connect with the Azure AD endpoint:

```
SET SERVEROUTPUT ON SIZE 40000
DECLARE
  req UTL_HTTP.REQ;
  resp UTL_HTTP.RESP;
BEGIN
  UTL_HTTP.SET_WALLET(path => 'system:');
  req := UTL_HTTP.BEGIN_REQUEST('https://login.windows.net/common/
discovery/keys');
  resp := UTL_HTTP.GET_RESPONSE(req);
  DBMS_OUTPUT.PUT_LINE('HTTP response status code: ' || resp.status_code);
  UTL_HTTP.END_RESPONSE(resp);
END;
//
```

If this test is successful, then a PL/SQL procedure successfully completed message appears.

If the following messages appear, then it means that a database network access control list (ACL) policy blocked your test and you will need to temporarily set an access control list policy to allow you to test this:

```
ORA-29273: HTTP request failed ORA-24247: network access denied by access control list (ACL)
```

1. Set the ACL as follows:

Replace username_placeholder with the user name of the database user who is running the test. For example:

2. Try running the test again.

3. Remove the ACL, because you now no longer need it. For example, assuming your user name is dba debra:

If the database cannot connect with the Azure AD endpoint, even after you set the ACL policy, you will most likely need to set the HTTP_PROXY package for your database. Review the topics listed in Related Topics, depending if you are using a default Oracle Database environment or an Oracle Real Application Clusters RAC environment. Your network administrator should be able to tell you what the correct HTTP PROXY setting should be.

Related Topics

- Creating the Network Proxy for the Default Oracle Database Environment
 To create the network proxy, you must set environment variables and then restart the listener.
- Creating the Network Proxy for an Oracle Real Application Clusters Environment
 To create the network proxy, you must set an environment variable and then restart the
 database.

8.4.8.3 Creating the Network Proxy for the Default Oracle Database Environment

To create the network proxy, you must set environment variables and then restart the listener.

You do not need to restart the database.

1. In the server where the Oracle database is installed, set the http_proxy environment variable.

For example:

```
export http proxy=http://www-proxy-example.com:80/
```

2. Restart the listener.

```
lsnrctl stop
lsnrctl start
```

8.4.8.4 Creating the Network Proxy for an Oracle Real Application Clusters Environment

To create the network proxy, you must set an environment variable and then restart the database.

1. In the server where the Oracle database is installed, set the http_proxy environment variable.

Use this syntax to set the network proxies. the proxy command that you enter must have http:// preceding the proxy name and must have the port number at the end of the proxy:

```
http proxy=http://...:80/
```

For example:

```
srvctl setenv database -db db_name -env "http_proxy=http://www-
proxy.example.com:80/"
```

2. Stop the database.

```
$srvctl stop database -db db_name
```

3. Display the environment variable values to ensure that they are correctly set.

```
$ srvctl getenv database -db db name
```

Output similar to the following should appear:

```
db_name:
http_proxy=http://www-proxy.example.com:80/
https_proxy=http://www-proxy.example.com:80/
```

Restart the database.

```
$ srvctl start database -db db name
```

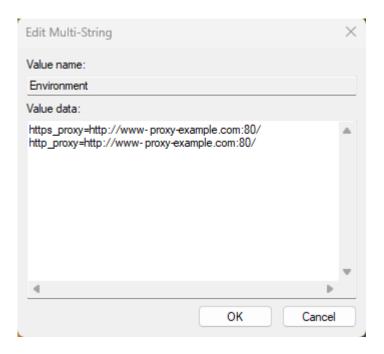
8.4.8.5 Creating the Network Proxy in the Windows Registry Editor

To create the network proxy in a Windows environment, you must update the Registry Editor (regedit).

- 1. Start the Registry Editor (regedit).
- Locate the
 \HKEY_LOCAL_MACHINE\SYSTEM\CurrentControlSet\Services\OracleServicerversion
 key.
- 3. Select this key, and then in the right panel, locate Environment.
- 4. Edit **Environment** to add a new multi-string value to it.

The following example uses the domain of example.com:





- 5. Click OK.
- Restart the database server.

For example:

```
net start Oracle_service_name
sqlplus "/as sysdba"
startup;
```

Open the PDBs.

ALTER PLUGGABLE DATABASE ALL OPEN;

8.4.9 Enabling Clients to Directly Retrieve Azure Tokens

You can set parameters to enable clients to directly retrieve Azure tokens on their own.

This feature is available in environments that use JDBC-thin clients, ODP.NET Core classes, or ODP.NET Managed Driver classes. It enables the client to display authentication requests to the user by using the following methods:

- If the user is using a Web application, then the authentication request appears in a dialog box prompting the user for their authentication.
- If the user is working in a command line shell, then the authentication request appears as a prompt.

To enable this feature for either of these authentication request types, you must set the following parameters in either the client's sqlnet.ora file or in a connect string. The connect string takes precedence over sqlnet.ora.



Table 8-1 Parameters to Directly Retrieve Tokens

Parameter	Description		
TOKEN AUTH	Sets the token authentication. Enter one of the following values:		
TOREN_AUTH	 AZURE_DEVICE_CODE signals the database driver to follow the device code flow for requesting an Azure AD access token. This is also for human users, when their environment cannot open a browser: a command line only environment. A device code and Azure AD login URL is written out to the standard output of the tool and the user logs into Azure AD on their cellphone or laptop, and then enters the device code. Users are authenticated through a separate channel and then allowed to continue access the database if the authentication is successful. 		
	 AZURE_INTERACTIVE tells the driver that it must flow the Azure OAuth2 interactive (OAuth2 authorization) flow to get an access token for the database. This configures the database client to get the token directly from Azure AD without having to use an external script. This is for human users who are logging into tools such as SQLcl and can also open a browser window in their environment to authenticate to Azure AD. 		
	 AZURE_MANAGED_IDENTITY enables the driver to authenticate as an identity that has been assigned to the host system. The host system must be a resource which is managed by Azure AD, such as a virtual machine. AZURE_SERVICE_PRINCIPAL enables the driver to authenticate using a secret or certificate of the registered application. 		
AZURE_CLIENT_ID	The unique application (client) ID assigned to your app by Azure AD when the app was registered. This app is your database client that will request to get an access token for the database for the user.		
AZURE_DB_APP_ID_URI	The application ID URI is a URI that uniquely identifies the application in your Azure AD. You get this value from the overview screen of your database Azure AD app registration.		
AZURE_TENANT_ID	Specifies the Azure tenancy ID of the database.		

8.5 Configuring Microsoft Azure AD Proxy Authentication

Proxy authentication allows an Azure AD user to proxy to a database schema for tasks such as application maintenance.

- About Configuring Microsoft Azure AD Proxy Authentication
 Azure users can connect to Oracle Autonomous Database by using proxy authentication.
- Configuring Proxy Authentication for the Azure AD User
 To configure proxy authentication for an Azure AD user, this user must already have a
 mapping to a global schema (exclusive or shared mapping). A separate database schema
 for the Azure AD user to proxy to must also be available.
- Validating the Azure AD User Proxy Authentication
 You can validate the Azure AD user proxy configuration for token authentication.

8.5.1 About Configuring Microsoft Azure AD Proxy Authentication

Azure users can connect to Oracle Autonomous Database by using proxy authentication.

Proxy authentication is typically used to authenticate the real user and then authorize them to use a database schema with the schema privileges and roles in order to manage an application. Alternatives such as sharing the application schema password are considered insecure and unable to audit which actual user performed an action.

A use case can be in an environment in which a named Azure AD user who is an application database administrator can authenticate by using their credentials and then proxy to a database schema user (for example, hrapp). This authentication enables the Azure AD administrator to use the hrapp privileges and roles as user hrapp in order to perform application maintenance, yet still use their Azure AD credentials for authentication. An application database administrator can sign in to the database and then proxy to an application schema to manage this schema.

8.5.2 Configuring Proxy Authentication for the Azure AD User

To configure proxy authentication for an Azure AD user, this user must already have a mapping to a global schema (exclusive or shared mapping). A separate database schema for the Azure AD user to proxy to must also be available.

After you ensure that you have this type of user, alter the database user to allow the Azure AD user to proxy to it.

- 1. Log in to the Autonomous Database instance as a user who has the ALTER USER system privileges.
- 2. Grant permission for the Azure AD user to proxy to the local database user account.

An Azure AD user cannot be referenced in the command so the proxy must be created between the database global user (mapped to the Azure AD user) and the target database user.

In the following example, hrapp is the database schema to proxy to, and peterfitch schema is the database global user exclusively mapped to user peterfitch.

ALTER USER hrapp GRANT CONNECT THROUGH peterfitch schema;

At this stage, the Azure AD user can log in to the database instance using the proxy. For example:

CONNECT [hrapp]/@connect string

8.5.3 Validating the Azure AD User Proxy Authentication

You can validate the Azure AD user proxy configuration for token authentication.

- 1. Log in to the Oracle Autonomous Database instance as a user who has the CREATE USER and ALTER USER system privileges.
- Connect as the Azure AD user and run the SHOW USER and SELECT SYS_CONTEXT commands.



For example, suppose you want to check the proxy authentication of the Azure AD user peterfitch when they proxy to database user hrapp:

```
CONNECT [hrapp]/@connect_string
SHOW USER;
--The output should be USER is "HRAPP"

SELECT SYS_CONTEXT('USERENV', 'AUTHENTICATION_METHOD') FROM DUAL;
--The output should be "TOKEN_GLOBAL"

SELECT SYS_CONTEXT('USERENV', 'PROXY_USER') FROM DUAL;
--The output should be "PETERFITCH_SCHEMA"

SELECT SYS_CONTEXT('USERENV', 'CURRENT_USER') FROM DUAL;
--The output should be "HRAPP"
```

8.6 Troubleshooting Microsoft Azure AD Connections

You can use trace files to diagnose problems with Microsoft Azure AD connections. You also can easily remedy ORA-12599 and ORA-03114 errors.

- Trace Files for Troubleshooting Oracle Database Client Connections with Azure AD
 You can use trace files to troubleshoot the Oracle Database integration with Microsoft
 Azure AD.
- ORA-12599 and ORA-03114 Errors Caused When Trying to Access a Database Using a Token

The ORA-12599: TNS: cryptographic checksum mismatch and ORA-03114: not connected to ORACLE errors indicate that the database to which you are trying to connect is protected by native network encryption.

Checking the Azure AD Access Token Version
 You can check the version of the Microsoft Azure AD access token that your site uses by
 using the JSON Web Tokens web site.

8.6.1 Trace Files for Troubleshooting Oracle Database Client Connections with Azure AD

You can use trace files to troubleshoot the Oracle Database integration with Microsoft Azure AD.

- About Trace Files Used for Troubleshooting Connections
 You can generate two levels of trace files to troubleshoot Microsoft Azure AD connections
 on client side.
- Setting Client Tracing for Token Authentication
 You can add EVENT settings to the client-side sqlnet.ora file to control client tracing.

8.6.1.1 About Trace Files Used for Troubleshooting Connections

You can generate two levels of trace files to troubleshoot Microsoft Azure AD connections on client side.

The two levels of trace files that you can generate are as follows:

Low level tracing prints traces in case of failures:

- If TCPS is not set up for the Azure AD connection, then it prints a message that the protocol has to be TCPS.
- If SSL_SERVER_DN_MATCH is not set to TRUE, then it prints a message that the value is FALSE.
- If TOKEN_LOCATION has not been specified, then it prints a message that the token location does not exist.
- If the token is not present at the specified TOKEN LOCATION, then it prints a message.
- If the application has passed in the token without setting OCI_ATTR_TOKEN_ISBEARER to
 true, it prints a message for the missing attribute.
- If the application has set OCI_ATTR_TOKEN_ISBEARER to TRUE and not passed in the token, it prints a message for the missing attribute.
- If the token has expired, then it prints a message.
- High level tracing prints traces in case of failure as mentioned above. In addition, it prints traces in case of success, as follows:
 - It prints where SSL_SERVER_DN_MATCH is present, tnsnames.ora or sqlnet.ora. It also prints the value as TRUE if set to TRUE.
 - If both the token and OCI_ATTR_TOKEN_ISBEARER=true are set by the application, then
 it prints a message.
 - If TOKEN AUTH has the correct value OAUTH, then it prints the value.
 - If the token is not expired, then it prints a message.

8.6.1.2 Setting Client Tracing for Token Authentication

You can add EVENT settings to the client-side sqlnet.ora file to control client tracing.

These EVENT settings can be used for both IAM and Azure AD connections with Oracle Database.

- Use either of the following methods:
 - Add the following settings to the client side sqlnet.ora file:
 - * EVENT 25701=14 for low level tracing
 - * EVENT 25701=15 for high level tracing
 - Set the environment variable EVENT 25701:
 - * EVENT 25701=14 for low level tracing
 - * EVENT 25701=15 for high level tracing

Client trace files are created in the following locations:

- Linux: \$ORACLE HOME/log/diag/clients
- Windows: %ORACLE HOME%\log\diag\clients

You can use the ADR_BASE parameter in the client side sqlnet.ora to specify the directory in which tracing messages are stored. Ensure that the directory path is valid and has write permissions. Ensure that the DIAG ADR ENABLED parameter is not set to FALSE.

An example of setting ADR BASE is as follows:

ADR_BASE=/oracle/oauth2/trace



8.6.2 ORA-12599 and ORA-03114 Errors Caused When Trying to Access a Database Using a Token

The ORA-12599: TNS: cryptographic checksum mismatch and ORA-03114: not connected to ORACLE errors indicate that the database to which you are trying to connect is protected by native network encryption.

When tokens are being used to access an Oracle database, a Transport Layer Security (TLS) connection must be established, not network native encryption. To remedy these errors, ensure that TLS is properly configured for your database. You should test the configuration with a local database user name and password and check the following SYSCONTEXT USERENV parameters:

- NETWORK PROTOCOL
- TLS_VERSION

Related Topics

Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.

8.6.3 Checking the Azure AD Access Token Version

You can check the version of the Microsoft Azure AD access token that your site uses by using the JSON Web Tokens web site.

By default, Azure AD Microsoft Azure AD v1 access token, but your site may have chosen to use v2. Oracle Database supports v1 tokens and Autonomous Database Serverless supports v2 tokens, as well. If you want to use the v2 access tokens, then you can enable their use for the Oracle database. To find the version of the Azure AD access token that you are using, you can either check with your Azure AD administrator, or confirm the version from the JSON Web Tokens website, as follows.

Go to the JSON Web Tokens website.

```
https://jwt.io/
```

- Copy and paste the token string into the Encoded field.
- Check the **Decoded** field, which displays information about the token string.

Near or at the bottom of the field, you will see a claim entitled ver, which indicates either of the following versions:

- "ver": "1.0"
- "ver": "2.0"

Related Topics

Enabling Microsoft Azure AD v2 Access Tokens
 Oracle Database supports integration with the v1 and v2 Azure AD OAuth2 access token.



9

Managing Security for Definer's Rights and Invoker's Rights

Invoker's rights and definer's rights have several security advantages when used to control access to privileges during user-defined procedure executions.

- About Definer's Rights and Invoker's Rights
 Definer's rights and invoker's rights are used to control access to the privileges necessary during the execution of a user-created procedure, or program unit.
- How Procedure Privileges Affect Definer's Rights
 The owner of a procedure, called the *definer*, must have the necessary object privileges for objects that the procedure references.
- How Procedure Privileges Affect Invoker's Rights
 An invoker's rights procedure executes with all of the invoker's privileges.
- When You Should Create Invoker's Rights Procedures
 Oracle recommends that you create invoker's rights procedures in certain situations.
- Controlling Invoker's Rights Privileges for Procedure Calls and View Access
 The INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges regulate the privileges
 used when invoker's rights procedures are run.
- Definer's Rights and Invoker's Rights in Views
 The BEQEATH clause in the CREATE VIEW SQL statement can control definer's rights and invoker's rights in user-created views.
- Using Code Based Access Control for Definer's Rights and Invoker's Rights
 Code based access control, used to attach database roles to PL/SQL functions,
 procedures, or packages, works well with invoker's rights and definer's procedures.
- Controlling Definer's Rights Privileges for Database Links
 You can control privilege grants for definer's rights procedures if your applications use database links and definer's rights procedures.

9.1 About Definer's Rights and Invoker's Rights

Definer's rights and invoker's rights are used to control access to the privileges necessary during the execution of a user-created procedure, or program unit.

In a definer's rights procedure, the procedure executes with the privileges of the owner. The privileges are bound to the schema in which they were created. An invoker's rights procedure executes with the privileges of the current user, that is, the user who invokes the procedure.

For example, suppose user bixby creates a procedure that is designed to modify table cust_records and then he grants the EXECUTE privilege on this procedure to user rlayton. If bixby had created the procedure with definer's rights, then the procedure would look for table cust_records in bixby's schema. Had the procedure been created with invoker's rights, then when rlayton runs it, the procedure would look for table cust_records in rlayton's schema.

By default, all procedures are considered definer's rights. You can designate a procedure to be an invoker's rights procedure by using the AUTHID CURRENT_USER clause when you create or modify it, or you can use the AUTHID DEFINER clause to make it a definer's rights procedure.

You can create privilege analysis policies to capture privilege use of definer's rights and invoker's rights procedures.

Related Topics

- Performing Privilege Analysis to Identify Privilege Use
 Privilege analysis dynamically analyzes the privileges and roles that users use and do not use
- Oracle Database PL/SQL Language Reference

9.2 How Procedure Privileges Affect Definer's Rights

The owner of a procedure, called the *definer*, must have the necessary object privileges for objects that the procedure references.

If the procedure owner grants to another user the right to use the procedure, then the privileges of the procedure owner (on the objects the procedure references) apply to the grantee's exercise of the procedure. The privileges of the procedure's definer must be granted directly to the procedure owner, not granted through roles. These are called definer's rights.

The user of a procedure who is not its owner is called the *invoker*. Additional privileges on referenced objects are required for an invoker's rights procedure, but not for a definer's rights procedure.

A user of a definer's rights procedure requires only the privilege to execute the procedure and no privileges on the underlying objects that the procedure accesses. This is because a definer's rights procedure operates under the security domain of the user who owns the procedure, regardless of who is executing it. The owner of the procedure must have all the necessary object privileges for referenced objects. Fewer privileges need to be granted to users of a definer's rights procedure. This results in stronger control of database access.

You can use definer's rights procedures to control access to private database objects and add a level of database security. By writing a definer's rights procedure and granting only the EXECUTE privilege to a user, this user can be forced to access the referenced objects only through the procedure.

At run time, Oracle Database checks whether the privileges of the owner of a definer's rights procedure allow access to that procedure's referenced objects, before the procedure is executed. If a necessary privilege on a referenced object was revoked from the owner of a definer's rights procedure, then no user, including the owner, can run the procedure.

An example of when you may want to use a definer's rights procedure is as follows: Suppose that you must create an API whose procedures have unrestricted access to its tables, but you want to prevent ordinary users from selecting table data directly, and from changing it with INSERT, UPDATE, and DELETE statements. To accomplish this, in a separate, low-privileged schema, create the tables and the procedures that comprise the API. By default, each procedure is a definer's rights unit, so you do not need to specify AUTHID DEFINER when you create it. Then grant the EXECUTE privilege to the users who must use this API, but do not grant any privileges that allow data access. This solution gives you complete control over your API behavior and how users have access to its underlying objects.

Oracle recommends that you create your definer's rights procedures, and views that access these procedures, in their own schema. Grant this schema very low privileges, or no privileges



at all. This way, when other users run these procedures or views, they will not have access to any unnecessarily high privileges from this schema.



Trigger processing follows the same patterns as definer's rights procedures. The user runs a SQL statement, which that user is privileged to run. As a result of the SQL statement, a trigger is fired. The statements within the triggered action temporarily execute under the security domain of the user that owns the trigger. For overview information about triggers, *Oracle Database Concepts*.

Related Topics

How Roles Work in PL/SQL Blocks
 Role behavior in a PL/SQL block is determined by the type of block and by definer's rights
 or invoker's rights.

9.3 How Procedure Privileges Affect Invoker's Rights

An invoker's rights procedure executes with all of the invoker's privileges.

Oracle Database enables the privileges that were granted to the invoker through any of the invoker's enabled roles to take effect, unless a definer's rights procedure calls the invoker's rights procedure directly or indirectly. A user of an invoker's rights procedure must have privileges (granted to the user either directly or through a role) on objects that the procedure accesses through external references that are resolved in the schema of the invoker. When the invoker runs an invoker's rights procedure, this user temporarily has *all* of the privileges of the invoker.

The invoker must have privileges at run time to access program references embedded in DML statements or dynamic SQL statements, because they are effectively recompiled at run time.

For all other external references, such as direct PL/SQL function calls, Oracle Database checks the privileges of the owner at compile time, but does not perform a run-time check. Therefore, the user of an invoker's rights procedure does not need privileges on external references outside DML or dynamic SQL statements. Therefore, the developer of an invoker's rights procedure only needs to grant privileges on the procedure itself, not on all objects directly referenced by the invoker's rights procedure.

You can create a software bundle that consists of multiple program units, some with definer's rights and others with invoker's rights, and restrict the program entry points *(controlled step-in)*. A user who has the privilege to run an entry-point procedure can also execute internal program units indirectly, but cannot directly call the internal programs. For very precise control over query processing, you can create a PL/SQL package specification with explicit cursors.

Related Topics

Controlling Invoker's Rights Privileges for Procedure Calls and View Access
 The INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges regulate the privileges used when invoker's rights procedures are run.

9.4 When You Should Create Invoker's Rights Procedures

Oracle recommends that you create invoker's rights procedures in certain situations.

These situations are as follows:

- When creating a PL/SQL procedure in a high-privileged schema. When lower-privileged users invoke the procedure, then it can do no more than those users are allowed to do. In other words, the invoker's rights procedure runs with the privileges of the invoking user
- When the PL/SQL procedure contains no SQL and is available to other users. The DBMS_OUTPUT PL/SQL package is an example of a PL/SQL subprogram that contains no SQL and is available to all users. The reason you should use an invoker's rights procedure in this situation is because the unit issues no SQL statements at run time, so the run-time system does not need to check their privileges. Specifying AUTHID CURRENT_USER makes invocations of the procedure more efficient, because when an invoker's right procedure is pushed onto, or comes from, the call stack, the values of CURRENT_USER and CURRENT_SCHEMA, and the currently enabled roles do not change.

See Also:

- Configuration of Oracle Virtual Private Database Policies
- About ANY Privileges and the PUBLIC Role
- Oracle Database PL/SQL Packages and Types Reference for information about how Oracle Database handles name resolution and privilege checking at runtime using invoker's and definer's rights
- Oracle Database PL/SQL Packages and Types Reference for more information about the differences between invoker's rights and definer's rights units
- Oracle Database PL/SQL Packages and Types Reference for information about defining explicit cursors in the CREATE PACKAGE statement

9.5 Controlling Invoker's Rights Privileges for Procedure Calls and View Access

The INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges regulate the privileges used when invoker's rights procedures are run.

- How the Privileges of a Schema Affect the Use of Invoker's Rights Procedures
 An invoker's rights procedure is useful in situations where a lower-privileged user must
 execute a procedure owned by a higher-privileged user.
- How the INHERIT [ANY] PRIVILEGES Privileges Control Privilege Access
 Use the INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges to secure invoker's
 rights procedures.
- Grants of the INHERIT PRIVILEGES Privilege to Other Users

 By default, all users are granted INHERIT PRIVILEGES ON USER newuser TO PUBLIC.
- Example: Granting INHERIT PRIVILEGES on an Invoking User
 The GRANT statement can grant the INHERIT PRIVILEGES privilege on an invoking user to a procedure owner.
- Example: Revoking INHERIT PRIVILEGES

 The REVOKE statement can revoke the INHERIT PRIVILEGES privilege from a user.



- Grants of the INHERIT ANY PRIVILEGES Privilege to Other Users

 By default, user SYS has the INHERIT ANY PRIVILEGES system privilege and can grant this privilege to other database users or roles.
- Example: Granting INHERIT ANY PRIVILEGES to a Trusted Procedure Owner

 The GRANT statement can grant the INHERIT ANY PRIVILEGES privilege to trusted procedure
 owners.
- Managing INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES
 By default, PUBLIC has the INHERIT PRIVILEGE privilege on new and upgraded user accounts; the SYS user has the INHERIT ANY PRIVILEGES privilege.

9.5.1 How the Privileges of a Schema Affect the Use of Invoker's Rights Procedures

An invoker's rights procedure is useful in situations where a lower-privileged user must execute a procedure owned by a higher-privileged user.

When a user runs an invoker's rights procedure (or any PL/SQL program unit that has been created with the AUTHID CURRENT_USER clause), the procedure temporarily inherits all of the privileges of the invoking user while the procedure runs.

During that time, the procedure owner has, through the procedure, access to this invoking user's privileges. Consider the following scenario:

- 1. User ebrown creates the check_syntax invoker's rights procedure and then grants user jward the EXECUTE privilege on it.
- 2. User <code>ebrown</code>, who is a junior programmer, has only the minimum set of privileges necessary for his job. The <code>check syntax procedure resides</code> in <code>ebrown's schema</code>.
- 3. User jward, who is a manager, has a far more powerful set of privileges than user ebrown.
- **4.** When user <code>jward</code> runs the <code>check_syntax</code> invoker's rights procedure, the procedure inherits user <code>jward</code>'s higher privileges while it runs.
- 5. Because user ebrown owns the check_syntax procedure, he has access to user jward's privileges whenever jward runs the check syntax procedure.

The danger in this type of situation—in which the lower privileged <code>ebrown</code>'s procedure has access to <code>jward</code>'s higher privileges whenever <code>jward</code> runs the procedure—lies in the risk that the procedure owner can misuse the higher privileges of the invoking user. For example, user <code>ebrown</code> could make use of <code>jward</code>'s higher privileges by rewriting the <code>check_syntax</code> procedure to give <code>ebrown</code> a raise or delete <code>ebrown</code>'s bad performance appraisal record. Or, <code>ebrown</code> originally could have created the procedure as a definer's rights procedure, granted its <code>EXECUTE</code> privilege to <code>jward</code>, and then later on change it to a potentially malicious invoker's rights procedure without letting <code>jward</code> know. These types of risks increase when random users, such as application users, have access to a database that uses invoker's rights procedures.

When user jward runs ebrown's invoker's rights procedure, there is an element of trust involved. He must be assured that ebrown will not use the <code>check_syntax</code> procedure in a malicious way when it accesses <code>jward's</code> privileges. The <code>INHERIT PRIVILEGES</code> and <code>INHERIT ANY PRIVILEGES</code> privileges can help user <code>jward</code> control whether user <code>ebrown's</code> procedure can have access to his (<code>jward's</code>) privileges. Any user can grant or revoke the <code>INHERIT PRIVILEGES</code> privilege on themselves to the user whose invoker's rights procedures they want to run. <code>SYS</code> users manage the <code>INHERIT ANY PRIVILEGES</code> privilege.



9.5.2 How the INHERIT [ANY] PRIVILEGES Privileges Control Privilege Access

Use the INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges to secure invoker's rights procedures.

The INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges regulate the privileges used when a user runs an invoker's rights procedure or queries a BEQUEATH CURRENT_USER view that references an invoker's rights procedure.

When a user runs an invoker's rights procedure, Oracle Database checks it to ensure that the procedure owner has either the INHERIT PRIVILEGES privilege on the invoking user, or if the owner has been granted the INHERIT ANY PRIVILEGES privilege. If the privilege check fails, then Oracle Database returns an ORA-06598: insufficient INHERIT PRIVILEGES privilege error.

The benefit of these two privileges is that they give invoking users control over who can access their privileges when they run an invoker's rights procedure or query a BEQUEATH CURRENT_USER view.

9.5.3 Grants of the INHERIT PRIVILEGES Privilege to Other Users

By default, all users are granted INHERIT PRIVILEGES ON USER newuser TO PUBLIC.

This grant takes place when the user accounts are created or when accounts that were created earlier are upgraded to the current release.

The invoking user can revoke the INHERIT PRIVILEGE privilege from other users on himself and then grant it only to users that he trusts.

The syntax for the INHERIT PRIVILEGES privilege grant is as follows:

GRANT INHERIT PRIVILEGES ON USER invoking_user TO procedure_owner;

In this specification:

- invoking_user is the user who runs the invoker's rights procedure. This user must be a
 database user account.
- procedure_owner is the user who owns the invoker's rights procedure. This value must be
 a database user account. As an alternative to granting the INHERIT PRIVILEGES privilege
 to the procedure's owner, you can grant the privilege to a role that is in turn granted to the
 procedure.

The following users or roles must have the INHERIT PRIVILEGES privilege granted to them by users who will run their invoker's rights procedures:

- Users or roles who own the invoker's rights procedures
- Users or roles who own bequeath current user views

9.5.4 Example: Granting INHERIT PRIVILEGES on an Invoking User

The GRANT statement can grant the INHERIT PRIVILEGES privilege on an invoking user to a procedure owner.



Example 9-1 shows how the invoking user jward can grant user ebrown the INHERIT PRIVILEGES privilege.

Example 9-1 Granting INHERIT PRIVILEGES on an Invoking User to a Procedure Owner

GRANT INHERIT PRIVILEGES ON USER jward TO ebrown;

The statement enables any invoker's rights procedure that <code>ebrown</code> writes, or will write in the future, to access <code>jward</code>'s privileges when <code>jward</code> runs it.

9.5.5 Example: Revoking INHERIT PRIVILEGES

The REVOKE statement can revoke the INHERIT PRIVILEGES privilege from a user.

Example 9-2 shows how user jward can revoke the use of his privileges from ebrown.

Example 9-2 Revoking INHERIT PRIVILEGES

REVOKE INHERIT PRIVILEGES ON USER jward FROM ebrown;

9.5.6 Grants of the INHERIT ANY PRIVILEGES Privilege to Other Users

By default, user SYS has the INHERIT ANY PRIVILEGES system privilege and can grant this privilege to other database users or roles.

As with all ANY privileges, only grant this privilege to trusted users or roles. Once a user or role has been granted the INHERIT ANY PRIVILEGES privilege, then this user's invoker's rights procedures have access to the privileges of the invoking user. You can find the users who have been granted the INHERIT ANY PRIVILEGES privilege by querying the DBA_SYS_PRIVS data dictionary view.

9.5.7 Example: Granting INHERIT ANY PRIVILEGES to a Trusted Procedure Owner

The GRANT statement can grant the INHERIT ANY PRIVILEGES privilege to trusted procedure owners.

Example 9-3 shows how to grant the INHERIT ANY PRIVILEGES privilege to user ebrown.

Example 9-3 Granting INHERIT ANY PRIVILEGES to a Trusted Procedure Owner

GRANT INHERIT ANY PRIVILEGES TO ebrown;

Be careful about revoking the INHERIT ANY PRIVILEGES privilege from powerful users. For example, suppose user SYSTEM has created a set of invoker's rights procedures. If you revoke INHERIT ANY PRIVILEGES from SYSTEM, then other users cannot run his procedures, unless they have specifically granted him the INHERIT PRIVILEGE privilege.

9.5.8 Managing INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES

By default, PUBLIC has the INHERIT PRIVILEGE privilege on new and upgraded user accounts; the SYS user has the INHERIT ANY PRIVILEGES privilege.

Oracle by default configures a set of grants of INHERIT PRIVILEGES that are designed to help protect against misuse of the privileges of various Oracle-defined users.



You can choose to revoke the default grant of INHERIT PRIVILEGES ON USER user_name TO PUBLIC for a customer-defined user and grant more specific grants of INHERIT PRIVILEGES as appropriate for that particular user. To find the users who have been granted the INHERIT ANY PRIVILEGES privilege, query the DBA SYS PRIVS data dictionary view.

1. Revoke the INHERIT PRIVILEGES privilege from PUBLIC.

For example:

REVOKE INHERIT PRIVILEGES ON invoking user FROM PUBLIC;

Be aware that this time, any users who run invoker's rights procedures cannot do so, due to run-time errors from failed INHERIT PRIVILEGES checks.

- 2. Selectively grant the INHERIT PRIVILEGES privilege to trusted users or roles.
- Similarly, selectively grant the INHERIT ANY PRIVILEGES privilege only to trusted users or roles

You can create an audit policy to audit the granting and revoking of these two privileges, but you cannot audit run-time errors that result from failed INHERIT PRIVILEGES privilege checks.

See Also:

- Oracle Database PL/SQL Packages and Types Reference for information about SQL injection attacks
- Oracle Database PL/SQL Packages and Types Reference for more information about the GRANT statement and default privileges

9.6 Definer's Rights and Invoker's Rights in Views

The BEQEATH clause in the CREATE VIEW SQL statement can control definer's rights and invoker's rights in user-created views.

- About Controlling Definer's Rights and Invoker's Rights in Views
 You can configure user-defined views to accommodate invoker's rights functions that are
 referenced in the view.
- Using the BEQUEATH Clause in the CREATE VIEW Statement
 The BEQUEATH controls how an invoker's right function can be executed using the rights of the invoking user.
- Finding the User Name or User ID of the Invoking User
 PL/SQL functions can be used to find the invoking user, based on whether invoker's rights or definer's rights are being used.
- Finding BEQUEATH DEFINER and BEQUEATH_CURRENT_USER Views
 You can find out if a view is a BEQUEATH DEFINER or BEQUEATH CURRENT USER view.

9.6.1 About Controlling Definer's Rights and Invoker's Rights in Views

You can configure user-defined views to accommodate invoker's rights functions that are referenced in the view.



When a user invokes an identity- or privilege-sensitive SQL function or an invoker's rights PL/SQL or Java function, then current schema, current user, and currently enabled roles within the operation's execution can be inherited from the querying user's environment, rather than being set to the owner of the view.

This configuration does not turn the view itself into an invoker's rights object. Name resolution within the view is still handled using the view owner's schema, and privilege checking for the view is done using the view owner's privileges. However, at runtime, the function referenced by view runs under the invoking user's privileges rather than those of the view owner's.

The benefit of this feature is that it enables functions such as SYS_CONTEXT and USERENV, which must return information accurate for the invoking user, to return consistent results when these functions are referenced in a view.

9.6.2 Using the BEQUEATH Clause in the CREATE VIEW Statement

The BEQUEATH controls how an invoker's right function can be executed using the rights of the invoking user.

To enable an invoker's rights function to be executed using the rights of the user issuing SQL that references the view, in the CREATE VIEW statement, you can set the BEQUEATH clause to CURRENT USER.

If you plan to issue a SQL query or DML statement against the view, then the view owner must be granted the INHERIT PRIVILEGES privilege on the invoking user or the view owner must have the INHERIT ANY PRIVILEGES privilege. If not, then when a SELECT query or DML statement involves a BEQUEATH CURRENT_USER view, the run-time system will raise error ORA-06598: insufficient INHERIT PRIVILEGES privilege.

Use the use BEQUEATH CURRENT_USER clause to set the view's function to be executed
using invoker's rights.

For example:

```
CREATE VIEW MY_OBJECTS_VIEW BEQUEATH CURRENT_USER AS SELECT GET_OBJS_FUNCTION;
```

If you want the function within the view to be executed using the view owner's rights, then you should either omit the BEQUEATH clause or set it to DEFINER.

For example:

```
CREATE VIEW my_objects_view BEQUEATH DEFINER AS SELECT OBJECT NAME FROM USER OBJECTS;
```

See Also:

- Controlling Invoker's Rights Privileges for Procedure Calls and View Access for more information about how the INHERIT PRIVILEGE privilege works
- Oracle Database SQL Language Reference for additional information about granting the INHERIT PRIVILEGES and INHERIT ANY PRIVILEGES privileges
- Oracle Database Real Application Security Administrator's and Developer's Guide for information about how to use BEQUEATH CURRENT_USER views with Oracle Database Real Application Security applications



9.6.3 Finding the User Name or User ID of the Invoking User

PL/SQL functions can be used to find the invoking user, based on whether invoker's rights or definer's rights are being used.

- Use the ORA_INVOKING_USER or ORA_INVOKING_USERID function to find the invoking user based on whether invoker's rights or definer's rights:
 - ORA_INVOKING_USER: Use this function to return the name of the user who is invoking
 the current statement or view. This function treats the intervening views as specified by
 their BEQUEATH clauses. If the invoking user is an Oracle Database Real Application
 Security-defined user, then this function returns XS\$NULL.
 - ORA_INVOKING_USERID: Use this function to return the identifier (ID) of the user who is invoking the current statement or view. This function treats the intervening views as specified by their BEQUEATH clauses. If the invoking user is an Oracle Database Real Application Security-defined user, then this function returns an ID that is common to all Real Application Security sessions but is different from the ID of any database user.

For example:

```
CONNECT HR
Enter password: password

SELECT ORA_INVOKING_USER FROM DUAL;

ORA_INVOKING_USER

HR
```

See Also:

Oracle Database Real Application Security Administrator's and Developer's Guide for information about similar functions that are used for Oracle Database Real Application Security applications

9.6.4 Finding BEQUEATH DEFINER and BEQUEATH_CURRENT_USER Views

You can find out if a view is a BEQUEATH DEFINER OF BEQUEATH CURRENT USER view.

• To find if a view is BEQUEATH DEFINER or BEQUEATH CURRENT_USER view, query the BEQUEATH column of a * VIEWS or * VIEWS AE static data dictionary view for that view.



See Also:

- Oracle Database Reference for more information about *_VIEWS static data dictionary views
- Oracle Database Reference for more information about *_VIEWS_AE static data dictionary views

For example:

```
SELECT BEQUEATH FROM USER_VIEWS WHERE VIEW_NAME = 'MY_OBJECTS';

BEQUEATH
------
CURRENT_USER
```

9.7 Using Code Based Access Control for Definer's Rights and Invoker's Rights

Code based access control, used to attach database roles to PL/SQL functions, procedures, or packages, works well with invoker's rights and definer's procedures.

- About Using Code Based Access Control for Applications
 You can use code based access control (CBAC) to better manage definer's rights program
 units.
- Who Can Grant Code Based Access Control Roles to a Program Unit?
 Code based access control roles can be granted to a program unit if a set of conditions are met.
- How Code Based Access Control Works with Invoker's Rights Program Units
 Code based access control can run a program unit in an invoking user's context and with
 roles associated with this context.
- How Code Based Access Control Works with Definer's Rights Program Units Code based access control can be used to secure definer's rights.
- Grants of Database Roles to Users for Their CBAC Grants
 The DELEGATE option in the GRANT statement can limit privilege grants to roles by users responsible for CBAC grants.
- Grants and Revokes of Database Roles to a Program Unit
 The GRANT and REVOKE statements can grant database roles to or revoke database roles from a program unit.
- Tutorial: Controlling Access to Sensitive Data Using Code Based Access Control
 This tutorial demonstrates how to control access to sensitive data in the HR schema by
 using code based access control.

9.7.1 About Using Code Based Access Control for Applications

You can use code based access control (CBAC) to better manage definer's rights program units.



Applications must often run program units in the caller's environment, while requiring elevated privileges. PL/SQL programs traditionally make use of definer's rights to temporarily elevate the privileges of the program.

However, definer's rights based program units run in the context of the definer or the owner of the program unit, as opposed to the invoker's context. Also, using definer's rights based programs often leads to the program unit getting more privileges than required.

Code based access control (CBAC) provides the solution by enabling you to attach database roles to a PL/SQL function, procedure, or package. These database roles are enabled at run time, enabling the program unit to execute with the required privileges in the calling user's environment.

You can create privilege analysis policies that capture the use of CBAC roles.

Related Topics

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

9.7.2 Who Can Grant Code Based Access Control Roles to a Program Unit?

Code based access control roles can be granted to a program unit if a set of conditions are met.

These conditions are as follows:

- The grantor is user SYS or owns the program unit.
- If the grantor owns the program unit, then the grantor must have the GRANT ANY ROLE system privilege, or have the ADMIN or DELEGATE option for the roles that they want to grant to program units.
- The roles to be granted are directly granted roles to the owner.
- The roles to be granted are standard database roles.

If these three conditions are not met, then error ORA-28702: Program unit string is not owned by the grantor is raised if the first condition is not met, and error ORA-1924: role 'string' not granted or does not exist is raised if the second and third conditions are not met.

Related Topics

- Grants of Database Roles to Users for Their CBAC Grants
 The DELEGATE option in the GRANT statement can limit privilege grants to roles by users responsible for CBAC grants.
- Grants and Revokes of Database Roles to a Program Unit
 The GRANT and REVOKE statements can grant database roles to or revoke database roles from a program unit.

9.7.3 How Code Based Access Control Works with Invoker's Rights Program Units

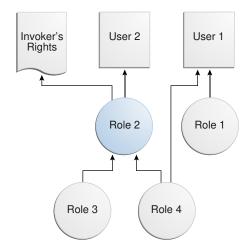
Code based access control can run a program unit in an invoking user's context and with roles associated with this context.



Consider a scenario where there are two application users, 1 and 2. Application user 2 creates the invoker's right program unit, grants database role 2 to the invoker's rights unit, and then grants execute privileges on the invoker's rights unit to application user 1.

The following diagram shows the database roles 1 and 2 granted to application users 1 and 2, and an invoker's right program unit.

Figure 9-1 Roles Granted to Application Users and Invoker's Right Program Unit



The grants are as follows:

- Application user 1 is directly granted database roles 1 and 4.
- Application user 2 is directly granted database role 2, which includes application roles 3 and 4.
- The invoker's right program unit is granted database role 2.

When application user 1 logs in and executes the invoker's rights program unit, then the invoker's rights unit executes with the combined database roles of user 1 and the database roles attached to the invoker's rights unit.

The following diagram shows the security context in which the invoker's rights unit is executed. When application user 1 first logs on, application user 1 has the database PUBLIC role (by default), and the database roles 1 and 4, which have been granted to it. Application user 1 next executes the invoker's rights program unit created by application user 2.

The invoker's rights unit executes in application user 1's context, and has the additional database role 2 attached to it. Database roles 3 and 4 are included, as they are a part of database role 2. After the invoker's rights unit exits, then application user 1 only has the application roles that have been granted to it, PUBLIC, role 1, and role 4.



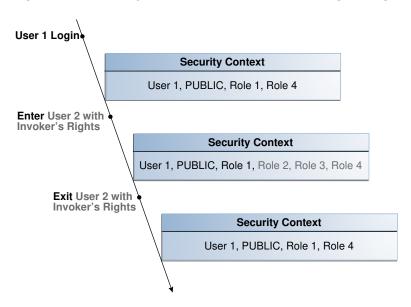


Figure 9-2 Security Context in Which Invoker's Right Program Unit IR Is Executed

9.7.4 How Code Based Access Control Works with Definer's Rights Program Units

Code based access control can be used to secure definer's rights.

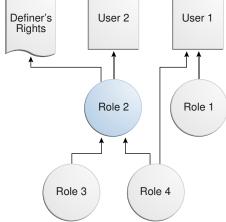
Code based access control works with definer's rights program units to enable the program unit to run using the defining user's rights, with the privileges of a combined set of database roles that are associated with this user.

Consider a scenario where application user 2 creates a definer's rights program unit, grants role 2 to the definer's rights program unit, and then grants the EXECUTE privilege on the definer's rights program unit to application user 1.

The following diagram shows the database roles granted to application users 1 and 2, and a definer's rights program unit.

Figure 9-3 Roles Granted to Application Users and Definer's Rights Program Unit

| Definer's | User 2 | User 1 |





The grants are as follows:

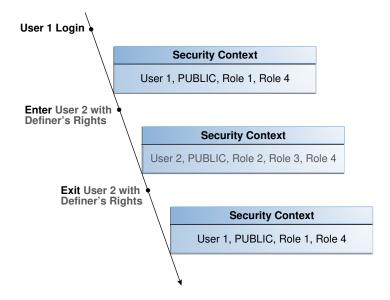
- Application user 1 is directly granted database roles 1 and 4.
- Application user 2 is directly granted database role2, which includes database roles 3 and 4.
- The definer's right program unit is granted database role 2.

When application user 1 logs in and executes definer's right program unit, then the definer's rights unit executes with the combined database roles of application user 2 and the database roles attached to the definer's rights unit (roles 2, 3, and 4).

The following diagram shows the security context in which the definer's right program unit is executed. When application user 1 first logs on, application user 1 has the database PUBLIC role (by default), and the database roles 1 and 4, which have been granted to it. Application user 1 next executes the definer's rights program unit created by application user 2.

The definer's rights program unit executes in application user 2's context, and has the additional database role 2 attached to it. Database roles 3 and 4 are included, as they are a part of database role 2. After the definer's rights unit exits, application user 1 only has the database roles that have been granted to it (PUBLIC, role 1, and role 4).

Figure 9-4 Security Context in Which Definer's Right Program Unit DR Is Executed



9.7.5 Grants of Database Roles to Users for Their CBAC Grants

The DELEGATE option in the GRANT statement can limit privilege grants to roles by users responsible for CBAC grants.

When you grant a database role to a user who is responsible for CBAC grants, you can include the DELEGATE option in the GRANT statement to prevent giving the grantee additional privileges on the roles.

The DELEGATE option enables the roles to be granted to program units, but it does not permit the granting of the role to other principals or the administration of the role itself. You also can use the ADMIN option for the grants, which does permit the granting of the role to other principals. Both the ADMIN and DELEGATE options are compatible; that is, you can grant both to

a user, though you must do this in separate <code>GRANT</code> statements for each option. To find if a user has been granted a role with these options, query the <code>DELEGATE_OPTION</code> column or the <code>ADMIN_OPTION</code> column of either the <code>USER_ROLE_PRIVS</code> or <code>DBA_ROLE_PRIVS</code> for the user.

The syntax for using the Delegate and Admin option is as follows:

```
GRANT role_list to user_list WITH DELEGATE OPTION;

GRANT role_list to user_list WITH ADMIN OPTION;

For example:

GRANT cb_role1 to usr1 WITH DELEGATE OPTION;

GRANT cb_role1 to usr1 WITH ADMIN OPTION;

GRANT cb_role1, cb_role2 to usr1, usr2 with DELEGATE OPTION;

GRANT cb role1, cb role2 to usr1, usr2 with ADMIN OPTION;
```

In a multitenant environment, you can use the DELEGATE option for common grants such as granting common roles to common users, just as you can with the ADMIN option.

For example:

```
GRANT c##cb role1 to c##usr1 WITH DELEGATE OPTION CONTAINER = ALL;
```

Be aware that CBAC grants themselves can only take place locally in a PDB.

See Also:

Oracle Database SQL Language Reference for more information about the ADMIN option

9.7.6 Grants and Revokes of Database Roles to a Program Unit

The GRANT and REVOKE statements can grant database roles to or revoke database roles from a program unit.

The following syntax to grants or revokes database roles for a PL/SQL function, procedure, or package:

```
GRANT role_list TO code_list
REVOKE {role_list | ALL} FROM code_list
```

In this specification:

For example:



```
GRANT cb_role1 TO FUNCTION func1, PACKAGE pack1;

GRANT cb_role2, cb_role3 TO FUNCTION HR.func2, PACKAGE SYS.pack2;

REVOKE cb_role1 FROM FUNCTION func1, PACKAGE pack1;

REVOKE ALL FROM FUNCTION HR.func2, PACKAGE SYS.pack2;
```

Related Topics

- Who Can Grant Code Based Access Control Roles to a Program Unit?
 Code based access control roles can be granted to a program unit if a set of conditions are met.
- Grants of Database Roles to Users for Their CBAC Grants
 The DELEGATE option in the GRANT statement can limit privilege grants to roles by users responsible for CBAC grants.

9.7.7 Tutorial: Controlling Access to Sensitive Data Using Code Based Access Control

This tutorial demonstrates how to control access to sensitive data in the HR schema by using code based access control.

- About This Tutorial
 In this tutorial, you will create a user who must have access to specific employee information for his department.
- Step 1: Create the User and Grant HR the CREATE ROLE Privilege
 To begin, you must create the "Finance" user account and then grant this the HR user the
 CREATE ROLE privilege.
- Step 2: Create the print_employees Invoker's Rights Procedure
 The print_employees invoker's rights procedure shows employee information in the
 current user's department.
- Step 3: Create the hr_clerk Role and Grant Privileges for It

 Next, you are ready to create the hr_clerk role, which must have the EXECUTE privilege on
 the print employees procedure.
- Step 4: Test the Code Based Access Control HR.print_employees Procedure
 At this stage, you are ready to test the code based access control HR.print_employees procedure.
- Step 5: Create the view_emp_role Role and Grant Privileges for It Next, user HR must create the view_emp_role role and then grant privileges to it.
- Step 6: Test the HR.print_employees Procedure Again
 With the appropriate privileges in place, user "Finance" can try the HR.print_employees procedure again.
- Step 7: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

9.7.7.1 About This Tutorial

In this tutorial, you will create a user who must have access to specific employee information for his department.



However, the table HR.EMPLOYEES contains sensitive information such as employee salaries, which must not be accessible to the user. You will implement access control using code based access control. The employee data will be shown to the user through an invoker's rights procedure. Instead of granting the SELECT privilege directly to the user, you will grant the SELECT privilege to the invoker's rights procedure through a database role. In the procedure, you will hide the sensitive information, such as salaries. Because the procedure is an invoker's rights procedure, you know the caller's context inside the procedure. In this case, the caller's context is for the Finance department. The user is named "Finance", so that only data for employees who work in the Finance department is accessible to the user.

9.7.7.2 Step 1: Create the User and Grant HR the CREATE ROLE Privilege

To begin, you must create the "Finance" user account and then grant this the HR user the CREATE ROLE privilege.

1. Log into the database instance as an administrator who has privileges to create user accounts and roles.

For example:

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

2. Create the "Finance" user account.

```
GRANT CONNECT TO "Finance" IDENTIFIED BY password;
```

Ensure that you enter "Finance" in the case shown, enclosed by double quotation marks. Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

3. Grant the CREATE ROLE privilege to user HR.

```
GRANT CREATE ROLE TO HR;
```

9.7.7.3 Step 2: Create the print_employees Invoker's Rights Procedure

The print_employees invoker's rights procedure shows employee information in the current user's department.

You must create this procedure as an invoker's rights procedure because you must know who the caller is when inside the procedure.

1. Connect as user HR.

```
CONNECT HR
Enter password: password
```

2. Create the print employees procedure as follows.

In this example:

- dbms output.put line prints the table header.
- for rec in (select ... finds the employee information for the caller's department, which for this tutorial is the Finance department for user "Finance". Had you created a user named "Marketing" (which is also listed in the DEPARTMENT_NAME column of the HR.EMPLOYEES table), then the procedure could capture information for Marketing employees.
- loop and dbms_output.put_line populate the output with the employee data from the Finance department.

9.7.7.4 Step 3: Create the hr_clerk Role and Grant Privileges for It

Next, you are ready to create the hr_clerk role, which must have the EXECUTE privilege on the print employees procedure.

After you create this role, you must grant it to "Finance".

Create the hr clerk role.

```
CREATE ROLE hr clerk;
```

2. Grant the EXECUTE privilege on the print_employees procedure to the hr clerk role.

```
GRANT EXECUTE ON print employees TO hr clerk;
```

3. Grant the hr clerk role to "Finance".

```
GRANT hr clerk TO "Finance";
```

9.7.7.5 Step 4: Test the Code Based Access Control HR.print_employees Procedure

At this stage, you are ready to test the code based access control HR.print_employees procedure.

To test the code based access control HR.print_employees procedure, user "Finance" must query the HR.EMPLOYEES table and try to run the HR.print employees procedure.

Connect to the database instance as user "Finance".

```
CONNECT "Finance"
Enter password: password
```

2. Try to directly query the HR. EMPLOYEES table.

```
SELECT EMPLOYEE ID, FIRST NAME, LAST NAME, SALARY FROM HR.EMPLOYEES;
```

The query fails because user Finance does not have the SELECT privilege for HR.EMPLOYEES.

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

3. Execute the HR.print employees procedure.

```
EXEC HR.print employees;
```

The query fails because user "Finance" does not have the appropriate privileges.

```
ERROR at line 1: ORA-00942: table or view does not exist ORA-06512: at "HR.PRINT EMPLOYEES", line 13ORA-06512: at line 1
```

9.7.7.6 Step 5: Create the view_emp_role Role and Grant Privileges for It

Next, user HR must create the view emp role role and then grant privileges to it.

User HR grants the SELECT privilege HR.EMPLOYEES and HR.DEPARTMENTS to the view_emp_role role, and then grants SELECT on HR.EMPLOYEES and HR.DEPARTMENTS to the view emp_role role.

Connect as user HR.

```
CONNECT HR
Enter password: password
```

Create the view emp role role.

```
CREATE ROLE view emp role;
```

Grant the SELECT privilege on HR.EMPLOYEES and HR.DEPARTMENTS to the view_emp_role role.

```
GRANT SELECT ON HR.EMPLOYEES TO view_emp_role;
GRANT SELECT ON HR.DEPARTMENTS TO view_emp_role;
```

4. Grant the view emp role role to the HR.print employees invoker's rights procedure.

```
GRANT view emp role TO PROCEDURE HR.print employees;
```

9.7.7.7 Step 6: Test the HR.print_employees Procedure Again

With the appropriate privileges in place, user "Finance" can try the HR.print_employees procedure again.

1. Connect as user "Finance".

```
CONNECT "Finance"
Enter password: password
```

2. Set the server output to display.

```
SET SERVEROUTPUT ON;
```

3. Try to directly query the HR. EMPLOYEES table.

```
SELECT EMPLOYEE ID, FIRST NAME, LAST NAME, SALARY FROM HR.EMPLOYEES;
```

The query fails.

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

4. Execute the HR.print employees procedure to show the employee information.

EXEC HR.print employees;

The call succeeds.

ID	First Name	Last Name	Email	Phone Number
108	Nancy	Greenberg	NGREENBE	515.124.4569
109	Daniel	Faviet	DFAVIET	515.124.4169
110	John	Chen	JCHEN	515.124.4269
111	Ismael	Sciarra	ISCIARRA	515.124.4369
112	Jose Manuel	Urman	JMURMAN	515.124.4469
113	Luis	Popp	LPOPP	515.124.4567

PL/SQL procedure successfully completed.

9.7.7.8 Step 7: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

1. Connect as a user with administrative privileges.

For example:

```
CONNECT sec_admin
Enter password: password
```

2. Drop the user "Finance".

```
DROP USER "Finance";
```

Drop the hr clerk role.

```
DROP ROLE hr clerk;
```

Connect as user HR.

```
CONNECT HR
Enter password: password
```

5. Drop the view_emp_role role and the HR.print employees procedure.

```
DROP ROLE view_emp_role;
DROP PROCEDURE print employees;
```

6. Connect as the administrative user.

```
CONNECT sec_admin
Enter password: password
```

7. Revoke the CREATE ROLE privilege from HR.

```
REVOKE CREATE ROLE FROM HR;
```

9.8 Controlling Definer's Rights Privileges for Database Links

You can control privilege grants for definer's rights procedures if your applications use database links and definer's rights procedures.

About Controlling Definer's Rights Privileges for Database Links
 When a definer's rights procedure connects to a database link, operations on the database link should use the procedure owner's credentials.

- Grants of the INHERIT REMOTE PRIVILEGES Privilege to Other Users

 The INHERIT REMOTE PRIVILEGES privilege enables the current user to have explicit privileges over the connected user in the database.
- Example: Granting INHERIT REMOTE PRIVILEGES on a Connected User You can grant the INHERIT REMOTE PRIVILEGES privilege on a connected user to the current user.
- Grants of the INHERIT ANY REMOTE PRIVILEGES Privilege to Other Users
 The INHERIT ANY REMOTE PRIVILEGES privilege enables the grantee user to open a
 connected user database link as any user.
- Revokes of the INHERIT [ANY] REMOTE PRIVILEGES Privilege
 The methods for revoking the INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges differ.
- Example: Revoking the INHERIT REMOTE PRIVILEGES Privilege

 The REVOKE SQL statement can revoke the INHERIT REMOTE PRIVILEGES privilege.
- Example: Revoking the INHERIT REMOTE PRIVILEGES Privilege from PUBLIC

 The REVOKE SQL statement can revoke the INHERIT REMOTE PRIVILEGES from PUBLIC, as well as from individual procedure owners.
- Tutorial: Using a Database Link in a Definer's Rights Procedure
 This tutorial demonstrates how the INHERIT REMOTE PRIVILEGES privilege works in a definer's rights procedure that uses a database link.

9.8.1 About Controlling Definer's Rights Privileges for Database Links

When a definer's rights procedure connects to a database link, operations on the database link should use the procedure owner's credentials.

The INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges apply when a connected user database link is used with a definer's rights procedure. These privileges allow the use of the credentials of the logged-in user for connected user database link operations with definer rights procedures.

You can perform a grant of the INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges so the users who invoke the definer's rights procedure can use a connected user database link within a definer's rights block. A definer's rights procedure executes with the privileges of the procedure owner. However, a connected user database link operation must have the credentials of the logged in user. Hence, the INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges are required to be granted to enable the database link operations within the definer's rights block.

Be aware that during an upgrade, the INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges are not granted by default to any existing users.

The INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges apply only to situations in which users are trying to connect to user database links in a definer's rights procedure. In addition, these privileges apply to both privately created and publicly created database links. By default, database links are created as private links. In addition, by default, INHERIT REMOTE PRIVILEGES is not granted to PUBLIC.

The ways that you can perform grants of these privileges are as follows:

• GRANT INHERIT REMOTE PRIVILEGES ON USER dbuser_1 TO dbuser_2: In this scenario, dbuser_1 can explicitly grant the INHERIT REMOTE PRIVILEGE privilege to dbuser_2 and use a definer's rights procedure that user dbuser 2 owns.



- GRANT INHERIT REMOTE PRIVILEGES ON USER *dbuser_1* TO PUBLIC. In this scenario, dbuser_1 grants the INHERIT REMOTE PRIVILEGE privilege to public. This grant enables dbuser 1 to use the definer's rights procedures that any other user owns.
- GRANT INHERIT ANY REMOTE PRIVILEGES TO *dbuser_2*: In this scenario, any user can use the definer's rights procedures that dbuser_2 owns.

If the user does not have the INHERIT REMOTE PRIVILEGE privilege and tries to execute the definer's rights privilege, then the ORA-25433: User does not have INHERIT REMOTE PRIVILEGES error appears.

9.8.2 Grants of the INHERIT REMOTE PRIVILEGES Privilege to Other Users

The INHERIT REMOTE PRIVILEGES privilege enables the current user to have explicit privileges over the connected user in the database.

The syntax for granting the INHERIT REMOTE PRIVILEGES privilege is as follows:

GRANT INHERIT REMOTE PRIVILEGES ON USER connected user TO current user:

In this specification:

- connected user is the user who runs the definer's rights procedure.
- current_user is the user who owns the definer's right procedure. This value must be a
 database user account. As an alternative to granting the INHERIT REMOTE PRIVILEGES
 privilege to the procedure's owner, you can grant the privilege to a role that is in turn
 granted to the procedure.

Users or roles who own the definer's rights procedures must have the INHERIT REMOTE PRIVILEGES privilege granted to them by users who will run their definer's rights procedures.

Any user can grant or revoke the INHERIT REMOTE PRIVILEGES privilege on themselves to the user whose definer's rights procedures they want to run.

9.8.3 Example: Granting INHERIT REMOTE PRIVILEGES on a Connected User

You can grant the INHERIT REMOTE PRIVILEGES privilege on a connected user to the current user.

In this example, the connected user, jward, must have remote privileges on the current user, ebrown. This enables jward to execute the definer's right procedure that ebrown created.

Example 9-4 shows how an administrator (or user jward) can grant the INHERIT REMOTE PRIVILEGES on user jward to user ebrown. This privilege grant enables any definer's rights procedure that ebrown writes, or will write in the future, to access ebrown's privileges when the procedure is run.

Example 9-4 Granting INHERIT REMOTE PRIVILEGES on a Connected User to the Current User

GRANT INHERIT REMOTE PRIVILEGES on user jward to ebrown;



9.8.4 Grants of the INHERIT ANY REMOTE PRIVILEGES Privilege to Other Users

The INHERIT ANY REMOTE PRIVILEGES privilege enables the grantee user to open a connected user database link as any user.

As with all any privileges, inherit any remote privileges is a powerful privilege that must only be granted to trusted users. By default, user sys has the inherit any remote privileges system privilege with grant option. To find users who have been granted the inherit any remote privileges privilege, query the DBA sys privs data dictionary view.

For better security in a multitenant environment, Oracle recommends that you protect the INHERIT ANY REMOTE PRIVILEGES privilege with a PDB lockdown profile. A PDB lockdown profile prevents local pluggable database (PDB) users from opening a connected user database link as a common user, irrespective of the kind of INHERIT REMOTE PRIVILEGE the PDB user has. If the PDB is protected by a PDB lockdown profile, then grants such as GRANT INHERIT REMOTE PRIVILEGES and GRANT INHERIT ANY REMOTE PRIVILEGES succeed but the effects of these grants do not apply as long as the PDB lockdown continues.

The syntax for granting the INHERIT ANY REMOTE PRIVILEGES privilege is as follows:

GRANT INHERIT ANY REMOTE PRIVILEGES TO current user;

In this specification, current user is the user who owns the define's right procedure.

Related Topics

Restricting Operations on PDBs Using PDB Lockdown Profiles
 You can use PDB lockdown profiles in a multitenant environment to restrict sets of user operations in pluggable databases (PDBs).

9.8.5 Revokes of the INHERIT [ANY] REMOTE PRIVILEGES Privilege

The methods for revoking the INHERIT REMOTE PRIVILEGES and INHERIT ANY REMOTE PRIVILEGES privileges differ.

The INHERIT REMOTE PRIVILEGES privilege can be revoked by a user from another user. The INHERIT ANY REMOTE PRIVILEGES privilege must be revoked by a user with administrative privileges.

The revocation syntax is as follows

REVOKE INHERIT REMOTE PRIVILEGES ON USER connected_user FROM current_user;

In this specification:

- connected user is the user who runs the definer's rights procedure.
- current user is the user who owns the definer's rights procedure.

If you want to revoke the INHERIT REMOTE PRIVILEGES or INHERIT ANY REMOTE PRIVILEGES privilege from a user, use the standard revocation syntax, as follows:

REVOKE INHERIT REMOTE PRIVILEGES FROM connected_user; REVOKE INHERIT ANY REMOTE PRIVILEGES FROM current user;





Oracle Database SQL Language Reference for more information about the REVOKE SQL statement

9.8.6 Example: Revoking the INHERIT REMOTE PRIVILEGES Privilege

The REVOKE SQL statement can revoke the INHERIT REMOTE PRIVILEGES privilege.

After you revoke the INHERIT REMOTE PRIVILEGES privilege, if user jward executes a definer's rights procedure that jward owns, then any operation on a connected user database link inside the definer's rights procedure fails because jward has explicitly denied ebrown the privilege to open a connected user database link using jward'credentials.

Example 9-5 shows how to revoke the INHERIT REMOTE PRIVILEGES procedure on the connecting user, jward, from the procedure owner, ebrown.

Example 9-5 Revoking the INHERIT REMOTE PRIVILEGES Privilege

REVOKE INHERIT REMOTE PRIVILEGES ON USER jward FROM ebrown;

9.8.7 Example: Revoking the INHERIT REMOTE PRIVILEGES Privilege from PUBLIC

The REVOKE SQL statement can revoke the INHERIT REMOTE PRIVILEGES from PUBLIC, as well as from individual procedure owners.

Example 9-6 shows how to revoke this privilege from PUBLIC.

Example 9-6 Revoking the INHERIT REMOTE PRIVILEGES Privilege from PUBLIC

REVOKE INHERIT REMOTE PRIVILEGES FROM PUBLIC;

9.8.8 Tutorial: Using a Database Link in a Definer's Rights Procedure

This tutorial demonstrates how the INHERIT REMOTE PRIVILEGES privilege works in a definer's rights procedure that uses a database link.

- About This Tutorial
 - In this tutorial, you test the privilege grant and revoke of the INHERIT REMOTE PRIVILEGES privilege.
- Step 1: Create User Accounts
 - You must create a user who creates a definer's rights procedure that has a database link, and a second user who executes this procedure.
- Step 2: As User dbuser2, Create a Table to Store User IDs
 The user IDs in this table are the IDs that the database link uses.
- Step 3: As User dbuser1, Create a Database Link and Definer's Rights Procedure User dbuser1 is ready to create a database link and then a definer's rights procedure that references the database link.
- Step 4: Test the Definer's Rights Procedure
 User dbuser2 must grant INHERIT REMOTE PRIVILEGES to dbuser1 before the definer's rights procedure can be tested.

Step 5: Remove the Components of This Tutorial
If you no longer need the components of this tutorial, then you can remove them.

9.8.8.1 About This Tutorial

In this tutorial, you test the privilege grant and revoke of the INHERIT REMOTE PRIVILEGES privilege.

To accomplish this, you must create two users, one who creates a definer's rights procedure that refers to a database link, and a second user to execute this definer's rights procedure. Both users create identical look-up tables in their schemas. The definer's rights procedure must enable the second user to query the lookup table that belongs to the definer's rights users.

9.8.8.2 Step 1: Create User Accounts

You must create a user who creates a definer's rights procedure that has a database link, and a second user who executes this procedure.

1. Connect as a user who has privileges to create users and perform privilege grants.

For example:

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

Create the user accounts as follows:

```
GRANT CONNECT, RESOURCE, UNLIMITED TABLESPACE TO dbuser1 IDENTIFIED BY password; GRANT CONNECT, RESOURCE, UNLIMITED TABLESPACE TO dbuser2 IDENTIFIED BY password;
```

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

9.8.8.3 Step 2: As User dbuser2, Create a Table to Store User IDs

The user IDs in this table are the IDs that the database link uses.

1. Connect as user dbuser2 to instance inst1.

```
connect dbuser2@inst1
Enter password: password
```

2. Create the following table:

```
CREATE TABLE dbusertab(ID NUMBER(2));
```

3. Populate this table with the ID value 10.

```
INSERT INTO dbusertab VALUES(10);
```

9.8.8.4 Step 3: As User dbuser1, Create a Database Link and Definer's Rights Procedure

User <code>dbuser1</code> is ready to create a database link and then a definer's rights procedure that references the database link.

1. Connect as user dbuser1 to instance inst1.

```
connect dbuser1@inst1
Enter password: password
```

2. Create a database link, which will be used in the definer's rights procedure.

```
CREATE DATABASE LINK dblink USING 'inst1';
```

3. Create a dbusertab table and then populate it with the ID 20.

```
CREATE TABLE DBUSERTAB(ID NUMBER(2));
INSERT INTO dbusertab VALUES(20);
```

4. Create a definer's rights procedure that contains a reference to the database lnk

Test the definer's rights procedure.

```
SET SERVEROUTPUT ON
EXEC test_remote_db_link;
```

The output should be as follows, indicating that user <code>dbuser1</code> has executed the procedure on his own version of the table <code>dbusertab</code>:

```
v id : 20
```

6. Grant the user dbuser2 the EXECUTE privilege on the test remote db link procedure.

```
GRANT EXECUTE ON test_remote_db_link TO dbuser2;
```

9.8.8.5 Step 4: Test the Definer's Rights Procedure

User dbuser2 must grant INHERIT REMOTE PRIVILEGES to dbuser1 before the definer's rights procedure can be tested.

1. Connect as user dbuser2 to instance inst1.

```
connect dbuser2@inst1
Enter password: password
```

2. Grant the INHERIT REMOTE PRIVILEGE privilege on user dbuser2 to dbuser1.

```
GRANT INHERIT REMOTE PRIVILEGES ON user dbuser2 TO dbuser1;
```

Relog back in, because the grant does not take effect until you start a new session.

```
connect dbuser2@inst1
Enter password: password
```

4. Execute the test_remote db link definer's rights procedure:

```
SET SERVEROUTPUT ON
EXEC dbuser1.test_remote_db_link;
```

The output shows the following, which indicates that user <code>dbuser1</code> is able to use the database link to connect to the schema of <code>dbuser2</code> and access the values in the <code>dbusertab</code> table in <code>dbuser2</code>'s schema.

```
v_id : 10
```



5. Revoke the INHERIT REMOTE PRIVILEGE privilege on dbuser2 from dbuser1.

REVOKE INHERIT REMOTE PRIVILEGES ON USER dbuser2 FROM dbuser1;

6. Try executing the test remote db link definer's rights procedure again.

```
EXEC dbuser1.test_remote_db_link;
```

The ORA-25433: User DBUSER1 does not have INHERIT REMOTE PRIVILEGES on connected user DBUSER2 error should appear.

9.8.8.6 Step 5: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

 Connect as a user who has privileges to drop user accounts and database links For example:

```
connect sec_admin
Enter password: password
```

2. Drop the user accounts.

```
DROP USER dbuser1 CASCADE;
DROP USER dbuser2 CASCADE;
```

3. Drop the dblink database link.

DROP PUBLIC DATABASE LINK dblink;

Managing Fine-Grained Access in PL/ SQL Packages and Types

Oracle Database provides PL/SQL packages and types for fine-grained access to control access to external network services and wallets.

- About Managing Fine-Grained Access in PL/SQL Packages and Types
 You can configure user access to external network services and wallets through a set of
 PL/SQL packages and one type.
- About Fine-Grained Access Control to External Network Services
 Oracle Application Security access control lists (ACL) can implement fine-grained access control to external network services.
- About Access Control to Oracle Wallets
 When accessing remote Web server-protected Web pages, users can authenticate themselves with passwords and client certificates stored in an Oracle wallet.
- Upgraded Applications That Depend on Packages That Use External Network Services
 Upgraded applications may have ORA-24247 network access errors.
- Configuring Access Control for External Network Services
 The DBMS_NETWORK_ACL packages configures access control for external network services.
- Configuring Access Control to an Oracle Wallet
 Fine-grained access control for Oracle wallets provide user access to network services that require passwords or certificates.
- Examples of Configuring Access Control for External Network Services
 You can configure access control for a variety of situations, such as for a single role and network connection.
- Specifying a Group of Network Host Computers
 You can use wildcards to specify a group of network host computers.
- Precedence Order for a Host Computer in Multiple Access Control List Assignments
 The access control list assigned to a domain has a lower precedence than those assigned to the subdomains.
- Precedence Order for a Host in Access Control List Assignments with Port Ranges
 The precedence order for a host in an access control list is determined by the use of port
 ranges.
- Checking Privilege Assignments That Affect User Access to Network Hosts Both administrators and users can check network connection and domain privileges.
- Configuring Network Access for Java Debug Wire Protocol Operations
 Before you can debug Java PL/SQL procedures, you must be granted the jdwp ACL privilege.
- Data Dictionary Views for Access Control Lists Configured for User Access
 Oracle Database provides data data dictionary views that you can use to find information about existing access control lists.

10.1 About Managing Fine-Grained Access in PL/SQL Packages and Types

You can configure user access to external network services and wallets through a set of PL/SQL packages and one type.

These packages are the <code>UTL_TCP</code>, <code>UTL_SMTP</code>, <code>UTL_MAIL</code>, <code>UTL_HTTP</code>, and <code>UTL_INADDR</code>, and the <code>DBMS_LDAP_PL/SQL</code> packages, and the <code>HttpUriType</code> type.

The following scenarios are possible:

- Configuring fine-grained access control for users and roles that need to access external network services from the database. This way, specific groups of users can connect to one or more host computers, based on privileges that you grant them. Typically, you use this feature to control access to applications that run on specific host addresses.
- Configuring fine-grained access control to Oracle wallets to make HTTP requests that require password or client-certificate authentication. This feature enables you to grant privileges to users who are using passwords and client certificates stored in Oracle wallets to access external protected HTTP resources through the UTL_HTTP package. For example, you can configure applications to use the credentials stored in the wallets instead of hard-coding the credentials in the applications.

10.2 About Fine-Grained Access Control to External Network Services

Oracle Application Security access control lists (ACL) can implement fine-grained access control to external network services.

This guide explains how to configure the access control for database users and roles by using the DBMS NETWORK ACL ADMIN PL/SQL package.

This feature enhances security for network connections because it restricts the external network hosts that a database user can connect to using the PL/SQL network utility packages UTL_TCP, UTL_SMTP, UTL_MAIL, UTL_HTTP, and UTL_INADDR; the DBMS_LDAP and DBMS_DEBUG_JDWP PL/SQL packages; and the HttpUriType type. Otherwise, an intruder who gained access to the database could maliciously attack the network, because, by default, the PL/SQL utility packages are created with the EXECUTE privilege granted to PUBLIC users. These PL/SQL network utility packages, and the DBMS_NETWORK_ACL_ADMIN and DBMS_NETWORK_ACL_UTILITY packages, support both IP Version 4 (IPv4) and IP Version 6 (IPv6) addresses. This guide explains how to manage access control to both versions. For detailed information about how the IPv4 and IPv6 notation works with Oracle Database, see Oracle Database Net Services Administrator's Guide.

Be aware that outbound Secure Sockets Layer (SSL) connections with $\mathtt{UTL}_\mathtt{HTTP}$ cannot use the default trust store. You must create an Oracle wallet to hold the trust certificates.



See Also:

Tutorial: Adding an Email Alert to a Fine-Grained Audit Policy for an example of configuring access control to external network services for email alerts

10.3 About Access Control to Oracle Wallets

When accessing remote Web server-protected Web pages, users can authenticate themselves with passwords and client certificates stored in an Oracle wallet.

The Oracle wallet provides secure storage of user passwords and client certificates.

To configure access control to a wallet, you must have the following components:

- An Oracle wallet. You can create the wallet using the Oracle Database mkstore utility or
 Oracle Wallet Manager. The HTTP request will use the external password store or the
 client certificate in the wallet to authenticate the user
- An access control list to grant privileges to the user to use the wallet. To configure
 the access control list, you use the DBMS_NETWORK_ACL_ADMIN PL/SQL package.

The use of Oracle wallets is beneficial because it provides secure storage of passwords and client certificates necessary to access protected Web pages.

Related Topics

Configuring Access Control to an Oracle Wallet
 Fine-grained access control for Oracle wallets provide user access to network services that require passwords or certificates.

10.4 Upgraded Applications That Depend on Packages That Use External Network Services

Upgraded applications may have ORA-24247 network access errors.

If you have upgraded from a release before Oracle Database 11g Release 1 (11.1), and your applications depend on PL/SQL network utility packages (UTL_TCP, UTL_SMTP, UTL_MAIL, UTL_HTTP, UTL_INADDR, and DBMS_LDAP) or the HttpUriType type, then the ORA-24247 error may occur when you try to run the application.

The error message is as follows:

ORA-24247: network access denied by access control list (ACL)

Use the procedures in this chapter to reconfigure the network access for the application.

See Also:

Oracle Database Upgrade Guide for compatibility issues for applications that depend on the PL/SQL network utility packages



10.5 Configuring Access Control for External Network Services

The DBMS NETWORK ACL packages configures access control for external network services.

- Syntax for Configuring Access Control for External Network Services
 You can use the DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE procedure to grant the
 access control privileges to a user.
- Enabling the Listener to Recognize Access Control for External Network Services
 A TNS-01166: Listener rejected registration or update of service ACL error can
 result if the listener is not configured to recognize access control for external network
 services.
- Example: Configuring Access Control for External Network Services
 The DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE procedure can configure access control for external network services.
- Revoking Access Control Privileges for External Network Services
 You can remove access control privileges for external network services.
- Example: Revoking External Network Services Privileges
 The DBMS_NETWORK_ACL_ADMIN.REMOVE_HOST_ACE procedure can be used to revoke external network privileges.

10.5.1 Syntax for Configuring Access Control for External Network Services

You can use the <code>DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE</code> procedure to grant the access control privileges to a user.

This procedure appends an access control entry (ACE) with the specified privilege to the ACL for the given host, and creates the ACL if it does not exist yet. The resultant configuration resides in the SYS schema, not the schema of the user who created it.

The syntax is as follows:

In this specification:

host: Enter the name of the host. It can be the host name or an IP address of the host. You can use a wildcard to specify a domain or an IP subnet. (See Precedence Order for a Host Computer in Multiple Access Control List Assignments for the precedence order when you use wildcards in domain names.) The host or domain name is case insensitive. Examples are as follows:

```
host => 'www.example.com',
host => '*example.com',
```

lower_port: (Optional) For TCP connections, enter the lower boundary of the port range.
Use this setting for the connect privilege only. Omit it for the resolve privilege. The default
is null, which means that there is no port restriction (that is, the ACL applies to all ports).
The range of port numbers is between 1 and 65535.

For example:

```
lower port => 80,
```

• upper_port: (Optional) For TCP connections, enter the upper boundary of the port range. Use this setting for connect privileges only. Omit it for the resolve privilege. The default is null, which means that there is no port restriction (that is, the ACL applies to all ports). The range of port numbers is between 1 and 65535

For example:

```
upper port => 3999);
```

If you enter a value for the <code>lower_port</code> and <code>leave</code> the <code>upper_port</code> at <code>null</code> (or just omit it), then Oracle Database assumes the <code>upper_port</code> setting is the same as the <code>lower_port</code>. For example, if you set <code>lower_port</code> to 80 and omit <code>upper_port</code>, the <code>upper_port</code> setting is assumed to be 80.

The resolve privilege in the access control list has no effect when a port range is specified in the access control list assignment.

• ace: Define the ACE by using the XS\$ACE TYPE constant, in the following format:

In this specification:

 privilege_list: Enter one or more of the following privileges, which are case insensitive. Enclose each privilege with single quotation marks and separate each with a comma (for example, 'http', 'http proxy').

For tighter access control, grant only the http_proxy, or Http">Http"UTL_SMTP, or UTL_SMTP, o

- http: Makes an HTTP request to a host through the UTL_HTTP package and the HttpUriType type
- http_proxy: Makes an HTTP request through a proxy through the UTL_HTTP package and the HttpUriType type. You must include http_proxy in conjunction to the http privilege if the user makes the HTTP request through a proxy.
- smtp: Sends SMTP to a host through the UTL SMTP and UTL MAIL packages
- resolve: Resolves a network host name or IP address through the utl_{INADDR} package
- connect: Grants the user permission to connect to a network service at a host through the <code>UTL_TCP</code>, <code>UTL_SMTP</code>, <code>UTL_MAIL</code>, <code>UTL_HTTP</code>, and <code>DBMS_LDAP</code> packages, or the <code>HttpUriType</code> type
- jdwp: Used for Java Debug Wire Protocol debugging operations for Java or PL/SQL stored procedures. See Configuring Network Access for Java Debug Wire Protocol Operations for more information.
- principal_name: Enter a database user name or role. This value is case insensistive, unless you enter it in double quotation marks (for example, '"ACCT_MGR'").
- principal_type: Enter XS_ACL.PTYPE_DB for a database user or role. You must specify
 PTYPE_DB because the principal_type value defaults to PTYPE_XS, which is used to specify an Oracle Database Real Application Security application user.



See Also:

Oracle Database Real Application Security Administrator's and Developer's Guide for information about additional XS\$ACE_TYPE parameters that you can include for the ace parameter setting: granted, inverted, start_date, and end_date

10.5.2 Enabling the Listener to Recognize Access Control for External Network Services

A TNS-01166: Listener rejected registration or update of service ACL error can result if the listener is not configured to recognize access control for external network services.

1. Add the following line to the listener.ora file:

```
LOCAL_REGISTRATION_ADDRESS_LISTENER = ON
```

2. Restart the listener.

```
./lsnrctl stop
./lsnrctl start
```

10.5.3 Example: Configuring Access Control for External Network Services

The DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE procedure can configure access control for external network services.

Example 10-1 shows how to grant the http and smtp privileges to the acct_mgr database role for an ACL created for the host www.example.com.

Example 10-1 Granting Privileges to a Database Role External Network Services

10.5.4 Revoking Access Control Privileges for External Network Services

You can remove access control privileges for external network services.

• To revoke access control privileges for external network services, run the DBMS_NETWORK_ACL_ADMIN.REMOVE_HOST_ACE procedure.

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS NETWORK ACL ADMIN.REMOVE HOST ACE procedure

10.5.5 Example: Revoking External Network Services Privileges

The DBMS_NETWORK_ACL_ADMIN.REMOVE_HOST_ACE procedure can be used to revoke external network privileges.

Example 10-2 shows how to revoke external network privileges.

Example 10-2 Revoking External Network Services Privileges

In this specification, the \mathtt{TRUE} setting for $\mathtt{remove_empty_acl}$ removes the ACL when it becomes empty when the ACE is removed.

10.6 Configuring Access Control to an Oracle Wallet

Fine-grained access control for Oracle wallets provide user access to network services that require passwords or certificates.

- About Configuring Access Control to an Oracle Wallet
 You can configure access control to grant access to passwords and client certificates.
- Step 1: Create an Oracle Wallet
 An Oracle wallet can use both standard and PKCS11 wallet types, as well as being an auto-login wallet.
- Step 2: Configure Access Control Privileges for the Oracle Wallet
 After you have created the wallet, you are ready to configure access control privileges for
 the wallet.
- Step 3: Make the HTTP Request with the Passwords and Client Certificates
 The UTL_HTTP package can create an HTTP request object to hold wallet information,
 which can authenticate using a client certificate or a password.
- Revoking Access Control Privileges for Oracle Wallets
 You can revoke access control privileges for an Oracle wallet.
- Troubleshooting ORA-29024 Errors
 The ORA-29024: Certificate validation failure error occurs when the facility, component, or product or a failing operation is expecting an Oracle wallet.

10.6.1 About Configuring Access Control to an Oracle Wallet

You can configure access control to grant access to passwords and client certificates.

These passwords and client certificates are stored in an Oracle wallet. The access control that you configure enables users to authenticate themselves to an external network service when using the PL/SQL network utility packages.

This enables the user to gain access to the network service that requires password or certificate identification.

10.6.2 Step 1: Create an Oracle Wallet

An Oracle wallet can use both standard and PKCS11 wallet types, as well as being an autologin wallet.

 To create the wallet, use either the mkstore command-line utility or the Oracle Wallet Manager user interface.

To store passwords in the wallet, you must use the mkstore utility.

- 2. Ensure that you have exported the wallet to a file.
- 3. Make a note of the directory in which you created the wallet. You will need this directory path when you complete the procedures in this section.

Related Topics

- Example: Configuring ACL Access Using Passwords in a Non-Shared Wallet
 The DBMS_NETWORK_ACL_ADMIN and UTL_HTTP PL/SQL packages can configure ACL access using passwords in a non-shared wallet.
- Example: Configuring ACL Access for a Wallet in a Shared Database Session
 The DBMS_NETWORK_ACL_ADMIN and UTL_HTTP PL/SQL packages can configure ACL access for a wallet in a shared database session.

10.6.3 Step 2: Configure Access Control Privileges for the Oracle Wallet

After you have created the wallet, you are ready to configure access control privileges for the wallet.

 Use the DBMS_NETWORK_ACL_ADMIN.APPEND_WALLET_ACE procedure to configure the wallet access control privileges.

The syntax for the DBMS NETWORK ACL ADMIN.APPEND WALLET ACE procedure is as follows:

In this specification:

 wallet_path: Enter the path to the directory that contains the wallet that you created in Step 1: Create an Oracle Wallet. When you specify the wallet path, you must use an absolute path and include file: before this directory path. Do not use environment variables, such as precent in the sum of t

```
wallet_path => 'file:/oracle/wallets/hr_wallet',
```

ace: Define the ACL by using the XS\$ACE_TYPE constant. For example:

In this specification, *privilege* must be one of the following when you enter wallet privileges using xs\$ace type (note the use of underscores in these privilege names):

- use client certificates
- use_passwords

For detailed information about these parameters, see the ace parameter description in Syntax for Configuring Access Control for External Network Services. Be aware that for wallets, you must specify either the use_client_certificates or use_passwords privileges.

See Also:

Oracle Database Real Application Security Administrator's and Developer's Guide for information about additional XS\$ACE_TYPE parameters that you can include for the ace parameter setting: granted, inverted, start date, and end date

10.6.4 Step 3: Make the HTTP Request with the Passwords and Client Certificates

The UTL_HTTP package can create an HTTP request object to hold wallet information, which can authenticate using a client certificate or a password.

- Making the HTTPS Request with the Passwords and Client Certificates
 The UTL_HTTP package makes Hypertext Transfer Protocol (HTTP) callouts from SQL and PL/SQL.
- Using a Request Context to Hold the Wallet When Sharing the Session with Other Applications

You should use a request context to hold the wallet when other applications share the database session.

Use of Only a Client Certificate to Authenticate

Only a client certificate can authenticate users, as long as the user has been granted the appropriate privilege in the ACL wallet.

Use of a Password to Authenticate

If the protected URL being requested requires username and password authentication, then set the username and password from the wallet to authenticate.



10.6.4.1 Making the HTTPS Request with the Passwords and Client Certificates

The $\tt UTL_HTTP$ package makes Hypertext Transfer Protocol (HTTP) callouts from SQL and PL/SQL.

Use the UTL_HTTP PL/SQL package to create a request context object that is used privately
with the HTTP request and its response.

For example:

In this specification:

- req_context: Use the UTL_HTTP.CREATE_REQUEST_CONTEXT_KEY data type to create the
 request context object. This object stores a randomly-generated numeric key that Oracle
 Database uses to identify the request context. The UTL_HTTP.CREATE_REQUEST_CONTEXT
 function creates the request context itself.
- req: Use the UTL_HTTP.REQ data type to create the object that will be used to begin the HTTP request. You will refer to this object later on, when you set the user name and password from the wallet to access a password-protected Web page.
- wallet_path: Enter the path to the directory that contains the wallet. Ensure that this path
 is the same path you specified when you created access control list in Step 2: Configure
 Access Control Privileges for the Oracle Wallet in the previous section. You must include
 file: before the directory path. Do not use environment variables, such as \$ORACLE HOME.

For example:

```
wallet_path => 'file:/oracle/wallets/hr_wallet',
```

wallet_password: Enter the password used to open the wallet. The default is NULL, which
is used for auto-login wallets. For example:

```
wallet password => 'wallet password');
```

url: Enter the URL to the application that uses the wallet.

For example:

```
url => 'www.hr_access.example.com',
```

 request_context: Enter the name of the request context object that you created earlier in this section. This object prevents the wallet from being shared with other applications in the same database session.

For example:

```
request context => req context);
```



See Also:

Oracle Database PL/SQL Packages and Types Reference for detailed information about the \mathtt{UTL} \mathtt{HTTP} package

10.6.4.2 Using a Request Context to Hold the Wallet When Sharing the Session with Other Applications

You should use a request context to hold the wallet when other applications share the database session.

If your application has exclusive use of the database session, you can hold the wallet in the database session by using the $\verb"UTL HTTP.SET"$ WALLET procedure.

Use the UTL HTTP.SET WALLET procedure to configure the request to hold the wallet.

For example:

If the protected URL being requested requires the user name and password to authenticate, then you can use the <code>SET_AUTHENTICATION_FROM_WALLET</code> procedure to set the user name and password from the wallet to authenticate.

10.6.4.3 Use of Only a Client Certificate to Authenticate

Only a client certificate can authenticate users, as long as the user has been granted the appropriate privilege in the ACL wallet.

If the protected URL being requested requires only the client certificate to authenticate, then the BEGIN_REQUEST function sends the necessary client certificate from the wallet. assuming the user has been granted the use_client_certificates privilege in the ACL assigned to the wallet.

The authentication should succeed at the remote Web server and the user can proceed to retrieve the HTTP response by using the GET RESPONSE function.

10.6.4.4 Use of a Password to Authenticate

If the protected URL being requested requires username and password authentication, then set the username and password from the wallet to authenticate.

For example:

```
DECLARE

req_context UTL_HTTP.REQUEST_CONTEXT_KEY;

req UTL_HTTP.REQ;

BEGIN
```



In this specification:

• r: Enter the HTTP request defined in the <code>UTL_HTTP.BEGIN_REQUEST</code> procedure that you created above, in the previous section. For example:

```
r \Rightarrow req_{r}
```

 alias: Enter the alias used to identify and retrieve the user name and password credential stored in the Oracle wallet. For example, assuming the alias used to identify this user name and password credential is hr access.

```
alias => 'hr_access',
```

- scheme: Enter one of the following:
 - AWS: Specifies the Amazon Simple Storage Service (S3) scheme. Use this scheme
 only if you are configuring access to the Amazon.com Web site. (Contact Amazon for
 more information about this setting.)
 - Basic: Specifies HTTP basic authentication. The default is Basic.

For example:

```
scheme => 'Basic',
```

 for_proxy: Specify whether the HTTP authentication information is for access to the HTTP proxy server instead of the Web server. The default is FALSE.

For example:

```
for proxy => TRUE);
```

The use of the user name and password in the wallet requires the use_passwords privilege to be granted to the user in the ACL assigned to the wallet.

10.6.5 Revoking Access Control Privileges for Oracle Wallets

You can revoke access control privileges for an Oracle wallet.

To revoke privileges from access control entries (ACE) in the access control list (ACL) of a
wallet, run the DBMS NETWORK ACL ADMIN.REMOVE WALLET ACE procedure.

For example:

In this example, the TRUE setting for remove_empty_acl removes the ACL when it becomes empty when the wallet ACE is removed.

10.6.6 Troubleshooting ORA-29024 Errors

The ORA-29024: Certificate validation failure error occurs when the facility, component, or product or a failing operation is expecting an Oracle wallet.

You can troubleshoot this error by using the following methods, in this order:

 Check is the relevant Oracle documentation for the steps related to the failing configuration.

For example, if this error is occurs while using UTL_HTTP, then it means that a secure web site is being accessed without a wallet and this operation needs a wallet created. See *Oracle Database PL/SQL Packages and Types Reference* for information about using the UTL HTTP PL/SQL package.

In another example, the error can occur while making a remote connection to the database server over a TLS connection, which indicates that this connection is expecting an Oracle wallet. Troubleshooting this problem requires a proper understanding of Oracle Wallets and certificates. See Configuring Transport Layer Security Authentication.

- 2. After the wallet is configured according to the documentation, if the error still occurs, then try the following solutions:
 - Open the wallet using the orapki utility as follows:

```
orapki wallet display -wallet wallet file directory
```

If this command fails, then it means that the wallet is corrupt. Create a new wallet and recheck the scenario.

- If the current configuration needs a wallet with a user and trusted certificates, then
 check whether both the user and trusted certificates are valid and not expired or
 revoked.
- If this error occurs while using the wallet with a UTL_HTTP configuration, then check whether all the certificates of the secure web site are there in the wallet and the certificate chain is complete.
- If there is a proxy server involved, then ensure that the target website is in the proxy allowlist.

See the following My Oracle Support notes for information about getting a complete certificate chain of a secure site for a $\tt UTL_HTTPS$ call.

- Note 169768.1 Configuring Wallet Manager to enable HTTPS connections via UTL_HTTP.REQUEST
- Note 230917.1 Troubleshooting the UTL_HTTP Package

10.7 Examples of Configuring Access Control for External Network Services

You can configure access control for a variety of situations, such as for a single role and network connection.

Example: Configuring Access Control for a Single Role and Network Connection
 The DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE procedure can configure access control for a single role and network connection.



- Example: Configuring Access Control for a User and Role

 The DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE can configure access control to deny or grant privileges for a user and a role.
- Example: Using the DBA_HOST_ACES View to Show Granted Privileges

 The DBA HOST ACE data dictionary view shows privileges that have been granted to users.
- Example: Configuring ACL Access Using Passwords in a Non-Shared Wallet
 The DBMS_NETWORK_ACL_ADMIN and UTL_HTTP PL/SQL packages can configure ACL access
 using passwords in a non-shared wallet.
- Example: Configuring ACL Access for a Wallet in a Shared Database Session
 The DBMS_NETWORK_ACL_ADMIN and UTL_HTTP PL/SQL packages can configure ACL access for a wallet in a shared database session.

10.7.1 Example: Configuring Access Control for a Single Role and Network Connection

The DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE procedure can configure access control for a single role and network connection.

Example 10-3 shows how you would configure access control for a single role (acct_mgr) and grant this role the http privilege for access to the www.us.example.com host. The privilege expires January 1, 2013.

Example 10-3 Configuring Access Control for a Single Role and Network Connection

10.7.2 Example: Configuring Access Control for a User and Role

The DBMS_NETWORK_ACL_ADMIN.APPEND_HOST_ACE can configure access control to deny or grant privileges for a user and a role.

Afterwards, you can query the DBA_HOST_ACES data dictionary view to find information about the privilege grants.

Example 10-4 grants to a database role (acct_mgr) but denies a particular user (psmith) even if he has the role. The order is important because ACEs are evaluated in the given order. In this case, the deny ACE (granted => false) must be appended first or else the user cannot be denied.

Example 10-4 Configuring Access Control Using a Grant and a Deny for User and Role

10.7.3 Example: Using the DBA_HOST_ACES View to Show Granted Privileges

The DBA HOST ACE data dictionary view shows privileges that have been granted to users.

Example 10-5 shows how the DBA_HOST_ACES data dictionary view displays the privilege granted in the previous access control list.

Example 10-5 Using the DBA_HOST_ACES View to Show Granted Privileges

10.7.4 Example: Configuring ACL Access Using Passwords in a Non-Shared Wallet

The DBMS_NETWORK_ACL_ADMIN and UTL_HTTP PL/SQL packages can configure ACL access using passwords in a non-shared wallet.

Example 10-6 configures wallet access for two Human Resources department roles, hr_clerk and $hr_manager$. These roles use the use_passwords privilege to access passwords stored in the wallet. In this example, the wallet will not be shared with other applications within the same database session.

Example 10-6 Configuring ACL Access Using Passwords in a Non-Shared Wallet

```
/* 1. At a command prompt, create the wallet. The following example uses the
    user name hr_access as the alias to identify the user name and password
    stored in the wallet. You must use this alias name when you call the
    SET_AUTHENTICATION_FROM_WALLET procedure later on. */
$ mkstore -wrl $ORACLE_HOME/wallets/hr_wallet -create
```

```
$ mkstore -wrl $ORACLE_HOME/wallets/hr_wallet -create
Enter password: password
Enter password again: password
$ mkstore -wrl $ORACLE_HOME/wallets/hr_wallet -createCredential hr_access hr_usr
Your secret/Password is missing in the command line
Enter your secret/Password: password
Re-enter your secret/Password: password
Enter wallet password: password
```

/* 2. In SQL*Plus, create an access control list to grant privileges for the wallet. The following example grants the use_passwords privilege to the

```
hr_clerk role.*/
BEGIN
 DBMS NETWORK_ACL_ADMIN.APPEND_WALLET_ACE (
 wallet path => 'file:/oracle/wallets/hr wallet',
             => xs$ace_type(privilege_list => xs$name_list('use_passwords'),
                             principal name => 'hr clerk',
                             principal type => xs acl.ptype db));
END;
/* 3. Create a request context and request object, and then set the authentication
     for the wallet. */
DECLARE
 req context UTL HTTP.REQUEST CONTEXT KEY;
            UTL HTTP.REQ;
BEGIN
req context := UTL HTTP.CREATE REQUEST CONTEXT(
    wallet_path => 'file:/oracle/wallets/hr_wallet',
    wallet_password => NULL,
enable_cookies => TRUE,
max_cookies => 300,
    max cookies per site => 20);
  req := UTL HTTP.BEGIN REQUEST(
    url
                        => 'www.hr_access.example.com',
                       => req_context);
    request context
 UTL HTTP.SET AUTHENTICATION FROM WALLET (
    r
                  => req,
                        => 'hr_access'),
    alias
    scheme
                         => 'Basic',
    for proxy
                      => FALSE);
END;
```

10.7.5 Example: Configuring ACL Access for a Wallet in a Shared Database Session

The <code>DBMS_NETWORK_ACL_ADMIN</code> and <code>UTL_HTTP PL/SQL</code> packages can configure ACL access for a wallet in a shared database session.

Example 10-7 configures the wallet to be used for a shared database session; that is, all applications within the current database session will have access to this wallet.

Example 10-7 Configuring ACL Access for a Wallet in a Shared Database Session

```
=> xs$ace type(privilege list => xs$name list('use client certificates'),
 ace
                         principal name => 'hr mgr',
                          principal_type => xs_acl.ptype_db));
END;
COMMIT;
/* 3. Create a request object to handle the HTTP authentication for the wallet.*/
DECLARE
 req UTL HTTP.req;
BEGIN
 UTL_HTTP.SET_WALLET(
  path => 'file:/oracle/wallets/hr_wallet',
password => NULL);
req := UTL HTTP.BEGIN REQUEST(
  http version => NULL,
  request context => NULL);
```

10.8 Specifying a Group of Network Host Computers

You can use wildcards to specify a group of network host computers.

To assign an access control list to a group of network host computers, use the asterisk (*) wildcard character.

For example, enter *.example.com for host computers that belong to a domain or 192.0.2.* for IPv4 addresses that belong to an IP subnet. The asterisk wildcard must be at the beginning, before a period (.) in a domain, or at the end, after a period (.), in an IP subnet. For example, *.example.com is valid, but *example.com and *.example.* are not. Be aware that the use of wildcard characters affects the order of precedence for multiple access control lists that are assigned to the same host computer. You cannot use wildcard characters for IPv6 addresses.

The Classless Inter-Domain Routing (CIDR) notation defines how IPv4 and IPv6 addresses are categorized for routing IP packets on the internet. The <code>DBMS_NETWORK_ACL_ADMIN</code> package supports CIDR notation for both IPv4 and IPv6 addresses. This package considers an IPv4-mapped IPv6 address or subnet equivalent to the IPv4-native address or subnet it represents. For example, <code>::ffff:192.0.2.1</code> is equivalent to <code>192.0.2.1</code>, and <code>::ffff:192.0.2.1/120</code> is equivalent to <code>192.0.2.*</code>.

10.9 Precedence Order for a Host Computer in Multiple Access Control List Assignments

The access control list assigned to a domain has a lower precedence than those assigned to the subdomains.

For multiple access control lists that are assigned to the host computer and its domains, the access control list that is assigned to the host computer takes precedence over those assigned to the domains.

The access control list assigned to a domain has a lower precedence than those assigned to the subdomains. For example, Oracle Database first selects the access control list assigned to the host <code>server.us.example.com</code>, ahead of other access control lists assigned to its domains.

If additional access control lists were assigned to the sub domains, their order of precedence is as follows:

```
    server.us.example.com
    *.us.example.com
    *.example.com
    *.com
```

Similarly, for multiple access control lists that are assigned to the IP address (both IPv4 and IPv6) and the subnets it belongs to, the access control list that is assigned to the IP address takes precedence over those assigned to the subnets. The access control list assigned to a subnet has a lower precedence than those assigned to the smaller subnets it contains.

For example, Oracle Database first selects the access control list assigned to the IP address 192.0.2.3, ahead of other access control lists assigned to the subnets it belongs to. If additional access control lists were assigned to the subnets, their order of precedence is as follows:

```
1. 192.0.2.3 (or ::ffff:192.0.2.3)
  192.0.2.3/31 (or ::ffff:192.0.2.3/127)
  192.0.2.3/30 (or ::ffff:192.0.2.3/126)
  192.0.2.3/29 (or ::ffff:192.0.2.3/125)
5.
  192.0.2.3/24 (or ::ffff:192.0.2.3/120 or 192.0.2.*)
7.
  . . .
  192.0.2.3/16 (or ::fffff:192.0.2.3/112 or 192.0.*)
9.
10. 192.0.2.3/8 (or ::ffff:192.0.2.3/104 or 192.*)
11. ...
12. ::ffff:192.0.2.3/95
13. ::ffff:192.0.2.3/94
14. ...
15. *
```

10.10 Precedence Order for a Host in Access Control List Assignments with Port Ranges

The precedence order for a host in an access control list is determined by the use of port ranges.

When an access control list is assigned to a host computer, a domain, or an IP subnet with a port range, it takes precedence over the access control list assigned to the same host, domain, or IP subnet without a port range.

For example, suppose you have TCP connections to any port between port 80 and 99 at server.us.example.com. Oracle Database first selects the access control list assigned to port 80 through 99 at server.us.example.com, ahead of the other access control list assigned to server.us.example.com that is without a port range.

10.11 Checking Privilege Assignments That Affect User Access to Network Hosts

Both administrators and users can check network connection and domain privileges.

- About Privilege Assignments that Affect User Access to Network Hosts
 Oracle provides DBA-specific data dictionary views to find information about privilege assignments.
- How to Check User Network Connection and Domain Privileges
 A database administrator can query the DBA_HOST_ACES data dictionary view to find the privileges that have been granted for specific users or roles.
- Example: Administrator Checking User Network Access Control Permissions
 The DBA_HOST_ACES data dictionary view can check the network access control permissions for users.
- How Users Can Check Their Network Connection and Domain Privileges
 Users can query the USER_HOST_ACES data dictionary view to check their network and
 domain permissions.
- Example: User Checking Network Access Control Permissions
 The USER_HOST_ACES data dictionary view shows network access control permissions for a host computer.

10.11.1 About Privilege Assignments that Affect User Access to Network Hosts

Oracle provides DBA-specific data dictionary views to find information about privilege assignments.

Database administrators can use the DBA_HOST_ACES data dictionary view to query network privileges that have been granted to or denied from database users and roles in the access control lists, and whether those privileges take effect during certain times only

Using the information provided by the view, you may need to combine the data to determine if a user is granted the privilege at the current time, the roles the user has, the order of the access control entries, and so on.

Users without database administrator privileges do not have the privilege to access the access control lists or to invoke those <code>DBMS_NETWORK_ACL_ADMIN</code> functions. However, they can query the <code>USER_HOST_ACES</code> data dictionary view to check their privileges instead.

Database administrators and users can use the following <code>DBMS_NETWORK_ACL_UTILITY</code> functions to determine if two hosts, domains, or subnets are equivalent, or if a host, domain, or subnet is equal to or contained in another host, domain, or subnet:

- EQUALS HOST: Returns a value to indicate if two hosts, domains, or subnets are equivalent
- CONTAINS_HOST: Returns a value to indicate if a host, domain, or subnet is equal to or
 contained in another host, domain, or subnet, and the relative order of precedence of the
 containing domain or subnet for its ACL assignments



If you do not use IPv6 addresses, database administrators and users can use the following <code>DBMS_NETWORK_ACL_UTILITY</code> functions to generate the list of domains or IPv4 subnet a host belongs to and to sort the access control lists by their order of precedence according to their host assignments:

- DOMAINS: Returns a list of the domains or IP subnets whose access control lists may affect permissions to a specified network host, subdomain, or IP subnet
- DOMAIN LEVEL: Returns the domain level of a given host

10.11.2 How to Check User Network Connection and Domain Privileges

A database administrator can query the DBA_HOST_ACES data dictionary view to find the privileges that have been granted for specific users or roles.

The DBA_HOST_ACES view shows the access control lists that determine the access to the network connection or domain, and then determines if each access control list grants (GRANTED), denies (DENIED), or does not apply (NULL) to the access privilege of the user. Only the database administrator can guery this view.

10.11.3 Example: Administrator Checking User Network Access Control Permissions

The DBA_HOST_ACES data dictionary view can check the network access control permissions for users.

Example 10-8 shows how a database administrator can check the privileges for user preston to connect to www.us.example.com.

In this example, user preston was granted privileges for all the network host connections found for www.us.example.com. However, suppose preston had been granted access to a host connection on port 80, but then denied access to the host connections on ports 3000–3999. In this case, you must configure access control for the host connection on port 80, and a separate access control configuration for the host connection on ports 3000–3999.

Example 10-8 Administrator Checking User Network Access Control Permissions

```
SELECT HOST, LOWER PORT, UPPER PORT,
      ACE ORDER, PRINCIPAL, PRINCIPAL TYPE,
      GRANT TYPE, INVERTED PRINCIPAL, PRIVILEGE,
     START DATE, END DATE
 FROM (SELECT ACES.*,
DBMS NETWORK ACL UTILITY.CONTAINS_HOST('www.us.example.com', HOST) PRECEDENCE
       FROM DBA HOST ACES ACES)
WHERE PRECEDENCE IS NOT NULL
ORDER BY PRECEDENCE DESC,
       LOWER PORT NULLS LAST,
        UPPER PORT NULLS LAST,
        ACE ORDER;
        LOWER PORT UPPER PORT ACE ORDER PRINCIPAL PRINCIPAL TYPE GRANT TYPE
INVERTED PRINCIPAL PRIVILEGE START DATE END DATE
www.us.example.com 80 80 1 PRESTON DATABASE USER
NO HTTP
                                                                     GRANT
NO HTTP
www.us.example.com 80 80
                                          2 SEBASTIAN DATABASE USER
                                                                    GRANT
*.us.example.com
                                           1 ACCT MGR DATABASE USER GRANT
```



NO	CONNECT	
*		1 HR_DBA DATABASE USER GRANT
NO	CONNECT	
*		1 HR_DBA DATABASE USER GRANT
NO	RESOLVE	

10.11.4 How Users Can Check Their Network Connection and Domain Privileges

Users can query the ${\tt USER_HOST_ACES}$ data dictionary view to check their network and domain permissions.

The USER HOST ACES view is PUBLIC, so all users can query it.

This view hides the access control lists from the user. It evaluates the permission status for the user (GRANTED or DENIED) and filters out the NULL case because the user does not need to know when the access control lists do not apply to him or her. In other words, Oracle Database only shows the user on the network hosts that explicitly grant or deny access to him or her. Therefore, the output does not display the *.example.com and * that appear in the output from the database administrator-specific DBA HOST ACES view.

10.11.5 Example: User Checking Network Access Control Permissions

The USER_HOST_ACES data dictionary view shows network access control permissions for a host computer.

Example 10-9 shows how user preston can check her privileges to connect to www.us.example.com.

Example 10-9 User Checking Network Access Control Permissions

10.12 Configuring Network Access for Java Debug Wire Protocol Operations

Before you can debug Java PL/SQL procedures, you must be granted the jdwp ACL privilege.

If you want to debug Java PL/SQL procedures in the database through a Java Debug Wire Protocol (JDWP)-based debugger, such as SQL Developer, JDeveloper, or Oracle Developer Tools For Visual Studio (ODT), then you must be granted the <code>jdwp</code> ACL privilege to connect your database session to the debugger at a particular host.

The jdwp privilege is needed in conjunction with the DEBUG CONNECT SESSION system privilege.

If you have not been granted the jdwp ACL privilege, then when you try to debug your Java and PL/SQL stored procedures from a remote host, the following errors may appear:

```
ORA-24247: network access denied by access control list (ACL) ORA-06512: at "SYS.DBMS DEBUG JDWP", line line\ number
```

• To configure network access for JDWP operations, use the DBMS NETWORK ACL ADMIN.APPEND HOST ACE procedure.

The following example illustrates how to configure network access for JDWP operations.

In this specification:

- host can be a host name, domain name, IP address, or subnet.
- port_number enables you to specify a range of ports. If you want to use any port, then omit the lower port and upper port values.

See Also:

- Oracle Database Java Developer's Guide for more information about debugging server applications with JDWP
- Oracle SQL Developer User's Guide for information about remote debugging in SQL Developer

10.13 Data Dictionary Views for Access Control Lists Configured for User Access

Oracle Database provides data data dictionary views that you can use to find information about existing access control lists.

Table 10-1 lists these views.

Table 10-1 Data Dictionary Views That Display Information about Access Control Lists

View	Description
DBA_HOST_ACES	Shows the network privileges defined for the network hosts. The SELECT privilege on this view is granted to the SELECT_CATALOG_ROLE role only.



Table 10-1 (Cont.) Data Dictionary Views That Display Information about Access Control Lists

View	Description
DBA_WALLET_ACES	Lists the wallet path, ACE order, start and end times, grant type, privilege, and information about principals
DBA_WALLET_ACLS	Shows the access control list assignments to the wallets. The SELECT privilege on this view is granted to the SELECT_CATALOG_ROLE role only.
DBA_HOST_ACLS	Shows the access control list assignments to the network hosts. The SELECT privilege on this view is granted to the SELECT_CATALOG_ROLE role only.
USER_HOST_ACES	Shows the status of the network privileges for the current user to access network hosts. The SELECT privilege on the view is granted to PUBLIC.
USER_WALLET_ACES	Shows the status of the wallet privileges for the current user to access contents in the wallets. The <code>SELECT</code> privilege on the view is granted to <code>PUBLIC</code> .

Related Topics

Oracle Database Reference



11

Managing Security for a Multitenant Environment in Enterprise Manager

You can manage common and local users and roles for a multitenant environment by using Oracle Enterprise Manager.

This section contains the following topics:

- About Managing Security for a Multitenant Environment in Enterprise Manager
 Oracle Enterprise Manager Cloud Control supports the management of multitenant
 environment security.
- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- Managing Common and Local Users in Enterprise Manager
 In a multitenant environment, Oracle Enterprise Manager enables you to create, edit, and drop common and local users.
- Managing Common and Local Roles and Privileges in Enterprise Manager
 In a multitenant environment, you can use Oracle Enterprise Manager to create, edit, drop, and revoke common and local roles.

11.1 About Managing Security for a Multitenant Environment in Enterprise Manager

Oracle Enterprise Manager Cloud Control supports the management of multitenant environment security.

In a multitenant environment, you can use Oracle Enterprise Manager Cloud Control to create, manage, and monitor common users and roles for both the root and the associated pluggable databases (PDBs).

Enterprise Manager enables you to switch easily between the root and a designated PDB.

11.2 Logging into a Multitenant Environment in Enterprise Manager

In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

This section contains the following topics:

Logging into a CDB or a PDB
 Different variations of the Enterprise Manager Database login page appear automatically based on the feature that you requested while logging in.

Switching to a Different PDB or to the Root
 From Oracle Enterprise Manager, you can switch from one PDB to a different PDB, or to
 the root.

11.2.1 Logging into a CDB or a PDB

Different variations of the Enterprise Manager Database login page appear automatically based on the feature that you requested while logging in.

To log into a multitenant environment as a CDB administrator (an Enterprise Manager user who has the CONNECT privilege on the CDB target) to use a CDB-scoped feature:

1. Log into Oracle Enterprise Manager Cloud Control as either user SYSTEM or SYSMAN.

The URL is as follows:

https://host:port/em

- 2. Navigate to the Databases page.
- Select the database that you want to access.

The database home page appears.

 Select the menu item for the action that you want to perform, such as selecting Administration, then Security, and then Users to authenticate a user.

The Database Login page appears. The following example shows the Database Login page for the CDB (because the database name is shown as CDB\$ROOT). Because of this name, this page is colloquially referred to as the database login page for the root of the multitenant environment. The **Database** field refers to the current database; had you selected a PDB, then the name of the PDB would appear in this field.



5. Log in using the appropriate credentials.

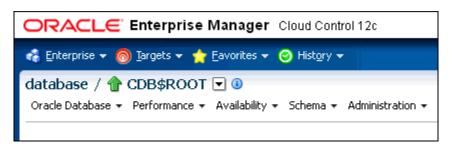
Remember that only common users can log into the root, and that the names of common users begin with C## or C##. Both common and local users can log into a PDB, depending on their privileges.

11.2.2 Switching to a Different PDB or to the Root

From Oracle Enterprise Manager, you can switch from one PDB to a different PDB, or to the root.

1. At the top left side of the page, find the **database** link.

In the **database** link, the current container name appears. The following example shows that the current database is the CDB itself (CDB\$ROOT), colloquially known as the root.



2. Select the menu icon to the right of the container, and from this menu, select the database that you want to access.

If the menu item does not appear, then navigate to a page where it does appear, such as the Database home page.

3. When you decide which activity you want to perform (such as creating users), log in with the appropriate privileges.

If you attempt to perform an activity without first having authenticated with the appropriate privileges, then you will be prompted to log in with the appropriate privilege.

11.3 Managing Common and Local Users in Enterprise Manager

In a multitenant environment, Oracle Enterprise Manager enables you to create, edit, and drop common and local users.

This section contains the following topics:

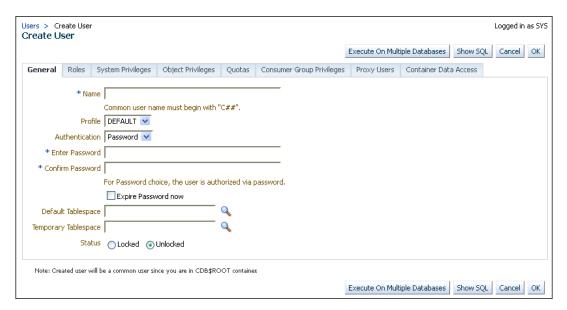
- Creating a Common User Account in Enterprise Manager
 A common user is a user that exists in the root and can access PDBs in the CDB.
- Editing a Common User Account in Enterprise Manager You can edit a common user account from the root.
- Dropping a Common User Account in Enterprise Manager You can drop a common user from the CDB root.
- Creating a Local User Account in Enterprise Manager
 A local user is a user that exists only in a specific PDB and does not have access to any other PDBs in the multitenant environment.
- Editing a Local User Account in Enterprise Manager
 You can edit a local user from the PDB in which the local user resides.
- Dropping a Local User Account in Enterprise Manager
 You can drop a local user from the PDB in which the local user resides.

11.3.1 Creating a Common User Account in Enterprise Manager

A common user is a user that exists in the root and can access PDBs in the CDB.

- 1. From the Enterprise Manager database home page, log in to the root as a common user who has the common CREATE USER and SET CONTAINER privileges.
- From the Administration menu, select Security, then Users.If prompted, enter your login information. Afterward, the Users page appears.
- Click Create.

The Create User page appears.



4. Select the options to create a common user and grant this user privileges.

Ensure that you preface the user name with C## or C##.

Click OK or Apply.

The common user is created in the root and will appear in the Users page for any associated PDBs.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.3.2 Editing a Common User Account in Enterprise Manager

You can edit a common user account from the root.

- 1. From the Enterprise Manager database home page, log in to the root as a common user who has the common CREATE USER and SET CONTAINER privileges.
 - If you are logging into the root, then ensure that you are a common user who has the common CREATE USER and SET CONTAINER privileges.

- If you are logging into a PDB, ensure that you have the CREATE USER privilege for that PDB.
- 2. From the Administration menu, select Security, then Users.

If prompted, enter your login information. Afterward, the Users page appears. In the root, only common users are listed. In the PDB, both common and local users are listed.

Select the common user to be edited and then click Edit.

The Edit User page appears. For a common user in the root, you can modify all settings for the common user. For a common user in a PDB, you cannot change the user password, default tablespace, and temporary tablespace. The settings that you make apply only to the current PDB. The following screen shows how a common user Edit User page appears in a PDB.



- 4. Modify the common user as necessary.
- Click Apply.

Related Topics

- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- Methods of Altering Common or Local User Accounts
 You can use the ALTER USER statement or the PASSWORD command to alter both common and local user accounts.

11.3.3 Dropping a Common User Account in Enterprise Manager

You can drop a common user from the CDB root.

1. From the Enterprise Manager database home page, log in to the root as a common user who has the common CREATE USER and SET CONTAINER privileges.

You cannot drop common users from PDBs.

From the Administration menu, select Security, then Users.

If prompted, enter your login information. Afterward, the Users page appears, listing only common users.

- 3. Select the common user that you want to drop and then click **Delete**.
- 4. Confirm that you want to delete the common user.



Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.3.4 Creating a Local User Account in Enterprise Manager

A local user is a user that exists only in a specific PDB and does not have access to any other PDBs in the multitenant environment.

- 1. From the Enterprise Manager database home page, log in to the root as a local or common user who has the local CREATE USER privilege.
- 2. From the **Administration** menu, select **Security**, then **Users**.

If prompted, enter your login information. Afterward, the Users page appears, showing only local users for the current PDB.

3. Click Create.

The Create User page appears.

4. Select the options that create a local user and grant this user privileges.

Ensure that you do not preface the user name with C## or C##.

5. Click OK.

The local user is created in the current PDB.

Related Topics

- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- About Creating Local User Accounts
 Be aware of local user account restrictions such as where they can be created, naming conventions, and objects stored in their schemas.

11.3.5 Editing a Local User Account in Enterprise Manager

You can edit a local user from the PDB in which the local user resides.

- 1. From the Enterprise Manager database home page, log in to the PDB as a local or common user who has the local CREATE USER privilege.
- 2. From the **Administration** menu, select **Security**, then **Users**.

If prompted, enter your login information. Afterward, the Users page appears, showing only local users for the current PDB and common users.

3. Select the local user to be edited and then click **Edit**.

The Edit User page appears.

- 4. Modify the local user as necessary.
- Click Apply.



Related Topics

- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- Methods of Altering Common or Local User Accounts
 You can use the ALTER USER statement or the PASSWORD command to alter both common and local user accounts.

11.3.6 Dropping a Local User Account in Enterprise Manager

You can drop a local user from the PDB in which the local user resides.

- 1. From the Enterprise Manager database home page, log in to the PDB as a local or common user who has the local CREATE USER privilege.
- 2. From the **Administration** menu, select **Security**, then **Users**.

If prompted, enter your login information. Afterward, the Users page appears, showing only local users for the current PDB and common users. (You cannot drop common users from a PDB.)

- 3. Select the local user you want to drop and then click **Delete**.
 - Enterprise Manager prompts you to confirm deletion of the user.
- 4. Confirm that you want to delete the local user.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.4 Managing Common and Local Roles and Privileges in Enterprise Manager

In a multitenant environment, you can use Oracle Enterprise Manager to create, edit, drop, and revoke common and local roles.

This section contains the following topics:

- Creating a Common Role in Enterprise Manager
 Common roles can be used to assign common privileges to common users.
- Editing a Common Role in Enterprise Manager You can edit a common role from the root.
- Dropping a Common Role in Enterprise Manager You can drop a common role from the root.
- Revoking Common Privilege Grants in Enterprise Manager You can revoke common privilege grants from the root.
- Creating a Local Role in Enterprise Manager
 A common role can be used to assign a local set of privileges to local users later.
- Editing a Local Role in Enterprise Manager
 You can edit a local role in the PDB in which the local role resides.



- Dropping a Local Role in Enterprise Manager
 You can drop local role from the PDB in which the local role resides.
- Revoking Local Privilege Grants in Enterprise Manager
 You can revoke local privileges in the PDB in which the privileges are used.

11.4.1 Creating a Common Role in Enterprise Manager

Common roles can be used to assign common privileges to common users.

These roles are valid across all containers of the multitenant environment.

- 1. From the Enterprise Manager database home page, log in to the root as a common user who has the common CREATE ROLE and SET CONTAINER privileges.
- From the Administration menu, select Security, then Roles.
 If prompted, enter your login information. Afterward, the Create Role page appears.
- 3. Click Create.

The Create Role page appears.



Select the options that create a common role and grant this role privileges.

Ensure that you preface the role name with C## or C##.

5. Click OK.

The common role is created in the root.

Related Topics

- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- Rules for Creating Common Roles
 When you create a common role, you must follow special rules.
- Granting or Revoking Privileges to Access a PDB
 You can grant and revoke privileges for PDB access in a multitenant environment.

11.4.2 Editing a Common Role in Enterprise Manager

You can edit a common role from the root.

From the Enterprise Manager database home page, log in to the root or the PDB. If you
are logging into the root, then ensure that you are a common user who has the common

CREATE ROLE and SET CONTAINER privileges. If you are logging into a PDB, ensure that you have the CREATE ROLE privilege for that PDB.

2. From the Administration menu, select Security, then Roles.

If prompted, enter your login information. Afterward, the Roles page appears. In the root, only common roles are shown. In the PDB, both common and local roles are shown.

Select the common role to be edited and then click Edit.

The Edit Role page appears. For a common user in the root, you can modify all settings for the common user.

For a common role in a PDB, you can only change the role's authentication and grant this user different roles, system privileges, object privileges, and consumer group privileges. These settings apply only to the current PDB.

- 4. Modify the common user as necessary.
- Click Apply.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.4.3 Dropping a Common Role in Enterprise Manager

You can drop a common role from the root.

 From the Enterprise Manager database home page, log in to the root as a common user who has the common CREATE ROLE and SET CONTAINER privileges.

You cannot drop common roles from PDBs.

2. From the Administration menu, select Security, then Roles.

If prompted, enter your login information. Afterward, the Roles page appears, showing only common roles.

- 3. Select the common role that you want to drop and then click **Delete**.
- 4. Confirm that you want to delete the common role.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.4.4 Revoking Common Privilege Grants in Enterprise Manager

You can revoke common privilege grants from the root.

- 1. From the Enterprise Manager database home page, log in to the root as a common user who has the common CREATE USER, CREATE ROLE, and SET CONTAINER privileges.
- 2. From the Administration menu, select Security, then Users.

The Users page lists the common users.

3. Select the user whose privileges you want to revoke and then click **Edit**.

The Edit User page appears.



Select Roles or the appropriate Privileges tab.

Enterprise Manager displays a list of roles and privileges assigned to this user.

- Select Edit List and then remove the roles or privileges as necessary.
- 6. Click the **OK** button.

Related Topics

- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- Granting or Revoking Privileges to Access a PDB
 You can grant and revoke privileges for PDB access in a multitenant environment.

11.4.5 Creating a Local Role in Enterprise Manager

A common role can be used to assign a local set of privileges to local users later.

These roles will be valid across PDB containers for whom they are defined.

- 1. From the Enterprise Manager database home page, log in to the PDB as a user who has the local CREATE ROLE privilege.
- 2. From the Administration menu, select Security, then Roles.

The Roles page appears.

3. Click Create.

If prompted, enter your login information. Afterward, the Create Role page appears.

4. Select the options that create a local role and grant this role privileges.

Ensure that you do not preface the role name with C## or C##.

5. Click OK.

The local role is created in the current PDB.

Related Topics

- Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.
- Granting or Revoking Privileges to Access a PDB
 You can grant and revoke privileges for PDB access in a multitenant environment.

11.4.6 Editing a Local Role in Enterprise Manager

You can edit a local role in the PDB in which the local role resides.

- 1. From the Enterprise Manager database home page, log in to the PDB as a user who has the local CREATE ROLE privilege.
- 2. From the **Administration** menu, select **Security**, then **Roles**.

If prompted, enter your login information. Afterward, the Roles page appears, showing only local roles for the current PDB and common roles.

3. Select the local role to be edited and then click Edit.

The Edit User page appears.



- 4. Modify the local user as necessary.
- Click Apply.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.4.7 Dropping a Local Role in Enterprise Manager

You can drop local role from the PDB in which the local role resides.

- 1. From the Enterprise Manager database home page, log in to the PDB as a user who has the local CREATE ROLE privilege.
- 2. From the Administration menu, select Security, then Role.

If prompted, enter your login information. Afterward, the Roles page appears, showing only local roles for the current PDB and common roles. (You cannot drop common roles from a PDB.)

- 3. Select the local role you want to drop and then click **Delete**.
 - Enterprise Manager prompts you to confirm deletion of the role.
- Confirm that you want to delete the local role.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.

11.4.8 Revoking Local Privilege Grants in Enterprise Manager

You can revoke local privileges in the PDB in which the privileges are used.

- From the Enterprise Manager database home page, log in to the PDB as a common or local user who has the CREATE USER and CREATE ROLE privileges.
- 2. From the **Administration** menu, select **Security**, then **Users**.

If prompted, enter your login information. Afterward, the Users page appears. In a PDB, both common and local users are listed.

3. Select the user whose privileges you want to revoke and then click **Edit**.

The Edit User page appears.

4. Select **Roles** or the appropriate **Privileges** tab.

Enterprise Manager displays a list of roles and privileges assigned to this user.

- 5. Select **Edit List** and then remove the privileges as necessary.
- 6. Click the **OK** button.

Related Topics

Logging into a Multitenant Environment in Enterprise Manager
 In a multitenant environment, you can log in to a CDB or a PDB, and switch from a PDB to a different PDB or to the root.



Granting or Revoking Privileges to Access a PDB
 You can grant and revoke privileges for PDB access in a multitenant environment.



Part II

Application Development Security

Part II describes how to manage application development security.

Managing Security for Application Developers
 A security policy for application developers should encompass areas such as password management and securing external procedures and application privileges.



Managing Security for Application Developers

A security policy for application developers should encompass areas such as password management and securing external procedures and application privileges.

About Application Security Policies

An application security policy is a list of application security requirements and rules that regulate user access to database objects.

Considerations for Using Application-Based Security

An application security implementation should consider both application and database users and whether to enforce security in the application or in the database.

Securing Passwords in Application Design

Oracle provides strategies for securely invoking password services, such as from scripts, and for applying these strategies to other sensitive data.

Securing External Procedures

An external procedure is stored in a .dll or an .so file, separately from the database, and can be through a credential authentication.

Securing LOBs with LOB Locator Signatures

You can secure large objects (LOB) by regenerating their LOB locator signatures.

Managing Application Privileges

Most database applications involve different privileges on different schema objects.

Advantages of Using Roles to Manage Application Privileges

Grouping application privileges in a role aids privilege management.

Creating Secure Application Roles to Control Access to Applications A secure application role is only applied through its associated RL/S

A secure application role is only enabled through its associated PL/SQL package or procedure.

Association of Privileges with User Database Roles

Ensure that users have only the privileges associated with the current database role.

Protecting Database Objects by Using Schemas

A schema is a security domain that can contain database objects. Privileges granted to users and roles control access to these database objects.

Object Privileges in an Application

When you design an application, consider the types of users and the level access they need.

Parameters for Enhanced Security of Database Communication

Parameters can be used to manage security, such as handling bad packets from protocol errors or configuring the maximum number of authentication errors.

12.1 About Application Security Policies

An application security policy is a list of application security requirements and rules that regulate user access to database objects.

Creating an application security policy is the first step to create a secure database application. You should draft security policies for each database application. For example, each database

application should have one or more database roles that provide different levels of security when executing the application. You then can grant the database roles to other roles or directly to specific users.

Applications that can potentially allow unrestricted SQL statement processing (through tools such as SQL*Plus or SQL Developer) also need security policies that prevent malicious access to confidential or important schema objects. In particular, you must ensure that your applications handle passwords in a secure manner.

12.2 Considerations for Using Application-Based Security

An application security implementation should consider both application and database users and whether to enforce security in the application or in the database.

- Are Application Users Also Database Users?
 Where possible, build applications in which application users are database users, so that you can use the intrinsic security mechanisms of the database.
- Is Security Better Enforced in the Application or in the Database?
 Oracle recommends that applications use the security enforcement mechanisms of the database as much as possible.

12.2.1 Are Application Users Also Database Users?

Where possible, build applications in which application users are database users, so that you can use the intrinsic security mechanisms of the database.

For many commercial packaged applications, application users are not database users. For these applications, multiple users authenticate themselves to the application, and the application then connects to the database as a single, highly-privileged user. This is called the *One Big Application User* model.

Applications built in this way generally cannot use many of the intrinsic security features of the database, because the identity of the user is not known to the database. However, you can use client identifiers to perform some types of tracking. For example, the OCI_ATTR_CLIENT_IDENTIFIER attribute of the Oracle Call Interface method OCIAttrSet can be used to enable auditing and monitoring of client connections. Client identifiers can be used to gather audit trail data on individual Web users, apply rules that restrict data access by Web users, and monitor and trace applications that each Web user users.

Table 12-1 describes how the One Big Application User model affects various Oracle Database security features:

Table 12-1 Features Affected by the One Big Application User Model

Oracle Database Feature	Limitations of One Big Application User Model
Auditing	A basic principle of security is accountability through auditing. If One Big Application User performs all actions in the database, then database auditing cannot hold individual users accountable for their actions. The application must implement its own auditing mechanisms to capture individual user actions.
Oracle strong authentication	Strong forms of authentication (such as client authentication over SSL, tokens, and so on) cannot be used if the client authenticating to the database is the application, rather than an individual user.



1able 12-1	(Cont.) Features Affected by the One Big Application User Model

Oracle Database Feature	Limitations of One Big Application User Model
Roles	Roles are assigned to database users. Enterprise roles are assigned to enterprise users who, though not created in the database, are known to the database. If application users are not database users, then the usefulness of roles is diminished. Applications must then craft their own mechanisms to distinguish between the privileges which various application users need to access data within the application.
Enterprise user management	The Enterprise user management feature enables an Oracle database to use the Oracle Identity Management Infrastructure by securely storing and managing user information and authorizations in an LDAP-based directory such as Oracle Internet Directory. While enterprise users do not need to be created in the database, they do need to be known to the database. The One Big Application User model cannot take advantage of Oracle Identity Management.

12.2.2 Is Security Better Enforced in the Application or in the Database?

Oracle recommends that applications use the security enforcement mechanisms of the database as much as possible.

Applications, whose users are also database users, can either build security into the application, or rely on intrinsic database security mechanisms such as granular privileges, virtual private databases (fine-grained access control with application context), roles, stored procedures, and auditing (including fine-grained auditing).

When security is enforced in the database itself, rather than in the application, it cannot be bypassed. The main shortcoming of application-based security is that security is bypassed if the user bypasses the application to access data. For example, a user who has SQL*Plus access to the database can execute queries without going through the Human Resources application. The user, therefore, bypasses all of the security measures in the application.

Applications that use the One Big Application User model must build security enforcement into the application rather than use database security mechanisms. Because it is the application, and not the database, that recognizes users; the application itself must enforce security measures for each user.

This approach means that each application that accesses data must re-implement security. Security becomes expensive, because organizations must implement the same security policies in multiple applications, and each new application requires an expensive reimplementation.

Related Topics

Potential Security Problems of Using Ad Hoc Tools
 Ad hoc tools can pose problems if malicious users have access to such tools.

12.3 Securing Passwords in Application Design

Oracle provides strategies for securely invoking password services, such as from scripts, and for applying these strategies to other sensitive data.

- General Guidelines for Securing Passwords in Applications
 Guidelines for securing passwords in applications cover areas such as platform-specific security threats.
- Use of an External Password Store to Secure Passwords
 You can store password credentials for connecting to a database by using a client-side Oracle wallet.
- Securing Passwords Using the ORAPWD Utility
 SYSDBA or SYSOPER users can use password files to connect to an application over a
 network.
- Example: Java Code for Reading Passwords
 You can create Java packages that can be used to read passwords.

12.3.1 General Guidelines for Securing Passwords in Applications

Guidelines for securing passwords in applications cover areas such as platform-specific security threats.

- Platform-Specific Security Threats
 You should be aware of potential security threats, which may not be obvious.
- Guidelines for Designing Applications to Handle Password Input
 Oracle provides guidelines for designing applications to handle password input.
- Guidelines for Configuring Password Formats and Behavior
 Oracle Database provides guidelines for configuring password formats and behavior.
- Guidelines for Handling Passwords in SQL Scripts
 Oracle provides guidelines for handling passwords in SQL scripts.

12.3.1.1 Platform-Specific Security Threats

You should be aware of potential security threats, which may not be obvious.

These security threats are as follows:

- On UNIX and Linux platforms, command parameters are available for viewing by all
 operating system users on the same host computer. As a result, passwords entered on
 the command line could be exposed to other users. However, do not assume that nonUNIX and Linux platforms are safe from this threat.
- On some UNIX platforms, such as HP Tru64 and IBM AIX, environment variables for all processes are available for viewing by all operating system users. However, do not assume that non-UNIX and Linux platforms are safe from this threat.
- On Microsoft Windows, the command recall feature (the Up arrow) remembers user input across command invocations. For example, if you use the CONNECT SYSTEM/ password notation in SQL*Plus, exit, and then press the Up arrow to repeat the CONNECT command, the command recall feature reveals the connect string and displays the password. In addition, do not assume that non-Microsoft Windows platforms are safe from this threat.

12.3.1.2 Guidelines for Designing Applications to Handle Password Input

Oracle provides guidelines for designing applications to handle password input.

 Design applications to interactively prompt for passwords. For command-line utilities, do not force users to expose passwords at a command prompt. Check the APIs for the programming language you use to design applications for the best way to handle passwords from users. For an example of Java code that handles this functionality, see Example: Java Code for Reading Passwords.

• Protect your database against code injection attacks. Code injection attacks most commonly occur in the client application tool that sends SQL to the database (for example, SQL*Plus, or any Oracle Call Interface (OCI) or JDBC application. This includes database drivers that are built using these tools. A SQL injection attack causes SQL statements to behave in a manner that is not intended by the PL/SQL application. The injection attack takes place before the statement is sent to the database. For example, an intruder can bypass password authentication by setting a WHERE clause to TRUE.

To address the problem of SQL injection attacks, use bind variable arguments or create validation checks. If you cannot use bind variables, then consider using the <code>DBMS_ASSERT</code> PL/SQL package to validate the properties of input values. *Oracle Database PL/SQL Packages and Types Reference* describes the <code>DBMS_ASSERT</code> package in detail. You also should review any grants to roles such as <code>PUBLIC</code>.

Note that client applications users may not always associate SQL injection with PL/SQL, because the injection could occur before the statement is sent to the database.

See *Oracle Database PL/SQL Language Reference* for more information about preventing SQL injection.

- If possible, design your applications to defer authentication. For example:
 - Use certificates for logins.
 - Authenticate users by using facilities provided by the operating system. For example, applications on Microsoft Windows can use domain authentication.
- Mask or encrypt passwords. If you must store passwords, then mask or encrypt them.
 For example, you can mask passwords in log files and encrypt passwords in recovery files.
- Authenticate each connection. For example, if schema A exists in database 1, then do
 not assume that schema A in database 2 is the same user. Similarly, the local operating
 system user psmith is not necessarily the same person as remote user psmith.
- Do not store clear text passwords in files or repositories. Storing passwords in files increases the risk of an intruder accessing them.
- Use a single master password. For example:
 - You can grant a single database user proxy authentication to act as other database users. In this case, only a single database password is needed. See Proxy User Accounts and the Authorization of Users to Connect Through Them for more information.
 - You can create a password wallet, which can be opened by the master password. The
 wallet then contains the other passwords. See *Oracle Database Enterprise User*Security Administrator's Guide for more information about Wallet Manager.

12.3.1.3 Guidelines for Configuring Password Formats and Behavior

Oracle Database provides guidelines for configuring password formats and behavior.

- Limit the lifetime for passwords. You can set a password lifetime, after which the
 password expires and must be changed before the user can log in to the account. See
 About Controlling Password Aging and Expiration for parameters you can use to control
 the lifetime of a password.
- Limit the ability of users to reuse old passwords. See Controlling the User Ability to Reuse Previous Passwords for more information.

- Force users to create strong, secure passwords. See Guidelines for Securing Passwords for advice on creating strong passwords. About Password Complexity Verification explains how you can customize password requirements.
- Enable case sensitivity in passwords. See Managing Password Case Sensitivity for more information.

Related Topics

- Minimum Requirements for Passwords
 Oracle provides a set of minimum requirements for passwords.
- About Password Complexity Verification
 Complexity verification checks that each password is complex enough to protect against intruders who try to guess user passwords.

12.3.1.4 Guidelines for Handling Passwords in SQL Scripts

Oracle provides guidelines for handling passwords in SQL scripts.

• Do not invoke SQL*Plus with a password on the command line, either in programs or scripts. If a password is required but omitted, SQL*Plus prompts the user for it and then automatically disables the echo feature so that the password is not displayed.

The following examples are secure because passwords are not exposed on the command line. Oracle Database also automatically encrypts these passwords over the network.

```
$ sqlplus system
Enter password: password

SQL> CONNECT SYSTEM
Enter password: password
```

The following example exposes the password to other operating system users:

```
sqlplus system/password
```

The next example poses two security risks. First, it exposes the password to other users who may be watching over your shoulder. Second, on some platforms, such as Microsoft Windows, it makes the password vulnerable to a command line recall attack.

```
$ sqlplus /nolog
SQL> CONNECT SYSTEM/password
```

• For SQL scripts that require passwords or secret keys, for example, to create an account or to log in as an account, do not use positional parameters, such as substitution variables &1, &2, and so on. Instead, design the script to prompt the user for the value. You should also disable the echo feature, which displays output from a script or if you are using spool mode. To disable the echo feature, use the following setting:

```
SET ECHO OFF
```

A good practice is to ensure that the script makes the purpose of the value clear. For example, it should be clear whether or not the value will establish a new value, such as an account or a certificate, or if the value will authenticate, such as logging in to an existing account.

The following example is secure because it prevents users from invoking the script in a manner that poses security risks: It does not echo the password; it does not record the password in a spool file.

```
SET VERIFY OFF
ACCEPT user CHAR PROMPT 'Enter user to connect to: '
```



ACCEPT password CHAR PROMPT 'Enter the password for that user: ' HIDE CONNECT &user/&password

In this example:

- SET VERIFY OFF prevents the password from being displayed. (SET VERIFY lists each line of the script before and after substitution.) Combining the SET VERIFY OFF command with the HIDE command is a useful technique for hiding passwords and other sensitive input data.
- ACCEPT password CHAR PROMPT includes the HIDE option for the ACCEPT password prompt, which prevents the input password from being echoed.

The next example, which uses positional parameters, poses security risks because a user may invoke the script by passing the password on the command line. If the user does not enter a password and instead is prompted, the danger lies in that whatever the user types is echoed to the screen and to a spool file if spooling is enabled.

CONNECT &1/&2

- Control the log in times for batch scripts. For batch scripts that require passwords, configure the account so that the script can only log in during the time in which it is supposed to run. For example, suppose you have a batch script that runs for an hour each evening starting at 8 p.m. Set the account so that the script can only log in during this time. If an intruder manages to gain access, then he or she has less of a chance of exploiting any compromised accounts.
- Be careful when using DML or DDL SQL statements that prompt for passwords. In this case, sensitive information is passed in clear text over the network. You can remedy this problem by using Oracle strong authentication.

The following example of altering a password is secure because the password is not exposed:

password psmith
Changing password for psmith
New password: password
Retype new password: password

This example poses a security risk because the password is exposed both at the command line and on the network:

ALTER USER psmith IDENTIFIED BY password

12.3.2 Use of an External Password Store to Secure Passwords

You can store password credentials for connecting to a database by using a client-side Oracle wallet.

An Oracle wallet is a secure software container that stores the authentication and signing credentials needed for a user to log in.

See Also:

- Managing the Secure External Password Store for Password Credentials for more information about the secure external password store
- Oracle Database Enterprise User Security Administrator's Guide for information about using Oracle Wallet Manager to configure Oracle wallets

12.3.3 Securing Passwords Using the ORAPWD Utility

SYSDBA or SYSOPER users can use password files to connect to an application over a network.

To create the password file, use the ORAPWD utility.



Oracle Database Administrator's Guide for more information about creating and maintaining a password file

12.3.4 Example: Java Code for Reading Passwords

You can create Java packages that can be used to read passwords.

Example 12-1 demonstrates how to create a Java package that can be used to read passwords.

Example 12-1 Java Code for Reading Passwords

```
// Change the following line to a name for your version of this package
package passwords.sysman.emSDK.util.signing;
import java.io.IOException;
import java.io.PrintStream;
import java.io.PushbackInputStream;
import java.util.Arrays;
^{\star} The static readPassword method in this class issues a password prompt
* on the console output and returns the char array password
* entered by the user on the console input.
public final class ReadPassword {
 //----
  ^{\star} Test driver for readPassword method.
  ^{\star} @param args the command line args
 public static void main(String[] args) {
   char[] pass = ReadPassword.readPassword("Enter password: ");
   System.out.println("The password just entered is \""
     + new String(pass) + "\"");
   System.out.println("The password length is " + pass.length);
```

```
* Issues a password prompt on the console output and returns
 * the char array password entered by the user on the console input.
 ^{\star} The password is not displayed on the console (chars are not echoed).
 * As soon as the returned char array is not needed,
 * it should be erased for security reasons (Arrays.fill(charArr, ' '));
 * A password should never be stored as a java String.
 * Note that Java 6 has a Console class with a readPassword method,
 * but there is no equivalent in Java 5 or Java 1.4.
 * The readPassword method here is based on Sun's suggestions at
 * http://java.sun.com/developer/technicalArticles/Security/pwordmask.
 ^{\star} @param prompt the password prompt to issue
 * @return new char array containing the password
* @throws RuntimeException if some error occurs
* /
public static final char[] readPassword(String prompt)
throws RuntimeException {
 try {
    StreamMasker masker = new StreamMasker(System.out, prompt);
   Thread threadMasking = new Thread(masker);
   int firstBvte = -1;
   PushbackInputStream inStream = null;
    try {
     threadMasking.start();
     inStream = new PushbackInputStream(System.in);
     firstByte = inStream.read();
    } finally {
     masker.stopMasking();
    try {
     threadMasking.join();
    } catch (InterruptedException e) {
     throw new RuntimeException("Interrupt occurred when reading password");
    if (firstByte == -1) {
    throw new RuntimeException("Console input ended unexpectedly");
    if (System.out.checkError()) {
     throw new RuntimeException("Console password prompt output error");
   inStream.unread(firstByte);
   return readLineSecure(inStream);
 catch (IOException e) {
    throw new RuntimeException ("I/O error occurred when reading password");
          -----
/**
* Reads one line from an input stream into a char array in a secure way
 * suitable for reading a password.
 * The char array will never contain a '\n' or '\r'.
 * @param inStream the pushback input stream
 * @return line as a char array, not including end-of-line-chars;
 * never null, but may be zero length array
 * @throws RuntimeException if some error occurs
private static final char[] readLineSecure(PushbackInputStream inStream)
throws RuntimeException {
 if (inStream == null) {
```

```
throw new RuntimeException("readLineSecure inStream is null");
  }
  try {
    char[] buffer = null;
    try {
      buffer = new char[128];
      int offset = 0;
      // EOL is '\n' (unix), '\r\n' (windows), '\r' (mac)
      loop:
      while (true) {
        int c = inStream.read();
        switch (c) {
        case -1:
        case '\n':
         break loop;
        case '\r':
         int c2 = inStream.read();
         if ((c2 != '\n') && (c2 != -1))
           inStream.unread(c2);
         break loop;
        default:
          buffer = checkBuffer(buffer, offset);
          buffer[offset++] = (char) c;
          break;
        }
      char[] result = new char[offset];
      System.arraycopy(buffer, 0, result, 0, offset);
      return result;
    finally {
      if (buffer != null)
        Arrays.fill(buffer, ' ');
  }
  catch (IOException e) {
    throw new RuntimeException("I/O error occurred when reading password");
//-----
/**
 * This is a helper method for readLineSecure.
 ^{\star} @param buffer the current char buffer
 * @param offset the current position in the buffer
 * @return the current buffer if it is not yet full;
 * otherwise return a larger buffer initialized with a copy
 ^{\star}~ of the current buffer and then erase the current buffer
 ^{\star} @throws RuntimeException if some error occurs
private static final char[] checkBuffer(char[] buffer, int offset)
throws RuntimeException
  if (buffer == null)
    throw new RuntimeException("checkBuffer buffer is null");
  if (offset < 0)
    throw new RuntimeException("checkBuffer offset is negative");
  if (offset < buffer.length)</pre>
    return buffer;
  else {
      char[] bufferNew = new char[offset + 128];
```

```
System.arraycopy(buffer, 0, bufferNew, 0, buffer.length);
     return bufferNew;
    } finally {
     Arrays.fill(buffer, ' ');
  }
* This private class prints a one line prompt
 * and erases reply chars echoed to the console.
private static final class StreamMasker
extends Thread {
 private static final String BLANKS = StreamMasker.repeatChars(' ', 10);
 private String m promptOverwrite;
 private String m setCursorToStart;
 private PrintStream m out;
  private volatile boolean m doMasking;
  //-----
  /**
  * Constructor.
   * @throws RuntimeException if some error occurs
  public StreamMasker(PrintStream outPrint, String prompt)
  throws RuntimeException {
   if (outPrint == null)
      throw new RuntimeException("StreamMasker outPrint is null");
    if (prompt == null)
     throw new RuntimeException("StreamMasker prompt is null");
    if (prompt.indexOf('\r') != -1)
     throw new RuntimeException ("StreamMasker prompt contains a CR");
    if (prompt.indexOf('\n') != -1)
     throw new RuntimeException("StreamMasker prompt contains a NL");
   m out = outPrint;
   m setCursorToStart = StreamMasker.repeatChars('\010',
     prompt.length() + BLANKS.length());
   m promptOverwrite = m setCursorToStart + prompt + BLANKS
     + m setCursorToStart + prompt;
  //----
  /**
   * Begin masking until asked to stop.
   * @throws RuntimeException if some error occurs
  public void run()
  throws RuntimeException {
    int priorityOriginal = Thread.currentThread().getPriority();
    Thread.currentThread().setPriority(Thread.MAX_PRIORITY);
    try {
     m doMasking = true;
     while (m doMasking) {
       m out.print(m promptOverwrite);
       if (m out.checkError())
         throw new RuntimeException("Console output error writing prompt");
       try {
         Thread.currentThread().sleep(1);
        } catch (InterruptedException ie) {
         Thread.currentThread().interrupt();
         return;
      }
```

```
m out.print(m setCursorToStart);
  } finally {
    Thread.currentThread().setPriority(priorityOriginal);
* Instructs the thread to stop masking.
public void stopMasking() {
 m doMasking = false;
^{\star} Returns a repeated char string.
 ^{\star} @param c the char to repeat
 ^{\star} @param length the number of times to repeat the char
 * @throws RuntimeException if some error occurs
private static String repeatChars(char c, int length)
throws RuntimeException {
  if (length < 0)
    throw new RuntimeException("repeatChars length is negative");
 StringBuffer sb = new StringBuffer(length);
  for (int i = 0; i < length; i++)
    sb.append(c);
  return sb.toString();
```

12.4 Securing External Procedures

An external procedure is stored in a .dll or an .so file, separately from the database, and can be through a credential authentication.

- About Securing External Procedures
 For safety reasons, Oracle external procedures run in a process that is physically separate
 from the database
- General Process for Configuring extproc for a Credential Authentication
 For better security, you can configure the extproc process to be authenticated through a credential.
- extproc Process Authentication and Impersonation Expected Behaviors
 The extproc process has a set of behaviors for authentication and impersonation.
- Configuring Authentication for External Procedures
 To configure a credential for extproc processes, you can use the DBMS_CREDENTIAL
 PL/SQL package.
- External Procedures for Legacy Applications
 For maximum security, set the ENFORCE_CREDENTIAL environment variable to TRUE.

12.4.1 About Securing External Procedures

For safety reasons, Oracle external procedures run in a process that is physically separate from the database.

In most cases, you configure this process to execute as a user other than the Oracle software account. When your application invokes this external procedure—such as when a library of .dll or .so files must be accessed—then Oracle Database creates an operating system process called <code>extproc</code>. By default, the <code>extproc</code> process communicates directly through your server process. In other words, if you do not use a credential, then Oracle Database creates an <code>extproc</code> process for you in the default Oracle Database server configuration, and runs <code>extproc</code> as the <code>oracle</code> software account. Alternatively, it can communicate through the Oracle Database listener.

See Also:

Guideline for Securing External Procedures

12.4.2 General Process for Configuring extproc for a Credential Authentication

For better security, you can configure the extproc process to be authenticated through a credential.

The general process is as follows:

- You create a credential.
 - The credential is in an encrypted container. Both public and private synonyms can refer to this credential. Configuring Authentication for External Procedures describes how to create this credential and configure your database to use it.
- 2. You make your initial connection to the database, which you are running in either a dedicated server or a shared server process.
- 3. Your application makes a call to an external procedure.
 - If this is the first call, then Oracle Database creates an extproc process. Note that if you want to use a credential for extproc, then you cannot use the Oracle listener to spawn the extproc process.
- 4. The extproc process impersonates (that is, it runs on behalf of your supplied credential), loads the requisite .dll, .so, .sl, or .a file, and then sends your data between SQL and C.

12.4.3 extproc Process Authentication and Impersonation Expected Behaviors

The extproc process has a set of behaviors for authentication and impersonation.

Table 12-2 describes the expected behaviors of an extproc process based on possible authentication and impersonation scenarios.

In this table, <code>GLOBAL_EXTPROC_CREDENTIAL</code> is a reserved credential name for the default credential if the credential is not explicitly specified and if the <code>ENFORCE_CREDENTIAL</code> environment variable is set to <code>TRUE</code>. Therefore, Oracle strongly recommends that you create a credential by the that name if <code>ENFORCE_CREDENTIAL</code> is set to <code>TRUE</code>.



Table 12-2 Expected Behaviors for extproc Process Authentication and Impersonation Settings

ENFORCE_CREDENTIA L Environment Variable Setting	PL/SQL Library with Credential?	GLOBAL_EXTPROC_CREDENTIAL Credential Existence	Expected Behavior
FALSE	No	No	Uses the pre-release 12c authentication, which authenticates by operating system privilege of the owners of the Oracle listener or Oracle server process.
FALSE	No	Yes	Authenticates and impersonates with the Oracle Database instance-wide supplied GLOBAL_EXTPROC_CREDENTIAL.
			If only the GLOBAL_EXTPROC_CREDENTIAL credential is in use, then the EXECUTE privilege on this global credential is automatically granted to all users implicitly.
FALSE	Yes	No	Authenticates and impersonates with the credential defined in the PL/SQL library
FALSE	Yes	Yes	Authenticates and impersonates. If both the PL/SQL library and the GLOBAL_EXTPROC_CREDENTIAL settings have credentials defined, then the credential of the PL/SQL library takes precedence.
TRUE	No	No	Returns error ORA-28575: unable to open RPC connection to external procedure agent
TRUE	No	Yes	Authenticates and impersonates with the Oracle system-wide supplied GLOBAL_EXTPROC_CREDENTIAL (footnote 1)
TRUE	Yes	No	Authenticates and impersonates with the credential defined in the PL/SQL library
TRUE	Yes	Yes	Authenticates and impersonates (footnote 2)

12.4.4 Configuring Authentication for External Procedures

To configure a credential for <code>extproc</code> processes, you can use the <code>DBMS_CREDENTIAL PL/SQL</code> package.

1. Log in to SQL*Plus as a user who has been granted the CREATE CREDENTIAL or CREATE ANY CREDENTIAL privilege.

sqlplus psmith
Enter password: password
Connected.

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

```
sqlplus psmith@hpdb
Enter password: password
Connected.
```

In addition, ensure that you also have the CREATE LIBRARY OF CREATE ANY LIBRARY privilege, and the EXECUTE object privilege on the library that contains the external calls.

2. Using the DBMS CREDENTIAL PL/SQL package, create a new credential.

For example:

```
BEGIN

DBMS_CREDENTIAL.CREATE_CREDENTIAL (
    credential_name => 'smith_credential',
    user_name => 'tjones',
    password => 'password')
END;
//
```

In this example:

- credential_name: Enter the name of the credential. Optionally, prefix it with the name
 of a schema (for example, psmith.smith_credential). If the ENFORCE_CREDENTIAL
 environment variable is set to TRUE, then you should create a credential with
 credential_name GLOBAL EXTPROC CREDENTIAL.
- user name: Enter a valid operating system user name to be to used to run as the user.
- password: Enter the password for the user name user.
- Associate the credential with a PL/SQL library.

For example:

```
CREATE OR REPLACE LIBRARY ps_lib
AS 'smith_lib.so' IN DLL_LOC
CREDENTIAL smith credential;
```

In this example, DLL_LOC is a directory object that points to the <code>\$ORACLE_HOME/bin</code> directory. Oracle does not recommend using absolute paths to the DLL.

When the PL/SQL library is loaded by an external procedure call through the extproc process, extproc now can authenticate and impersonate on behalf of the defined smith credential credential.

4. Register the external procedure by creating a PL/SQL procedure or function that tells PL/SQL how to call the external procedure and what arguments to pass to it.

For example, to create a function that registers an external procedure that was written in C, only use the AS LANGUAGE C, LIBRARY, and NAME clauses of the CREATE FUNCTION statement, as follows:

```
CREATE OR REPLACE FUNCTION getInt (x VARCHAR2, y BINARY_INTEGER)
RETURN BINARY_INTEGER
AS LANGUAGE C
LIBRARY ps_lib
NAME "get_int_vals"
PARAMETERS (x STRING, y int);
```



See Also:

- Guideline for Securing External Procedures
- Oracle Database PL/SQL Packages and Types Reference, for information about the DBMS CREDENTIAL package
- Oracle Call Interface Programmer's Guide for information about the extproc agent
- Oracle Database Net Services Administrator's Guide for detailed information about the extproc.ora file

12.4.5 External Procedures for Legacy Applications

For maximum security, set the ENFORCE CREDENTIAL environment variable to TRUE.

However, if you must accommodate backward compatibility, then set <code>ENFORCE_CREDENTIAL</code> to <code>FALSE</code>. <code>FALSE</code> enables the <code>extproc</code> process to authenticate, impersonate, and perform user-defined callout functions on behalf of the supplied credential when either of the following occurs:

- · The credential is defined with a PL/SQL library.
- The credential is not defined but the GLOBAL EXTPROC CREDENTIAL credential exists.

If neither of these credential definitions is in place, then setting the ENFORCE_CREDENTIAL parameter to FALSE sets the extproc process to be authenticated by the operating system privilege of the owners of the Oracle listener or Oracle server process.

For legacy applications that run on top of extproc processes, ideally you should change the legacy application code to associate all alias libraries with credentials. If you cannot do this, then Oracle Database uses the <code>GLOBAL_EXTPROC_CREDENTIAL</code> credential to determine how authentication will be handled. If the <code>GLOBAL_EXTPROC_CREDENTIAL</code> credential is not defined, then the <code>extproc</code> process is authenticated by the operating system privilege of the owners of the Oracle listener or Oracle server process.

12.5 Securing LOBs with LOB Locator Signatures

You can secure large objects (LOB) by regenerating their LOB locator signatures.

- About Securing LOBs with LOB Locator Signatures
 A LOB locator, which is a pointer to the actual location of a large object (LOB) value, can be assigned a signature, which can be used to secure the LOB.
- Managing the Encryption of a LOB Locator Signature Key
 You can use the ALTER DATABASE DICTIONARY SQL statement to encrypt a LOB locator
 signature key.

12.5.1 About Securing LOBs with LOB Locator Signatures

A LOB locator, which is a pointer to the actual location of a large object (LOB) value, can be assigned a signature, which can be used to secure the LOB.

When you create a LOB, Oracle Database automatically assigns a signature to the LOB locator. Oracle Database verifies the signature matches when it receives a locator from a client to ensure that the locator has not been tampered with. Signature-based security can be used for both persistent and temporary LOB locators. It is also used for distributed CLOBs, BLOBs, and NBLOBs that come from index organized table (IOT) locators.

In an Oracle Real Applications Clusters (Oracle RAC) environment, all instances will share the same signature key, which is persisted in the database. In a multitenant environment, each pluggable database (PDB) will have its own signature key. If a LOB locator has been tampered with, the signature verification rejects the LOB and raises an ORA-64219: invalid LOB locator encountered error.

You can encrypt, rekey, and delete the LOB signature key that was used to generate LOB signature for LOB locators that are sent from a standalone database or PDB to a client. If you plan to encrypt the signature key, then the database (or PDB) in which the key resides must have an open TDE keystore.

To enable the LOB signature feature, you must set the <code>LOB_SIGNATURE_ENABLE</code> initialization parameter to <code>TRUE</code>. By default, <code>LOB_SIGNATURE_ENABLE</code> is set to <code>FALSE</code> for Oracle Database release 19c.

12.5.2 Managing the Encryption of a LOB Locator Signature Key

You can use the ALTER DATABASE DICTIONARY SQL statement to encrypt a LOB locator signature key.

- 1. Log in to the database as a user who has ALTER DATABASE DICTIONARY privileges.
- 2. If necessary, enable the LOB signature key feature by setting the LOB_SIGNATURE_ENABLE initialization parameter to TRUE.

```
ALTER SYSTEM SET LOB SIGNATURE ENABLE = TRUE;
```

Alternatively, you can set the LOB_SIGNATURE_ENABLE parameter in the init.ora initialization file before a database restart. This enables the LOB signature key feature for all PDBs.

If you plan to encrypt the signature key, then ensure that the database or PDB has an open TDE keystore.

You must have the SYSKM administrative privilege to create a TDE keystore.

For example, to create and open a software TDE keystore:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl' IDENTIFIED BY password;
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY password;
```

- 4. Execute the ALTER DATABASE DICTIONARY statement to set the LOB signature key configuration.
 - To encrypt the LOB locator signature key instead of obfuscating it, execute the following statement:

```
ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;
```

To regenerate the LOB locator signature key for LOB locators that will be sent to a
client, use the following statement. If the database is in restricted mode, then Oracle
Database regenerates a new LOB signature key to encrypt the regenerated signature
key. If the database is in non-restricted mode, then a new signature key is not
regenerated but instead, Oracle Database uses a new encryption key to encrypt the
existing LOB signature key. Oracle recommends that a database administrator or PDB



administrator run this statement in restricted mode on a periodic basis, preferably during database down time.

ALTER DATABASE DICTIONARY REKEY CREDENTIALS;

 To delete the encrypted LOB locator signature key and then regenerate a new LOB signature key in obfuscated form instead of encrypted form, execute the following statement:

ALTER DATABASE DICTIONARY DELETE CREDENTIALS;

Related Topics

Configuring Transparent Data Encryption

12.6 Managing Application Privileges

Most database applications involve different privileges on different schema objects.

Keeping track of the privileges that are required for each application can be complex. In addition, authorizing users to run an application can involve many GRANT operations.

• To simplify application privilege management, create a role for each application and grant that role all the privileges a user must run the application.

In fact, an application can have several roles, each granted a specific subset of privileges that allow greater or lesser capabilities while running the application.

For example, suppose every administrative assistant uses the Vacation application to record the vacation taken by members of the department. To best manage this application, you should:

- Create a VACATION role.
- 2. Grant all privileges required by the Vacation application to the VACATION role.
- Grant the VACATION role to all administrative assistants. Better yet, create a role that
 defines the privileges the administrative assistants have, and then grant the VACATION role
 to that role.

See Also:

- Configuring Privilege and Role Authorization, for a complete discussion of creating, enabling, and disabling roles, and granting and revoking privileges
- User Privilege and Role Data Dictionary Views for more information about the security uses of the ROLE_TAB_PRIVS, ROLE_SYS_PRIVS, and DBA_ROLE_PRIVS data dictionary views

12.7 Advantages of Using Roles to Manage Application Privileges

Grouping application privileges in a role aids privilege management.

Consider the following administrative options:

- You can grant the role, rather than many individual privileges, to those users who run the
 application. Then, as employees change jobs, you need to grant or revoke only one role,
 rather than many privileges.
- You can change the privileges associated with an application by modifying only the
 privileges granted to the role, rather than the privileges held by all users of the application.
- You can determine the privileges that are necessary to run a particular application by querying the ROLE TAB PRIVS and ROLE SYS PRIVS data dictionary views.
- You can determine which users have privileges on which applications by querying the DBA ROLE PRIVS data dictionary view.

12.8 Creating Secure Application Roles to Control Access to Applications

A secure application role is only enabled through its associated PL/SQL package or procedure.

- Step 1: Create the Secure Application Role

 The CREATE ROLE statement with the IDENTIFIED USING clause creates a secure application role.
- Step 2: Create a PL/SQL Package to Define the Access Policy for the Application You can create a PL/SQL package that defines the access policy for your application.

12.8.1 Step 1: Create the Secure Application Role

The CREATE ROLE statement with the IDENTIFIED USING clause creates a secure application role.

You must have the CREATE ROLE system privilege to execute this statement.

For example, to create a secure application role called hr_admin that is associated with the sec mgr.hr admin package:

Create the security application role as follows:

```
CREATE ROLE hr admin IDENTIFIED USING sec mgr.hr admin role check;
```

This statement indicates the following:

- The role hr admin to be created is a secure application role.
- The role can only be enabled by modules defined inside the PL/SQL procedure sec_mgr.hr_admin_role_check. At this stage, this procedure does not need to exist; Step 2: Create a PL/SQL Package to Define the Access Policy for the Application explains how to create the package or procedure.
- Grant the security application role the privileges you would normally associate with this role.

For example, to grant the hr_admin role SELECT, INSERT, UPDATE, and DELETE privileges on the HR.EMPLOYEES table, you enter the following statement:

```
GRANT SELECT, INSERT, UPDATE, DELETE ON HR.EMPLOYEES TO hr admin;
```

Do not grant the role directly to the user. The PL/SQL procedure or package does that for you, assuming the user passes its security policies.



12.8.2 Step 2: Create a PL/SQL Package to Define the Access Policy for the Application

You can create a PL/SQL package that defines the access policy for your application.

- About Creating a PL/SQL Package to Define the Access Policy for an Application
 To enable or disable the secure application role, you must create the security policies of
 the role within a PL/SQL package.
- Creating a PL/SQL Package or Procedure to Define the Access Policy for an Application
 The PL/SQL package or procedure that you create must use invoker's rights to define the
 access policy.
- Testing the Secure Application Role
 As a user who has been granted the secure application role, try performing an action that requires the privileges the role grants.

12.8.2.1 About Creating a PL/SQL Package to Define the Access Policy for an Application

To enable or disable the secure application role, you must create the security policies of the role within a PL/SQL package.

You also can create an individual procedure to do this, but a package lets you group a set of procedures together. This lets you group a set of policies that, used together, present a solid security strategy to protect your applications. For users (or potential intruders) who fail the security policies, you can add auditing checks to the package to record the failure. Typically, you create this package in the schema of the security administrator.

The package or procedure must accomplish the following:

- It must use invoker's rights to enable the role. To create the package using invoker's
 rights, you must set the AUTHID property to CURRENT_USER. You cannot create the package
 by using definer's rights.
 - For more information about invoker's rights and definer's rights, see *Oracle Database PL/SQL Language Reference*.
- It must include one or more security checks to validate the user. One way to validate users is to use the SYS_CONTEXT SQL function. See Oracle Database SQL Language Reference for more information about SYS_CONTEXT. To find session information for a user, you can use SYS_CONTEXT with an application context. See Using Application Contexts to Retrieve User Information, for details.
- It must issue a SET ROLE SQL statement or DBMS_SESSION.SET_ROLE procedure when the user passes the security checks. Because you create the package using invoker's rights, you must set the role by issuing the SET ROLE SQL statement or the DBMS_SESSION.SET_ROLE procedure. (However, you cannot use the SET ROLE ALL statement for this type of role enablement.) The PL/SQL embedded SQL syntax does not support the SET ROLE statement, but you can invoke SET ROLE by using dynamic SQL (for example, with EXECUTE IMMEDIATE).

For more information about EXECUTE IMMEDIATE, see *Oracle Database PL/SQL Language Reference*.

Because of the way that you must create this package or procedure, you cannot use a logon trigger to enable or disable a secure application role. Instead, invoke the package directly from



the application when the user logs in, before the user must use the privileges granted by the secure application role.

12.8.2.2 Creating a PL/SQL Package or Procedure to Define the Access Policy for an Application

The PL/SQL package or procedure that you create must use invoker's rights to define the access policy.

For example, suppose you wanted to restrict anyone using the hr_admin role to employees who are on site (that is, using certain terminals) and between the hours of 8 a.m. and 5 p.m. As the system or security administrator, you can create a procedure that defines the access policy for the application.

1. Create the procedure as follows:

```
CREATE OR REPLACE PROCEDURE hr_admin_role_check
AUTHID CURRENT_USER

AS
BEGIN

IF (SYS_CONTEXT ('userenv','ip_address')

IN ('192.0.2.10' , '192.0.2.11')

AND

TO_CHAR (SYSDATE, 'HH24') BETWEEN 8 AND 17)
THEN

EXECUTE IMMEDIATE 'SET ROLE hr_admin';
END IF;
END;
/
```

In this example:

- AUTHID CURRENT_USER sets the AUTHID property to CURRENT_USER so that invoker's rights can be used.
- IF (SYS_CONTEXT ('userenv', 'ip_address') validates the user by using the SYS_CONTEXT SQL function to retrieve the user session information.
- BETWEEN ... TO_CHAR creates a test to grant or deny access. The test restricts access
 to users who are on site (that is, using certain terminals) and working between the
 hours of 8:00 a.m. and 5:00 p.m. If the user passes this check, the hr_admin role is
 granted.
- THEN... EXECUTE grants the role to the user by issuing the SET ROLE statement using the EXECUTE IMMEDIATE command, assuming the user passes the test.
- 2. Grant EXECUTE permissions for the hr_admin_role_check procedure to any user who was assigned it.

For example:

```
GRANT EXECUTE ON hr admin role check TO psmith;
```



12.8.2.3 Testing the Secure Application Role

As a user who has been granted the secure application role, try performing an action that requires the privileges the role grants.

When you log in as a user who has been granted the secure application role, the role is then enabled.

1. Log in to the database session as the user.

For example:

```
CONNECT PSMITH@hrpdb
Enter password: password
```

2. Perform an action that requires the privileges the secure application role grants.

For example, if the role grants the EXECUTE privilege for a procedure called sec admin.hr admin role check:

```
EXECUTE sec_admin.hr_admin_role_check;
```

12.9 Association of Privileges with User Database Roles

Ensure that users have only the privileges associated with the current database role.

- Why Users Should Only Have the Privileges of the Current Database Role
 A single user can use many applications and associated roles.
- Use of the SET ROLE Statement to Automatically Enable or Disable Roles
 You can use a SET ROLE statement at the beginning of each application to automatically enable its associated role and to disable all others.

12.9.1 Why Users Should Only Have the Privileges of the Current Database Role

A single user can use many applications and associated roles.

However, you should ensure that the user has only the privileges associated with the current database role.

Consider the following scenario:

- The ORDER role (for an application called Order) contains the UPDATE privilege for the INVENTORY table.
- The INVENTORY role (for an application called Inventory) contains the SELECT privilege for the INVENTORY table.
- Several order entry clerks were granted both the ORDER and INVENTORY roles.

In this scenario, an order entry clerk who was granted both roles can use the privileges of the ORDER role when running the INVENTORY application to update the INVENTORY table. The problem is that updating the INVENTORY table is not an authorized action for the INVENTORY application. It is an authorized action for the ORDER application. To avoid this problem, use the SET ROLE statement as explained in the following section.



12.9.2 Use of the SET ROLE Statement to Automatically Enable or Disable Roles

You can use a SET ROLE statement at the beginning of each application to automatically enable its associated role and to disable all others.

This way, each application dynamically enables particular privileges for a user only when required. The SET ROLE statement simplifies privilege management. You control what information users can access and when they can access it. The SET ROLE statement also keeps users operating in a well-defined privilege domain. If a user obtains privileges only from roles, then the user cannot combine these privileges to perform unauthorized operations.

Related Topics

- How Grants and Revokes Work with SET ROLE and Default Role Settings
 Privilege grants and the SET ROLE statement affect when and how grants and revokes take place.
- When Grants and Revokes Take Effect
 Depending on the privilege that is granted or revoked, a grant or revoke takes effect at different times.

12.10 Protecting Database Objects by Using Schemas

A schema is a security domain that can contain database objects. Privileges granted to users and roles control access to these database objects.

- Protecting Database Objects in a Unique Schema
 Think of most schemas as user names: the accounts that enable users to connect to a database and access the database objects.
- Protection of Database Objects in a Shared Schema
 For many applications, users only need access to an application schema; they do not need their own accounts or schemas in the database.

12.10.1 Protecting Database Objects in a Unique Schema

Think of most schemas as user names: the accounts that enable users to connect to a database and access the database objects.

However, a *unique schema* does not allow connections to the database, but is used to contain a related set of objects. Schemas of this sort are created as typical users, and yet are not granted the CREATE SESSION system privilege (either explicitly or through a role).

To protect the objects, temporarily grant the CREATE SESSION and RESOURCE privilege to a
unique schema if you want to use the CREATE SCHEMA statement to create multiple tables
and views in a single transaction.

For example, a given schema might own the schema objects for a specific application. If application users have the privileges to do so, then they can connect to the database using typical database user names and use the application and the corresponding objects. However, no user can connect to the database using the schema set up for the application. This configuration prevents access to the associated objects through the schema, and provides another layer of protection for schema objects. In this case, the application could issue an ALTER SESSION SET CURRENT_SCHEMA statement to connect the user to the correct application schema.



12.10.2 Protection of Database Objects in a Shared Schema

For many applications, users only need access to an application schema; they do not need their own accounts or schemas in the database.

For example, users John, Firuzeh, and Jane are all users of the Payroll application, and they need access to the payroll schema on the finance database. None of them need to create their own objects in the database. They need to only access the payroll objects. To address this issue, Oracle Database provides the enterprise users, which are schema-independent users.

Enterprise users, users managed in a directory service, do not need to be created as database users because they use a shared database schema. To reduce administration costs, you can create an enterprise user once in the directory, and point the user at a shared schema that many other enterprise users can also access.



Oracle Database Enterprise User Security Administrator's Guide for more information about managing enterprise users

12.11 Object Privileges in an Application

When you design an application, consider the types of users and the level access they need.

- What Application Developers Must Know About Object Privileges
 Object privileges enable end users to perform actions on objects such as tables, views, sequences, procedures, functions, or packages.
- SQL Statements Permitted by Object Privileges
 As you implement and test your application, you should create each necessary role.

12.11.1 What Application Developers Must Know About Object Privileges

Object privileges enable end users to perform actions on objects such as tables, views, sequences, procedures, functions, or packages.

Table 12-3 summarizes the object privileges available for each type of object.

Table 12-3 How Privileges Relate to Schema Objects

Object Privilege	Applies to Table?	Applies to View?	Applies to Sequence?	Applies to Procedure? ¹
ALTER	Yes	No	Yes	No
DELETE	Yes	Yes	No	No
EXECUTE	No	No	No	Yes
INDEX	Yes ²	No	No	No
INSERT	Yes	Yes	No	No
REFERENCES	YesYes(2)	No	No	No

Table 12-3 (Cont.) How Privileges Relate to Schema Objects

Object Privilege	Applies to Table?	Applies to View?	Applies to Sequence?	Applies to Procedure? ¹
SELECT	Yes	Yes ³	Yes	No
UPDATE	Yes	Yes	No	No

¹ Standalone stored procedures, functions, and public package constructs

Related Topics

Auditing Object Actions
 You can use the CREATE AUDIT POLICY statement to audit object actions.

12.11.2 SQL Statements Permitted by Object Privileges

As you implement and test your application, you should create each necessary role.

Test the usage scenario for each role to ensure that the users of your application will have proper access to the database. After completing your tests, coordinate with the administrator of the application to ensure that each user is assigned the proper roles.

Table 12-4 lists the SQL statements permitted by the object privileges shown in Table 12-3.

Table 12-4 SQL Statements Permitted by Database Object Privileges

Object Privilege	SQL Statements Permitted
ALTER	ALTER object (table or sequence)
	CREATE TRIGGER ON object (tables only)
DELETE	DELETE FROM object (table, view, or synonym)
EXECUTE	EXECUTE object (procedure or function)
	References to public package variables
INDEX	CREATE INDEX ON object (table, view, or synonym)
INSERT	INSERT INTO object (table, view, or synonym)
REFERENCES	CREATE or ALTER TABLE statement defining a FOREIGN KEY integrity constraint on object (tables only)
SELECT	SELECTFROM object (table, view, synonym, or snapshot)
	SQL statements using a sequence

Related Topics

About Privileges and Roles

Authorization permits users to access, process, or alter data; it also creates limitations on user access or actions.

Auditing Object Actions

You can use the CREATE AUDIT POLICY statement to audit object actions.



² Privilege that cannot be granted to a role

³ Can also be granted for snapshots

12.12 Parameters for Enhanced Security of Database Communication

Parameters can be used to manage security, such as handling bad packets from protocol errors or configuring the maximum number of authentication errors.

- Bad Packets Received on the Database from Protocol Errors
 The SEC_PROTOCOL_ERROR_TRACE_ACTION initialization parameter controls how trace files are managed when protocol errors are generated.
- Controlling Server Execution After Receiving a Bad Packet
 The SEC_PROTOCOL_ERROR_FURTHER_ACTION initialization parameter controls server execution after the server receives a bad packet.
- Configuration of the Maximum Number of Authentication Attempts
 The SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter sets the number of authentication attempts before the database will drop a failed connection.
- Configuring the Display of the Database Version Banner
 The SEC_RETURN_SERVER_RELEASE_BANNER initialization parameter can be used to prevent the display of detailed product information during authentication.
- Configuring Banners for Unauthorized Access and Auditing User Actions
 The SEC_USER_UNAUTHORIZED_ACCESS_BANNER and SEC_USER_AUDIT_ACTION_BANNER initialization parameters control the display of banners for unauthorized access and for auditing users.

12.12.1 Bad Packets Received on the Database from Protocol Errors

The SEC_PROTOCOL_ERROR_TRACE_ACTION initialization parameter controls how trace files are managed when protocol errors are generated.

Networking communication utilities such as Oracle Call Interface (OCI) or Two-Task Common (TTC) can generate a large disk file containing the stack trace and heap dump when the server receives a bad packet, out-of-sequence packet, or a private or an unused remote procedure call.

Typically, this disk file can grow quite large. An intruder can potentially cripple a system by repeatedly sending bad packets to the server, which can result in disk flooding and Denial of Service (DOS) attacks. An unauthenticated client can also mount this type of attack.

You can prevent these attacks by setting the <code>SEC_PROTOCOL_ERROR_TRACE_ACTION</code> initialization parameter to one of the following values:

 None: Configures the server to ignore the bad packets and does not generate any trace files or log messages. Use this setting if the server availability is overwhelmingly more important than knowing that bad packets are being received.

For example:

```
SEC_PROTOCOL_ERROR_TRACE_ACTION = None
```

Trace (default setting): Creates the trace files, but it is useful for debugging purposes, for example, when a network client is sending bad packets as a result of a bug.

For example:

```
{\tt SEC\_PROTOCOL\_ERROR\_TRACE\_ACTION} = {\tt Trace}
```



• Log: Writes a short, one-line message to the server trace file. This choice balances some level of auditing with system availability.

For example:

```
SEC_PROTOCOL_ERROR_TRACE_ACTION = Log
```

Alert: Sends an alert message to a database administrator or monitoring console.

For example:

```
SEC PROTOCOL ERROR TRACE ACTION = Alert
```

12.12.2 Controlling Server Execution After Receiving a Bad Packet

The SEC_PROTOCOL_ERROR_FURTHER_ACTION initialization parameter controls server execution after the server receives a bad packet.

After Oracle Database detects a client or server protocol error, it must continue execution. However, this could subject the server to further bad packets, which could lead to disk flooding or denial-of-service attacks.

- To control the further execution of a server process when it is receiving bad packets from a potentially malicious client, set the SEC_PROTOCOL_ERROR_FURTHER_ACTION initialization parameter to one of the following values:
 - Continue: Continues the server execution. However, be aware that the server may be subject to further attacks.

For example:

```
SEC PROTOCOL ERROR FURTHER ACTION = Continue
```

 (Delay, m): Delays the client m seconds before the server can accept the next request from the same client connection. This setting prevents malicious clients from excessively using server resources while legitimate clients experience a degradation in performance but can continue to function. When you enter this setting, enclose it in parentheses.

For example:

```
SEC PROTOCOL ERROR FURTHER ACTION = (Delay, 3)
```

If you are setting SEC_PROTOCOL_ERROR_FURTHER_ACTION by using the ALTER SYSTEM or ALTER SESSION SQL statement, then you must enclose the Delay setting in either single or double quotation marks.

```
ALTER SYSTEM SEC PROTOCOL ERROR FURTHER ACTION = '(Delay, 3)';
```

— (Drop, n): Forcefully terminates the client connection after n bad packets. This setting enables the server to protect itself at the expense of the client, for example, loss of a transaction. However, the client can still reconnect, and attempt the same operation again. Enclose this setting in parentheses. The default value of SEC PROTOCOL ERROR FURTHER ACTION is (Drop, 3).

For example:

```
SEC PROTOCOL ERROR FURTHER ACTION = (Drop, 10)
```

Similar to the Delay setting, you must enclose the Drop setting in single or double quotation marks if you are using ALTER SYSTEM or ALTER SESSION to change this setting.



12.12.3 Configuration of the Maximum Number of Authentication Attempts

The SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter sets the number of authentication attempts before the database will drop a failed connection.

As part of connection creation, the listener starts the server process and attaches it to the client. Using this physical connection, the client is this able to authenticate the connection. After a server process starts, client authenticates with this server process. An intruder could start a server process, and then issue an unlimited number of authenticated requests with different user names and passwords in an attempt to gain access to the database.

You can limit the number of failed login attempts for application connections by setting the SEC_MAX_FAILED_LOGIN_ATTEMPTS initialization parameter to restrict the number of authentication attempts on a connection. After the specified number of authentication attempts fail, the database process drops the connection and the server process is terminated. By default, SEC MAX FAILED LOGIN ATTEMPTS is set to 3.

Remember that the <code>SEC_MAX_FAILED_LOGIN_ATTEMPTS</code> initialization parameter is designed to prevent potential intruders from attacking your applications, as well as valid users who have forgotten their passwords. The <code>sqlnet.oraINBOUND_CONNECT_TIMEOUT</code> parameter and the <code>FAILED_LOGIN_ATTEMPTS</code> profile parameter also restrict failed logins, but the difference is that these two parameters only apply to valid user accounts.

For example, to limit the maximum attempts to 5, set <code>SEC_MAX_FAILED_LOGIN_ATTEMPTS</code> as follows in the <code>initsid.ora</code> initialization parameter file:

SEC_MAX_FAILED_LOGIN_ATTEMPTS = 5

12.12.4 Configuring the Display of the Database Version Banner

The SEC_RETURN_SERVER_RELEASE_BANNER initialization parameter can be used to prevent the display of detailed product information during authentication.

Detailed product version information should not be accessible before a client connection (including an Oracle Call Interface client) is authenticated. An intruder could use the database version to find information about security vulnerabilities that may be present in the database software.

• To restrict the display of the database version banner to unauthenticated clients, set the SEC_RETURN_SERVER_RELEASE_BANNER initialization parameter in the initsid.ora initialization parameter file to either TRUE or FALSE.

By default, SEC RETURN SERVER RELEASE BANNER is set to False.

For example, if you set it to TRUE, then Oracle Database displays the full correct database version. For example, for Release 19.1.0.0:

```
Oracle Database 19c Enterprise Edition Release 19.1.0.0 - Production
```

If a release number uses point release notation (for example, Oracle Database Release 19.1.0.1), then the banner displays as follows:

```
Oracle Database 19c Enterprise Edition Release 19.1.0.1 - Production
```

However, if in that same release, you set it to NO, then Oracle Database restricts the banner to display the following fixed text starting with Release 19.1, which instead of 19.1.0.1 is 19.1.0.0.0:



Oracle Database 19c Release 19.1.0.0.0 - Production

12.12.5 Configuring Banners for Unauthorized Access and Auditing User Actions

The <code>SEC_USER_UNAUTHORIZED_ACCESS_BANNER</code> and <code>SEC_USER_AUDIT_ACTION_BANNER</code> initialization parameters control the display of banners for unauthorized access and for auditing users.

You should create and configure banners to warn users against unauthorized access and possible auditing of user actions. The notices are available to the client application when it logs into the database.

- To configure these banners to display, set the following sqlnet.ora parameters on the database server side to point to a text file that contains the banner information:
 - SEC_USER_UNAUTHORIZED_ACCESS_BANNER. For example:
 SEC_USER_UNAUTHORIZED_ACCESS_BANNER = /opt/Oracle/12c/dbs/unauthaccess.txt
 - SEC_USER_AUDIT_ACTION_BANNER. For example:
 SEC_USER_AUDIT_ACTION_BANNER = /opt/Oracle/12c/dbs/auditactions.txt

By default, these parameters are not set. In addition, be aware that there is a 512-byte limitation for the number of characters used for the banner text.

After you set these parameters, the Oracle Call Interface application must use the appropriate OCI APIs to retrieve these banners and present them to the end user.



Part III

Controlling Access to Data

Part III describes how to control access to data.

- Using Application Contexts to Retrieve User Information
 An application context stores user identification that can enable or prevent a user from accessing data in the database.
- Using Oracle Virtual Private Database to Control Data Access
 Oracle Virtual Private Database (VPD) enables you to filter users who access data.
- Using Transparent Sensitive Data Protection
- Encryption of Sensitive Credential Data in the Data Dictionary
 You can encrypt sensitive credential information, such as passwords that are stored in the
 data dictionary.
- Manually Encrypting Data
 You can use the DBMS_CRYPTO PL/SQL package to manually encrypt data.



Using Application Contexts to Retrieve User Information

An application context stores user identification that can enable or prevent a user from accessing data in the database.

About Application Contexts

An application context provides many benefits in controlling the access that a user has to data.

Types of Application Contexts

There are three general categories of application contexts.

Using Database Session-Based Application Contexts

A database session-based application context enables you to retrieve session-based information about a user.

Global Application Contexts

You can use a global application context to access application values across database sessions, including an Oracle Real Application Clusters environment.

Using Client Session-Based Application Contexts

A client session-based application context is stored in the User Global Area (UGA).

Application Context Data Dictionary Views

Oracle Database provides data dictionary views that provide information about application contexts.

13.1 About Application Contexts

An application context provides many benefits in controlling the access that a user has to data.

What Is an Application Context?

An **application context** is a set of name-value pairs that Oracle Database stores in memory.

Components of the Application Context

An application context has two components, comprising a name-value pair.

Where Are the Application Context Values Stored?

Oracle Database stores the application context values in a secure data cache.

Benefits of Using Application Contexts

Most applications contain the kind of information that can be used for application contexts.

How Editions Affects Application Context Values

Oracle Database sets the application context in all editions that are affected by the application context package.

• Application Contexts in a Multitenant Environment

Where you create an application in a multitenant environment determines where you must create the application context.

13.1.1 What Is an Application Context?

An application context is a set of name-value pairs that Oracle Database stores in memory.

The context has a label called a **namespace** (for example, <code>empno_ctx</code> for an application context that retrieves employee IDs). This context enables Oracle Database to find information about both database and nondatabase users during authentication.

Inside the context are the name-value pairs (an associative array): the name points to a location in memory that holds the value. An application can use the application context to access session information about a user, such as the user ID or other user-specific information, or a client ID, and then securely pass this data to the database.

You can then use this information to either permit or prevent the user from accessing data through the application. You can use application contexts to authenticate both database and non-database users.

Related Topics

Auditing Application Context Values
 You can use the AUDIT statement to audit application context values.

13.1.2 Components of the Application Context

An application context has two components, comprising a name-value pair.

These components are as follows:

- Name. Refers to the name of the attribute set that is associated with the value. For example, if the <code>empno_ctx</code> application context retrieves an employee ID from the <code>HR.EMPLOYEES</code> table, it could have a name such as <code>employee id</code>.
- Value. Refers to a value set by the attribute. For example, for the empno_ctx application context, if you wanted to retrieve an employee ID from the HR.EMPLOYEES table, you could create a value called emp_id that sets the value for this ID.

Think of an application context as a global variable that holds information that is accessed during a database session. To set the values for a secure application context, you must create a PL/SQL package procedure that uses the <code>DBMS_SESSION.SET_CONTEXT</code> procedure. In fact, this is the only way that you can set application context values if the context is not marked <code>INITIALIZED EXTERNALLY</code> or <code>INITIALIZED GLOBALLY</code>. You can assign the values to the application context attributes at run time, not when you create the application context. Because the **trusted** procedure, and not the user, assigns the values, it is a called secure application context. For client-session based application contexts, another way to set the application context is to use Oracle Call Interface (OCI) calls.

13.1.3 Where Are the Application Context Values Stored?

Oracle Database stores the application context values in a secure data cache.

This cache is available in the User Global Area (UGA) or the System (sometimes called "Shared") Global Area (SGA). This way, the application context values are retrieved during the session. Because the application context stores the values in this data cache, it increases performance for your applications. You can use an application context by itself, with Oracle Virtual Private Databases policies, or with other fine-grained access control policies.



Related Topics

Oracle Virtual Private Database Use with an Application Context
 You can use application contexts with Oracle Virtual Private Database policies.

13.1.4 Benefits of Using Application Contexts

Most applications contain the kind of information that can be used for application contexts.

For example, in an order entry application that uses a table containing the columns <code>ORDER_NUMBER</code> and <code>CUSTOMER_NUMBER</code>, you can use the values in these columns as security attributes to restrict access by a customer to his or her own orders, based on the ID of that customer.

Application contexts are useful for the following purposes:

- Enforcing fine-grained access control (for example, in Oracle Virtual Private Database polices)
- Preserving user identity across multitier environments
- Enforcing stronger security for your applications, because the application context is controlled by a trusted procedure, not the user
- Increasing performance by serving as a secure data cache for attributes needed by an application for fine-grained auditing or for use in PL/SQL conditional statements or loops
 - This cache saves the repeated overhead of querying the database each time these attributes are needed. Because the application context stores session data in cache rather than forcing your applications to retrieve this data repeatedly from a table, it greatly improves the performance of your applications.
- Serving as a holding area for name-value pairs that an application can define, modify, and access

13.1.5 How Editions Affects Application Context Values

Oracle Database sets the application context in all editions that are affected by the application context package.

The values the application context sets are visible in all editions the application context affects. To find all editions in your database, and whether they are usable, you can query the <code>ALL_EDITIONS</code> data dictionary view.



Oracle Database Development Guide for detailed information about editions

13.1.6 Application Contexts in a Multitenant Environment

Where you create an application in a multitenant environment determines where you must create the application context.



If an application is installed in the application root or CDB root, then it becomes accessible across the application container or system container and associated application PDBs. You will need to create a common application context in this root.

When you create a common application context for use with an application container, note the following:

- You can create application contexts in a multitenant environment by setting the CONTAINER
 clause in the CREATE CONTEXT SQL statement. For example, to create a common
 application context in the application root, you must execute CREATE CONTEXT with
 CONTAINER set to ALL. To create the application context in a PDB, set CONTAINER to
 CURRENT.
- You cannot use the same name for a local application context for a common application context. You can find the names of existing application contexts by running the following query:

```
SELECT OBJECT NAME FROM DBA OBJECTS WHERE OBJECT TYPE = CONTEXT';
```

- The PL/SQL package that you create to manage a common application context must be a common PL/SQL package. That is, it must exist in the application root or CDB root. If you create the application context for a specific PDB, then you must store the associated PL/SQL package in that PDB.
- The name-value pairs that you set under a common session application context from an application container or a system container for a common application context are not accessible from other application containers or system containers when a common user accesses a different container.
- The name-value pairs that you set under a common global application context from an application container or a system container, are accessible only within the same container in the same user session.
- An application can retrieve the value of an application context whether it resides in the application root, the CDB root, or a PDB.
- During a plug-in operation of a PDB into a CDB or an application container, if the name of the common application context conflicts with a PDB's local application context, then the PDB must open in restricted mode. A database administrator would then need to correct the conflict before opening the PDB in normal mode.
- During an unplug operation, a common application context retains its common semantics, so that later on, if the PDB is plugged into another CDB where a common application context with the same name exists, it would continue to behave like a common object. If a PDB is plugged into an application container or a system container, where the same common application context does not exist, then it behaves like a local object.

To find if an application context is a local application context or an application common application context, query the SCOPE column of the DBA_CONTEXT or ALL_CONTEXT data dictionary view.

13.2 Types of Application Contexts

There are three general categories of application contexts.

These categories are as follows:

 Database session-based application contexts. This type retrieves data that is stored in the database user session (that is, the UGA) cache. There are three categories of database session-based application contexts:

- Initialized locally. Initializes the application context locally, to the session of the user.
- Initialized externally. Initializes the application context from an Oracle Call Interface (OCI) application, a job queue process, or a connected user database link.
- Initialized globally. Uses attributes and values from a centralized location, such as an LDAP directory.

Using Database Session-Based Application Contexts describes this type of application context.

Global application contexts. This type retrieves data that is stored in the System Global
Area (SGA) so that it can be used for applications that use a sessionless model, such as
middle-tier applications in a three-tiered architecture. A global application context is useful
if the session context must be shared across sessions, for example, through connection
pool implementations.

Global Application Contexts describes this type.

 Client session-based application contexts. This type uses Oracle Call Interface functions on the client side to set the user session data, and then to perform the necessary security checks to restrict user access.

Using Client Session-Based Application Contexts describes this type.

Table 13-1 summarizes the different types of application contexts.

Table 13-1 Types of Application Contexts

Application Context Type	Stored in UGA	Stored in SGA	Supports Connected User Database Links	Supports Centralized Storage of Users' Application Context	Supports Sessionless Multitier Applications
Database session-based application context initialized locally	Yes	No	No	No	No
Database session-based application context initialized externally	Yes	No	Yes	No	No
Database session-based application context initialized globally	Yes	No	No	Yes	No
Global application context	No	Yes	No	No	Yes
Client session-based application context	Yes	No	Yes	No	Yes

13.3 Using Database Session-Based Application Contexts

A database session-based application context enables you to retrieve session-based information about a user.

- About Database Session-Based Application Contexts
 A database session-based application context retrieves session information for database users.
- Components of a Database Session-Based Application Context
 A database session-based application context retrieves and sets data for the context and then sets this context when a user logs in.

- Creating Database Session-Based Application Contexts
 A database session-based application context is a named object that stores the user's session information.
- Creating a Package to Set a Database Session-Based Application Context
 A PL/SQL package can be used to retrieve the session information and set the name-value attributes of the application context.
- Logon Triggers to Run a Database Session Application Context Package
 Users must run database session application context package after when they log in to the
 database instance.
- Example: Creating a Simple Logon Trigger
 The CREATE TRIGGER statement can create a simple logon trigger.
- Example: Creating a Logon Trigger for a Production Environment
 The CREATE TRIGGER statement can create a logon trigger for a production environment.
- Example: Creating a Logon Trigger for a Development Environment
 The CREATE TRIGGER statement can create a logon trigger for a development environment.
- Tutorial: Creating and Using a Database Session-Based Application Context
 This tutorial demonstrates how to create an application context that checks the ID of users
 who try to log in to the database.
- Initializing Database Session-Based Application Contexts Externally
 Initializing database session-based application contexts externally increases performance
 because the application context is stored in the user global area (UGA).
- Initializing Database Session-Based Application Contexts Globally
 When a database session-based application is stored in a centralized location, it can be used globally from an LDAP directory.
- Externalized Database Session-Based Application Contexts
 Many applications store attributes used for fine-grained access control within a database metadata table.

13.3.1 About Database Session-Based Application Contexts

A database session-based application context retrieves session information for database users.

This type of application context uses a PL/SQL procedure within Oracle Database to retrieve, set, and secure the data it manages.

The database session-based application context is managed entirely within Oracle Database. Oracle Database sets the values, and then when the user exits the session, automatically clears the application context values stored in cache. If the user connection ends abnormally, for example, during a power failure, then the PMON background process cleans up the application context data. You do not need to explicitly clear the application context from cache.

The advantage of having Oracle Database manage the application context is that you can centralize the application context management. Any application that accesses this database will need to use this application context to permit or prevent user access to that application. This provides benefits both in improved performance and stronger security.





If your users are application users, that is, users who are not in your database, consider using a global application context instead.

Related Topics

 Global Application Contexts
 You can use a global application context to access application values across database sessions, including an Oracle Real Application Clusters environment.

13.3.2 Components of a Database Session-Based Application Context

A database session-based application context retrieves and sets data for the context and then sets this context when a user logs in.

You must use three components to create and use a database session-based application context: the application context, a procedure to retrieve the data and set the context, and a way to set the context when the user logs in.

- The application context. You use the CREATE CONTEXT SQL statement to create an
 application context. This statement names the application context (namespace) and
 associates it with a PL/SQL procedure that is designed to retrieve session data and set the
 application context.
- A PL/SQL procedure to perform the data retrieval and set the context. About the
 Package That Manages the Database Session-Based Application Context describes the
 tasks this procedure must perform. Ideally, create this procedure within a package, so that
 you can include other procedures if you want (for example, to perform error checking
 tasks).
- A way to set the application context when the user logs on. Users who log on to applications that use the application context must run a PL/SQL package that sets the application context. You can achieve this with either a logon trigger that fires each time the user logs on, or you can embed this functionality in your applications.

Tutorial: Creating and Using a Database Session-Based Application Context shows how to create and use a database session-based application context that is initialized locally.

In addition, you can initialize session-based application contexts either externally or globally. Either method stores the context information in the user session.

- External initialization. This type can come from an OCI interface, a job queue process, or a connected user database link. See Initializing Database Session-Based Application Contexts Externally for detailed information.
- Global initialization. This type uses attributes and values from a centralized location, such as an LDAP directory. Initializing Database Session-Based Application Contexts Globally provides more information.

13.3.3 Creating Database Session-Based Application Contexts

A database session-based application context is a named object that stores the user's session information.



- About Creating Database Session-Based Application Contexts
 A database user session (UGA) stores session-based application context, using a user-created namespace.
- Creating a Database Session-Based Application Context
 The CREATE CONTEXT SQL statement can be used to create a database session-based application context.
- Database Session-Based Application Contexts for Multiple Applications
 For each application, you can create an application context that has its own attributes.

13.3.3.1 About Creating Database Session-Based Application Contexts

A database user session (UGA) stores session-based application context, using a user-created namespace.

Each application context must have a unique attribute and belong to a namespace. That is, context names must be unique within the database, not just within a schema.

You must have the CREATE ANY CONTEXT system privilege to create an application context, and the DROP ANY CONTEXT privilege to use the DROP CONTEXT statement if you want to drop the application context.

The ownership of the application context is as follows: Even though a user who has been granted the CREATE ANY CONTEXT and DROP ANY CONTEXT privileges can create and drop the application context, it is owned by the SYS schema. Oracle Database associates the context with the schema account that created it, but if you drop this user, the context still exists in the SYS schema. As user SYS, you can drop the application context.

You can find the names of existing application contexts by running the following query:

SELECT OBJECT NAME FROM DBA OBJECTS WHERE OBJECT TYPE = 'CONTEXT';

13.3.3.2 Creating a Database Session-Based Application Context

The CREATE CONTEXT SQL statement can be used to create a database session-based application context.

When you create a database session-based application context, you must create a namespace for the application context and then associate it with a PL/SQL package that manages the name-value pair that holds the session information of the user. At the time that you create the context, the PL/SQL package does not need to exist, but it must exist at run time.

 To create a database session-based application context, use the CREATE CONTEXT SQL statement.

For example:

```
CREATE CONTEXT empno ctx USING set empno ctx pkg CONTAINER = CURRENT;
```

In this example:

- empno ctx is the context namespace.
- set_empno_ctx_pkg is the package (which does not need to exist when you create the
 context) that sets attributes for the empno_ctx namespace. Step 3: Create a Package to
 Retrieve Session Data and Set the Application Context shows an example of how to create
 a package that can be used with this application context.
- CONTAINER creates the application context in the current PDB. To create the application context in the application or CDB root, you must set CONTAINER to ALL.

Notice that when you create the context, you do not set its name-value attributes in the CREATE CONTEXT statement. Instead, you set these in the PL/SQL package that you associate with the application context. The reason you must do this is to prevent a malicious user from changing the context attributes without proper attribute validation. Ensure that this package is in the same container as the application context. For example, if you created the application context in a PDB, then the PL/SQL package must reside in that PDB.



You cannot create a context called CLIENTCONTEXT. This word is reserved for use with client session-based application contexts. See Using Client Session-Based Application Contexts for more information about this type of application context.

13.3.3.3 Database Session-Based Application Contexts for Multiple Applications

For each application, you can create an application context that has its own attributes.

Suppose, for example, you have three applications: General Ledger, Order Entry, and Human Resources.

You can specify different attributes for each application:

- For the order entry application context, you could specify the attribute CUSTOMER NUMBER.
- For the general ledger application context, you could specify the attributes SET_OF_BOOKS and TITLE.
- For the human resources application context, you could specify the attributes ORGANIZATION ID, POSITION, and COUNTRY.

The data the attributes access is stored in the tables behind the applications. For example, the order entry application uses a table called <code>OE.CUSTOMERS</code>, which contains the <code>CUSTOMER_NUMBER</code> column, which provides data for the <code>CUSTOMER_NUMBER</code> attribute. In each case, you can adapt the application context to your precise security needs.

13.3.4 Creating a Package to Set a Database Session-Based Application Context

A PL/SQL package can be used to retrieve the session information and set the name-value attributes of the application context.

- About the Package That Manages the Database Session-Based Application Context
 This defines procedures that manage the session data represented by the application
 context.
- Using the SYS_CONTEXT Function to Retrieve Session Information
 You can retrieve session information for the application context by using the SYS_CONTEXT
 function.
- Checking the SYS_CONTEXT Settings
 You can check the SYS_CONTEXT settings, which are stored in the DUAL table.
- Dynamic SQL with SYS_CONTEXT
 During a session in which you expect a change in policy between executions of a given query, the query must use dynamic SQL.



- SYS CONTEXT in a Parallel Query
 - If you use SYS_CONTEXT inside a SQL function that is embedded in a parallel query, then the function includes the application context.
- SYS_CONTEXT with Database Links
 The SYS_CONTEXT function is compatible with the use of database links.
- DBMS_SESSION.SET_CONTEXT for Setting Session Information
 After SYS_CONTEXT retrieves the session data of a user, you can set the application context values from the user session.
- Example: Simple Procedure to Create an Application Context Value You can use the DBMS_SESSION.SET_CONTEXT statement in a procedure to set an application context value.

13.3.4.1 About the Package That Manages the Database Session-Based Application Context

This defines procedures that manage the session data represented by the application context.

This package is usually created in the security administrator schema. The package must perform the following tasks:

- Retrieve session information. To retrieve the user session information, you can use the SYS_CONTEXT SQL function. The SYS_CONTEXT function returns the value of the parameter associated with the context namespace. You can use this function in both SQL and PL/SQL statements. Typically, you will use the built-in USERENV namespace to retrieve the session information of a user. You also can use the SYS_SESSION_ROLES namespace to indicate whether the specified role is currently enabled for the session.
- Set the name-value attributes of the application context you created with CREATE CONTEXT. You can use the DBMS_SESSION.SET_CONTEXT procedure to set the name-value attributes of the application context. The name-value attributes can hold information such as the user ID, IP address, authentication mode, the name of the application, and so on. The values of the attributes you set remain either until you reset them, or until the user ends the session. Note the following:
 - If the value of the parameter in the namespace already has been set, then SET_CONTEXT overwrites this value.
 - Be aware that any changes in the context value are reflected immediately and subsequent calls to access the value through the SYS_CONTEXT function will return the most recent value.
- Be executed by users. After you create the package, the user will need to execute the
 package when he or she logs on. You can create a logon trigger to execute the package
 automatically when the user logs on, or you can embed this functionality in your
 applications. Remember that the application context session values are cleared
 automatically when the user ends the session, so you do not need to manually remove the
 session data.

It is important to remember that the procedure is a trusted procedure: It is designed to prevent the user from setting his or her own application context attribute values. The user runs the procedure, but the procedure sets the application context values, not the user.



See Also:

- Oracle Database SQL Language Reference for detailed information about the SYS CONTEXT function
- Tutorial: Creating and Using a Database Session-Based Application Context shows how to create a database session-based application context

13.3.4.2 Using the SYS_CONTEXT Function to Retrieve Session Information

You can retrieve session information for the application context by using the SYS_CONTEXT function.

The SYS_CONTEXT function provides a default namespace, USERENV, which describes the current session of the user logged on. SYS_CONTEXT enables you to retrieve different types of session-based information about a user, such as the user host computer ID, host IP address, operating system user name, and so on. Remember that you only use USERENV to retrieve session data, not set it. The predefined attributes are listed in the description for the PL/SQL function in the Oracle Database SQL Language Reference.

• To use retrieve session information, set the namespace, parameter, and optionally, the length values of the SYS CONTEXT function.

For example:

```
SYS CONTEXT ('USERENV', 'HOST')
```

The syntax for the PL/SQL function SYS CONTEXT is as follows:

```
SYS CONTEXT ('namespace', 'parameter'[,length])
```

In this specification:

- namespace is the name of the application context. You can specify either a string or an
 expression that evaluates to a string. The SYS_CONTEXT function returns the value of
 parameter associated with the context namespace at the current instant. If the value of the
 parameter in the namespace already has been set, then SET_CONTEXT overwrites this
 value.
- parameter is a parameter within the namespace application context. This value can be a string or an expression.
- *length* is the default maximum size of the return type, which is 256 bytes, but you can override the length by specifying a value up to 4000 bytes. Enter a value that is a NUMBER data type, or a value that can be can be implicitly converted to NUMBER. The data type of the SYS CONTEXT return type is a VARCHAR2. This setting is optional.

Note:

The USERENV application context namespace replaces the USERENV function provided in earlier Oracle Database releases.



13.3.4.3 Checking the SYS CONTEXT Settings

You can check the SYS CONTEXT settings, which are stored in the DUAL table.

The DUAL table is a small table in the data dictionary that Oracle Database and user-written programs can reference to guarantee a known result. This table has one column called DUMMY and one row that contains the value x.

To check the SYS CONTEXT settings, issue a SELECT SQL statement on the DUAL table.

For example, to find the host computer on which you are logged, assuming that you are logged on to the Shobeen PC host computer under EMP USERS:

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

SYS_CONTEXT (USERENV, HOST)
-----
EMP_USERS\SHOBEEEN_PC
```

13.3.4.4 Dynamic SQL with SYS CONTEXT

During a session in which you expect a change in policy between executions of a given query, the query must use dynamic SQL.

You must use dynamic SQL because static SQL and dynamic SQL parse statements differently:

- Static SQL statements are parsed at compile time. They are not parsed again at execution time for performance reasons.
- Dynamic SQL statements are parsed every time they are executed.

Consider a situation in which Policy A is in force when you compile a SQL statement, and then you switch to Policy B and run the statement. With static SQL, Policy A remains in force. Oracle Database parses the statement at compile time, but does not parse it again upon execution. With dynamic SQL, Oracle Database parses the statement upon execution, then the switch to Policy B takes effect.

For example, consider the following policy:

```
EMPLOYEE NAME = SYS CONTEXT ('USERENV', 'SESSION USER')
```

The policy EMPLOYEE_NAME matches the database user name. It is represented in the form of a SQL predicate in Oracle Virtual Private Database: the predicate is considered a policy. If the predicate changes, then the statement must be parsed again to produce the correct result.

Related Topics

Automatic Reparsing for Fine-Grained Access Control Policies Functions
 Queries against objects enabled with fine-grained access control run the policy function so that the most current predicate is used for each policy.

13.3.4.5 SYS_CONTEXT in a Parallel Query

If you use SYS_CONTEXT inside a SQL function that is embedded in a parallel query, then the function includes the application context.

Consider a user-defined function within a SQL statement, which sets the user ID to 5:

```
CREATE FUNCTION set_id
RETURN NUMBER IS
BEGIN
IF SYS_CONTEXT ('hr', 'id') = 5
   THEN RETURN 1; ELSE RETURN 2;
END IF;
END;
```

Now consider the following statement:

```
SELECT * FROM emp WHERE set id() = 1;
```

When this statement is run as a parallel query, the user session, which contains the application context information, is propagated to the parallel execution servers (query child processes).

13.3.4.6 SYS_CONTEXT with Database Links

The SYS CONTEXT function is compatible with the use of database links.

When SQL statements within a user session involve database links, Oracle Database runs the SYS_CONTEXT function at the host computer of the database link, and then captures the context information in the host computer.

If remote PL/SQL procedure calls are run on a database link, then Oracle Database runs any SYS CONTEXT function inside such a procedure at the destination database of the link.

In this case, only externally initialized application contexts are available at the database link destination site. For security reasons, Oracle Database propagates only the externally initialized application context information to the destination site from the initiating database link site.

13.3.4.7 DBMS_SESSION.SET_CONTEXT for Setting Session Information

After SYS_CONTEXT retrieves the session data of a user, you can set the application context values from the user session.

To set the context values, you can use the <code>DBMS_SESSION.SET_CONTEXT</code> procedure. You must have the <code>EXECUTE</code> privilege for the <code>DBMS_SESSION</code> PL/SQL package.

The syntax for DBMS SESSION.SET CONTEXT is as follows:

```
DBMS_SESSION.SET_CONTEXT (
namespace VARCHAR2,
attribute VARCHAR2,
value VARCHAR2,
username VARCHAR2,
client id VARCHAR2);
```

In this specification:

 namespace is the namespace of the application context to be set, limited to 30 bytes. For example, if you were using a namespace called custno_ctx, you would specify it as follows:

```
namespace => 'custno ctx',
```

 attribute is the attribute of the application context to be set, limited to 30 bytes. For example, to create the ctx attrib attribute for the custno ctx namespace:

```
attribute => 'ctx_attrib',
```



• value is the value of the application context to be set, limited to 4000 bytes. Typically, this is the value retrieved by the SYS CONTEXT function and stored in a variable. For example:

```
value => ctx_value,
```

username is the database user name attribute of the application context. The default is
 NULL, which permits any user to access the session. For database session-based
 application contexts, omit this setting so that it uses the NULL default. This setting is
 optional.

The username and client_id parameters are used for globally accessed application contexts. See DBMS_SESSION.SET_CONTEXT username and client_id Parameters for more information.

• client_id is the application-specific client_id attribute of the application context (64-byte maximum). The default is NULL, which means that no client ID is specified. For database session-based application contexts, omit this setting so that it uses the NULL default.

See Also:

- Tutorial: Creating and Using a Database Session-Based Application Context for how to create a package that retrieves the user session information and then sets the application context based on this information
- Oracle Database PL/SQL Packages and Types Reference for detailed information about the DBMS SESSION.SET CONTEXT procedure
- Oracle Database PL/SQL Packages and Types Reference for detailed information about the DBMS SESSION package

13.3.4.8 Example: Simple Procedure to Create an Application Context Value

You can use the DBMS_SESSION.SET_CONTEXT statement in a procedure to set an application context value.

Example 13-1 shows how to create a simple procedure that creates an attribute for the empno ctx application context.

Example 13-1 Simple Procedure to Create an Application Context Value

```
CREATE OR REPLACE PROCEDURE set_empno_ctx_proc(
  emp_value IN VARCHAR2)
IS
BEGIN
  DBMS_SESSION.SET_CONTEXT('empno_ctx', 'empno_attrib', emp_value);
END;
//
```

In this example:

- emp_value IN VARCHAR2 takes emp_value as the input parameter. This parameter specifies
 the value associated with the application context attribute empno_attrib. The limit is 4000
 bytes.
- DBMS_SESSION.SET_CONTEXT('empno_ctx', 'empno_attrib', emp_value) sets the value of the application context by using the DBMS SESSION.SET CONTEXT procedure as follows:

- 'empno_ctx' refers to the application context namespace. Enclose its name in single quotation marks.
- 'empno_attrib' creates the attribute associated with the application context namespace.
- emp_value specifies the value for the empno_attrib attribute. Here, it refers to the emp value parameter.

At this stage, you can run the set_empno_ctx_proc procedure to set the application context:

```
EXECUTE set empno ctx proc ('42783');
```

(In a real world scenario, you would set the application context values in the procedure itself, so that it becomes a trusted procedure. This example is only used to show how data can be set for demonstration purposes.)

To check the application context setting, run the following SELECT statement:

You can also query the SESSION_CONTEXT data dictionary view to find all the application context settings in the current session of the database instance. For example:

13.3.5 Logon Triggers to Run a Database Session Application Context Package

Users must run database session application context package after when they log in to the database instance.

You can create a logon trigger that handles this automatically. You do not need to grant the user EXECUTE permissions to run the package.

Note the following:

- If the PL/SQL package procedure called by the logon trigger has any unhandled exceptions or raises any exceptions (because, for example, a security check failed), then the logon trigger fails. When the logon trigger fails, the logon fails, that is, the user is denied permission to log in to the database.
- Logon triggers may affect performance. In addition, test the logon trigger on a sample schema user first before creating it for the database. That way, if there is an error, you can easily correct it.
- Be aware of situations in which if you have a changing set of books, or if positions change constantly. In these cases, the new attribute values may not be picked up right away, and you must force a cursor reparse to pick them up.





A logon trigger can be used because the user context (information such as EMPNO, GROUP, MANAGER) should be set before the user accesses any data.

13.3.6 Example: Creating a Simple Logon Trigger

The CREATE TRIGGER statement can create a simple logon trigger.

Example 13-2 shows a simple logon trigger that executes a PL/SQL procedure.

Example 13-2 Creating a Simple Logon Trigger

```
CREATE OR REPLACE TRIGGER set_empno_ctx_trig AFTER LOGON ON DATABASE BEGIN sec_mgr.set_empno_ctx_proc; END:
```

13.3.7 Example: Creating a Logon Trigger for a Production Environment

The CREATE TRIGGER statement can create a logon trigger for a production environment.

Example 13-3 shows how to create a logon trigger that uses a WHEN OTHERS exception. Otherwise, if there is an error in the PL/SQL logic that creates an unhandled exception, then all connections to the database are blocked.

This example shows a WHEN OTHERS exception that writes errors to a table in the security administrator's schema. In a production environment, this is safer than sending the output to the user session, where it could be vulnerable to security attacks.

Example 13-3 Creating a Logon Trigger for a Production Environment

```
CREATE OR REPLACE TRIGGER set_empno_ctx_trig AFTER LOGON ON DATABASE
BEGIN

sec_mgr.set_empno_ctx_proc;
EXCEPTION

WHEN OTHERS THEN

v_code := SQLCODE;

v_errm := SUBSTR(SQLERRM, 1 , 64);

-- Invoke another procedure,

-- declared with PRAGMA AUTONOMOUS_TRANSACTION,

-- to insert information about errors.

INSERT INTO sec_mgr.errors VALUES (v_code, v_errm, SYSTIMESTAMP);
END;

/
```

13.3.8 Example: Creating a Logon Trigger for a Development Environment

The CREATE TRIGGER statement can create a logon trigger for a development environment.

Example 13-4 shows how to create the same logon trigger for a development environment, in which you may want to output errors the user session for debugging purposes.

Example 13-4 Creating a Logon Trigger for a Development Environment

```
CREATE TRIGGER set_empno_ctx_trig
AFTER LOGON ON DATABASE
```



```
BEGIN
    sysadmin_ctx.set_empno_ctx_pkg.set_empno;
EXCEPTION
    WHEN OTHERS THEN
    RAISE_APPLICATION_ERROR(
          -20000, 'Trigger sysadmin_ctx.set_empno_ctx_trig violation. Login denied.');
END;
```

13.3.9 Tutorial: Creating and Using a Database Session-Based Application Context

This tutorial demonstrates how to create an application context that checks the ID of users who try to log in to the database.

- Step 1: Create User Accounts and Ensure the User SCOTT Is Active
 To begin this tutorial, you must create the necessary database accounts and endure that
 the SCOTT user account is active.
- Step 2: Create the Database Session-Based Application Context

 As the sysadmin_ctx user, you are ready to create the database session-based application context.
- Step 3: Create a Package to Retrieve Session Data and Set the Application Context
 Next, you must create a PL/SQL package that retrieves the session data and then sets the
 application context.
- Step 4: Create a Logon Trigger for the Package
 The logon trigger will execute when the user logs in.
- Step 5: Test the Application Context

 Now that the components are all in place, you are ready to test the application context.
- Step 6: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

13.3.9.1 Step 1: Create User Accounts and Ensure the User SCOTT Is Active

To begin this tutorial, you must create the necessary database accounts and endure that the ${\tt SCOTT}$ user account is active.

1. Log on as user SYS and connect using the SYSDBA administrative privilege.

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

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

For example:

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

To find the available PDBs, run the show pdbs command. To check the current PDB, run the show con name command.

Create the local user account sysadmin_ctx, who will administer the database sessionbased application context.

```
CREATE USER sysadmin_ctx IDENTIFIED BY password;
GRANT CREATE SESSION, CREATE ANY CONTEXT, CREATE PROCEDURE, CREATE TRIGGER,
ADMINISTER DATABASE TRIGGER TO sysadmin ctx;
```

```
GRANT READ ON HR.EMPLOYEES TO sysadmin_ctx;
GRANT EXECUTE ON DBMS SESSION TO sysadmin ctx;
```

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

4. Create the following user account for Lisa Ozer, who is listed as having lozer for her email account in the HR.EMPLOYEES table.

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

Replace password with a password that is secure.

5. The sample user SCOTT will also be used in this tutorial, so query the DBA_USERS data dictionary view to ensure that the account status for SCOTT is OPEN.

```
SELECT USERNAME, ACCOUNT STATUS FROM DBA USERS WHERE USERNAME = 'SCOTT';
```

If the DBA_USERS view lists user SCOTT as locked and expired, then enter the following statement to unlock the SCOTT account and create a new password for him:

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

Enter a password that is secure. For greater security, do **not** give the SCOTT account the same password from previous releases of Oracle Database. See Minimum Requirements for Passwords for the minimum requirements for creating passwords.

13.3.9.2 Step 2: Create the Database Session-Based Application Context

As the <code>sysadmin_ctx</code> user, you are ready to create the database session-based application context.

Log on to SQL*Plus as sysadmin_ctx.

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

Create the application context using the following statement:

```
CREATE CONTEXT empno_ctx USING set_empno_ctx_pkg;
```

Remember that even though user sysadmin_ctx has created this application context, the SYS schema owns the context.

13.3.9.3 Step 3: Create a Package to Retrieve Session Data and Set the Application Context

Next, you must create a PL/SQL package that retrieves the session data and then sets the application context.

To create the package, use the CREATE OR REPLACE PACKAGE statement.

Example 13-5 shows how to create the package you need to retrieve the session data and set the application context. Before creating the package, ensure that you are still logged on as user sysadmin_ctx. (You can copy and paste this text by positioning the cursor at the start of CREATE OR REPLACE in the first line.)



Example 13-5 Package to Retrieve Session Data and Set a Database Session Context

```
CREATE OR REPLACE PACKAGE set_empno_ctx_pkg IS
    PROCEDURE set_empno;
END;
/
CREATE OR REPLACE PACKAGE BODY set_empno_ctx_pkg IS
    PROCEDURE set_empno
    IS
    emp_id HR.EMPLOYEES.EMPLOYEE_ID*TYPE;
BEGIN
    SELECT EMPLOYEE_ID INTO emp_id FROM HR.EMPLOYEES
        WHERE email = SYS_CONTEXT('USERENV', 'SESSION_USER');
    DBMS_SESSION.SET_CONTEXT('empno_ctx', 'employee_id', emp_id);
    EXCEPTION
    WHEN NO_DATA_FOUND THEN NULL;
END;
END;
```

This package creates a procedure called set empno that performs the following actions:

- emp_id HR.EMPLOYEES.EMPLOYEE_ID%TYPE declares a variable, emp_id, to store the employee ID for the user who logs on. It uses the same data type as the EMPLOYEE_ID column in HR.EMPLOYEES.
- SELECT EMPLOYEE_ID INTO emp_id FROM HR.EMPLOYEES performs a SELECT statement to copy the employee ID that is stored in the employee_id column data from the HR.EMPLOYEES table into the emp id variable.
- WHERE email = SYS_CONTEXT('USERENV', 'SESSION_USER') uses a WHERE clause to find all employee IDs that match the email account for the session user. The SYS_CONTEXT function uses the predefined USERENV context to retrieve the user session ID, which is the same as the email column data. For example, the user ID and email address for Lisa Ozer are both the same: lozer.
- DBMS_SESSION.SET_CONTEXT('empno_ctx', 'employee_id', emp_id) uses the DBMS_SESSION.SET_CONTEXT procedure to set the application context:
 - 'empno_ctx': Calls the application context empno_ctx. Enclose empno_ctx in single quotes.
 - 'employee_id': Creates the attribute value of the empno_ctx application context name-value pair, by naming it employee id. Enclose employee id in single quotes.
 - emp_id: Sets the value for the employee_id attribute to the value stored in the emp_id variable.

To summarize, the $set_empno_ctx_pkg.set_empno$ procedure says, "Get the session ID of the user and then match it with the employee ID and email address of any user listed in the HR.EMPLOYEES table."

• EXCEPTION ... WHEN_NO_DATA_FOUND adds a WHEN NO_DATA_FOUND system exception to catch any no data found errors that may result from the SELECT statement. Without this exception, the package and logon trigger will work fine and set the application context as needed, but then any non-system administrator users other than the users listed in the HR.EMPLOYEES table will not be able to log in to the database. Other users should be able to log in to the database, assuming they are valid database users. Once the application

context information is set, then you can use this session information as a way to control user access to a particular application.

13.3.9.4 Step 4: Create a Logon Trigger for the Package

The logon trigger will execute when the user logs in.

 As user sysadmin_ctx, create a logon trigger for set_empno_ctx_pkg.set_empno package procedure.

```
CREATE TRIGGER set_empno_ctx_trig AFTER LOGON ON DATABASE
BEGIN
   sysadmin_ctx.set_empno_ctx_pkg.set_empno;
END;
//
```

13.3.9.5 Step 5: Test the Application Context

Now that the components are all in place, you are ready to test the application context.

1. Log on as user lozer.

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

When user lozer logs on, the empno_ctx application context collects her employee ID. You can check it as follows:

```
SELECT SYS CONTEXT('empno ctx', 'employee id') emp id FROM DUAL;
```

The following output should appear:

2. Log on as user SCOTT.

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

User SCOTT is not listed as an employee in the HR.EMPLOYEES table, so the $empno_ctx$ application context cannot collect an employee ID for him.

```
SELECT SYS_CONTEXT('empno_ctx', 'employee_id') emp_id FROM DUAL;
```

The following output should appear:

```
EMP_ID
```

From here, the application can use the user session information to determine how much access the user can have in the database. You can use Oracle Virtual Private Database to accomplish this. .

Related Topics

Using Oracle Virtual Private Database to Control Data Access
 Oracle Virtual Private Database (VPD) enables you to filter users who access data.

13.3.9.6 Step 6: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

Connect as SYS with the SYSDBA administrative privilege.

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

2. Drop the users sysadmin ctx and lozer:

```
DROP USER sysadmin_ctx CASCADE;
DROP USER lozer;
```

3. Drop the application context.

```
DROP CONTEXT empno ctx;
```

Remember that even though sysadmin_ctx created the application context, it is owned by the SYS schema.

4. If you want, lock and expire SCOTT, unless other users want to use this account:

ALTER USER SCOTT PASSWORD EXPIRE ACCOUNT LOCK;

13.3.10 Initializing Database Session-Based Application Contexts Externally

Initializing database session-based application contexts externally increases performance because the application context is stored in the user global area (UGA).

- About Initializing Database Session-Based Application Contexts Externally
 You must use a special type of namespace to initialize session-based application context
 externally.
- Default Values from Users

Oracle Database enables you to capture and use default values from users for your applications.

Values from Other External Resources

An application context can accept the initialization of attributes and values through external resources.

- Example: Creating an Externalized Database Session-based Application Context
 The CREATE CONTEXT SQL statement can create an externalized database session-based application context.
- Initialization of Application Context Values from a Middle-Tier Server
 Middle-tier servers can initialize application context values on behalf of database users.

13.3.10.1 About Initializing Database Session-Based Application Contexts Externally

You must use a special type of namespace to initialize session-based application context externally.

This namespace must accept the initialization of attribute values from external resources and then stores them in the local user session.

Initializing an application context externally enhances performance because it is stored in the UGA and enables the automatic propagation of attributes from one session to another. Connected user database links are supported only by application contexts initialized from OCI-based external sources.

13.3.10.2 Default Values from Users

Oracle Database enables you to capture and use default values from users for your applications.

Sometimes you need the default values from users. Initially, these default values may be hints or preferences, and then after validation, they become trusted contexts. Similarly, it may be more convenient for clients to initialize some default values, and then rely on a login event trigger or applications to validate the values.

For job queues, the job submission routine records the context being set at the time the job is submitted, and restores it when executing the batched job. To maintain the integrity of the context, job queues cannot bypass the designated PL/SQL package to set the context. Rather, the externally initialized application context accepts initialization of context values from the job queue process.

Automatic propagation of context to a remote session may create security problems. Developers or administrators can effectively handle the context that takes default values from resources other than the designated PL/SQL procedure by using logon triggers to reset the context when users log in.

13.3.10.3 Values from Other External Resources

An application context can accept the initialization of attributes and values through external resources.

Examples include an Oracle Call Interface (OCI) interface, a job queue process, or a database link.

Externally initialized application contexts provide the following features:

- For remote sessions, automatic propagation of context values that are in the externally initialized application context namespace
- For job queues, restoration of context values that are in the externally initialized application context namespace
- For OCI interfaces, a mechanism to initialize context values that are in the externally initialized application context namespace

Although any client program that is using Oracle Call Interface can initialize this type of namespace, you can use login event triggers to verify the values. It is up to the application to interpret and trust the values of the attributes.

13.3.10.4 Example: Creating an Externalized Database Session-based Application Context

The CREATE CONTEXT SQL statement can create an externalized database session-based application context.

Example 13-6 shows how to create a database session-based application context that obtains values from an external source.

Example 13-6 Creating an Externalized Database Session-based Application Context

CREATE CONTEXT ext ctx USING ext ctx pkg INITIALIZED EXTERNALLY;



13.3.10.5 Initialization of Application Context Values from a Middle-Tier Server

Middle-tier servers can initialize application context values on behalf of database users.

In this process, context attributes are propagated for the remote session at initialization time, and the remote database accepts the values if the namespace is externally initialized.

For example, a three-tier application creating lightweight user sessions through OCI or JDBC/OCI can access the PROXY_USER attribute in USERENV. This attribute enables you to determine if the user session was created by a middle-tier application. You could allow a user to access data only for connections where the user is proxied. If users connect directly to the database, then they would not be able to access any data.

You can use the PROXY_USER attribute from the USERENV namespace within Oracle Virtual Private Database to ensure that users only access data through a particular middle-tier application. For a different approach, you can develop a secure application role to enforce your policy that users access the database only through a specific proxy.

See Also:

- Preserving User Identity in Multitiered Environments for information about proxy authentication and about using the USERENV attribute CLIENT_IDENTIFIER to preserve user identity across multiple tiers
- Middle Tier Server Use for Proxy Authentication for information about using a secure application role to enforce a policy through a specific proxy
- Oracle Call Interface Programmer's Guide

13.3.11 Initializing Database Session-Based Application Contexts Globally

When a database session-based application is stored in a centralized location, it can be used globally from an LDAP directory.

- About Initializing Database Session-Based Application Contexts Globally
 You can use a centralized location to store the database session-based application context
 of the user.
- Database Session-Based Application Contexts with LDAP
 An application context that is initialized globally uses LDAP, a standard, extensible, and efficient directory access protocol.
- How Globally Initialized Database Session-Based Application Contexts Work
 To use a globally initialized secure application, you must first configure Enterprise User
 Security.
- Initializing a Database Session-Based Application Context Globally
 You can configure and store the initial application context for a user, such as the department name and title, in the LDAP directory.

13.3.11.1 About Initializing Database Session-Based Application Contexts Globally

You can use a centralized location to store the database session-based application context of the user.

A centralized location enables applications to set up a user context during initialization based upon user identity.

In particular, this feature supports Oracle Label Security labels and privileges. Initializing an application context globally makes it easier to manage contexts for large numbers of users and databases.

For example, many organizations want to manage user information centrally, in an LDAP-based directory. Enterprise User Security supports centralized user and authorization management in Oracle Internet Directory. However, there may be additional attributes an application must retrieve from Lightweight Directory Access Protocol (LDAP) to use for Oracle Virtual Private Database enforcement, such as the user title, organization, or physical location. Initializing an application context globally enables you to retrieve these types of attributes.

13.3.11.2 Database Session-Based Application Contexts with LDAP

An application context that is initialized globally uses LDAP, a standard, extensible, and efficient directory access protocol.

The LDAP directory stores a list of users to which this application is assigned. Oracle Database uses a directory service, typically Oracle Internet Directory, to authenticate and authorize enterprise users.



You can use third-party directories such as Microsoft Active Directory and Sun Microsystems SunONE as the directory service.

The orcldBApplicationContext LDAP object (a subclass of groupOfUniqueNames) stores the application context values in the directory. The location of the application context object is described in Figure 13-1, which is based on the Human Resources example.

The LDAP object inetorgPerson enables multiple entries to exist for some attributes. However, be aware that when these entries are loaded into the database and accessed with the SYS_LDAP_USER_DEFAULT context namespace, then only the first of these entries is returned. For example, the inetorgPerson object for a user allows multiple entries for telephoneNumber (thus allowing a user to have multiple telephone numbers stored). When you use the SYS_LDAP_USER_DEFAULT context namespace, only the first telephone number is retrieved. If the list of attributes and values that are provided are not sufficient for your needs, then you can use the DBMS_LDAP_PL/SQL package to fetch additional values from the directory.

On the LDAP side, an internal C function is required to retrieve the <code>orclDBApplicationContext</code> value, which returns a list of application context values to the database. In this example, <code>HR</code> is the namespace; <code>Title</code> and <code>Project</code> are the attributes; and <code>Manager</code> and <code>Promotion</code> are the values.



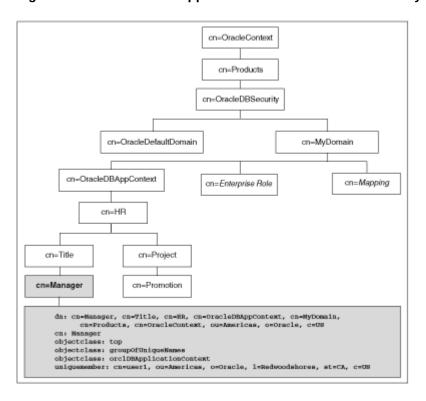


Figure 13-1 Location of Application Context in LDAP Directory Information Tree

13.3.11.3 How Globally Initialized Database Session-Based Application Contexts Work

To use a globally initialized secure application, you must first configure Enterprise User Security.

Then, you configure the application context values for the user in the database and the directory.

When a global user (enterprise user) connects to the database, Enterprise User Security verifies the identity of the user connecting to the database. After authentication, the global user roles and application context are retrieved from the directory. When the user logs on to the database, the global roles and initial application context are already set.



Oracle Database Enterprise User Security Administrator's Guide for information about configuring Enterprise User Security

13.3.11.4 Initializing a Database Session-Based Application Context Globally

You can configure and store the initial application context for a user, such as the department name and title, in the LDAP directory.

The values are retrieved during user login so that the context is set properly. In addition, any information related to the user is retrieved and stored in the <code>SYS_USER_DEFAULTS</code> application context namespace.

1. Create the application context in the database.

```
CREATE CONTEXT hr USING hrapps.hr manage pkg INITIALIZED GLOBALLY;
```

2. Create and add new entries in the LDAP directory.

An example of the entries added to the LDAP directory follows. These entries create an attribute named <code>Title</code> with the attribute value <code>Manager</code> for the application (namespace) <code>HR</code>, and assign user names <code>user1</code> and <code>user2</code>. In the following, <code>cn=example</code> refers to the name of the domain.

```
dn:
cn=OracleDBAppContext,cn=example,cn=OracleDBSecurity,cn=Products,cn=OracleContext,ou=
Americas, o=oracle, c=US
changetype: add
cn: OracleDBAppContext
objectclass: top
objectclass: orclContainer
cn=hr,cn=OracleDBAppContext,cn=example,cn=OracleDBSecurity,cn=Products,cn=OracleConte
xt, ou=Americas, o=oracle, c=US
changetype: add
cn: hr
objectclass: top
objectclass: orclContainer
dn: cn=Title, cn=hr,
cn=OracleDBAppContext,cn=example,cn=OracleDBSecurity,cn=Products,cn=OracleContext,ou=
Americas, o=oracle, c=US
changetype: add
cn: Title
objectclass: top
objectclass: orclContainer
dn: cn=Manager, cn=Title, cn=hr,
cn=OracleDBAppContext,cn=example,cn=OracleDBSecurity,cn=Products,cn=OracleContext,ou=
Americas, o=oracle, c=US
cn: Manager
objectclass: top
objectclass: groupofuniquenames
objectclass: orclDBApplicationContext
uniquemember: CN=user1,OU=Americas,O=Oracle,L=Redwoodshores,ST=CA,C=US
uniquemember: CN=user2,OU=Americas,O=Oracle,L=Redwoodshores,ST=CA,C=US
```

3. If an LDAP <code>inetOrgPerson</code> object entry exists for the user, then the connection retrieves the attributes from <code>inetOrgPerson</code>, and assigns them to the namespace <code>SYS_LDAP_USER_DEFAULT</code>. Note that the context is only populated with non-<code>NULL</code> values that are part of the <code>inetOrgPerson</code> object class. No other attributes will be populated.

The following is an example of an inetOrgPerson entry:

```
dn: cn=user1, ou=Americas, O=oracle, L=redwoodshores, ST=CA, C=US
changetype: add
objectClass: top
objectClass: person
objectClass: organizationalPerson
objectClass: inetOrgPerson
cn: user1
sn: One
givenName: User
initials: UO
title: manager, product development
uid: uone
mail: uone@us.example.com
telephoneNumber: +1 650 555 0105
employeeNumber: 00001
employeeType: full time
```

Connect to the database.

When user1 connects to a database that belongs to the example domain, user1 will have his Title set to Manager. Any information related to user1 will be retrieved from the LDAP directory. The value can be obtained using the following syntax:

```
SYS_CONTEXT('namespace','attribute name')
```

For example:

```
DECLARE
  tmpstr1 VARCHAR2(30);
  tmpstr2 VARCHAR2(30);
BEGIN
  tmpstr1 = SYS_CONTEXT('HR','TITLE);
  tmpstr2 = SYS_CONTEXT('SYS_LDAP_USER_DEFAULT','telephoneNumber');
  DBMS_OUTPUT.PUT_LINE('Title is ' || tmpstr1);
  DBMS_OUTPUT.PUT_LINE('Telephone Number is ' || tmpstr2);
END:
```

The output of this example is:

```
Title is Manager
Telephone Number is +1 650 555 0105
```

13.3.12 Externalized Database Session-Based Application Contexts

Many applications store attributes used for fine-grained access control within a database metadata table.

For example, an employees table could include cost center, title, signing authority, and other information useful for fine-grained access control. Organizations also centralize user information for user management and access control in LDAP-based directories, such as Oracle Internet Directory. Application context attributes can be stored in Oracle Internet Directory, and assigned to one or more enterprise users. They can also be retrieved automatically upon login for an enterprise user, and then used to initialize an application context.

See Also:

- Initializing Database Session-Based Application Contexts Externally for information about initializing local application context through external resources such as an OCI interface, a job queue process, or a database link
- Initializing Database Session-Based Application Contexts Globally for information about initializing local application context through a centralized resource, such as Oracle Internet Directory
- Oracle Database Enterprise User Security Administrator's Guide for information about enterprise users

13.4 Global Application Contexts

You can use a global application context to access application values across database sessions, including an Oracle Real Application Clusters environment.

- About Global Application Contexts
 A global application context enables application context values to be accessible across database sessions, including Oracle RAC instances.
- Uses for Global Application Contexts
 There are three general uses for global application contexts.
- Components of a Global Application Context
 A global application context uses a package to manage its attributes and middle-tier application to manage the client session ID.
- Global Application Contexts in an Oracle Real Application Clusters Environment
 In an Oracle RAC environment, whenever a global application context is loaded or
 changed, it is visible only to the existing active instances.
- Creating Global Application Contexts
 The CREATE CONTEXT SQL statement creates the global application context, which is then located in the SYS schema.
- PL/SQL Package to Manage a Global Application Context
 The DBMS SESSION PL/SQL package to manages global application contexts.
- Embedding Calls in Middle-Tier Applications to Manage the Client Session ID
 You can embed calls in middle-tier applications to manage client session IDs.
- Tutorial: Creating a Global Application Context That Uses a Client Session ID
 This tutorial demonstrates how you can create a global application context that uses a client session ID.
- Global Application Context Processes
 A simple global application context uses a database user account create the user session;
 a global application context is for lightweight users.

13.4.1 About Global Application Contexts

A global application context enables application context values to be accessible across database sessions, including Oracle RAC instances.

Oracle Database stores the global application context information in the System (sometimes called "Shared") Global Area (SGA) so that it can be used for applications that use a sessionless model, such as middle-tier applications in a three-tiered architecture.

These applications cannot use a session-based application context because users authenticate to the application, and then it typically connects to the database as a single identity. Oracle Database initializes the global application context once, rather than for each user session. This improves performance, because connections are reused from a connection pool.

You can clear a global application context value by running the ALTER SYSTEM FLUSH GLOBAL CONTEXT SQL statement.

13.4.2 Uses for Global Application Contexts

There are three general uses for global application contexts.

These uses are as follows:

- You must share application values globally for all database users. For example, you may need to disable access to an application based on a specific situation. In this case, the values the application context sets are not user-specific, nor are they based on the private data of a user. The application context defines a situation, for example, to indicate the version of application module that is running.
- You have database users who must move from one application to another. In this case, the second application the user is moving to has different access requirements from the first application.
- You must authenticate nondatabase users, that is, users who are not known to the database. This type of user, who does not have a database account, typically connects through a Web application by using a connection pool. These types of applications connect users to the database as single user, using the One Big Application User authentication model. To authenticate this type of user, you use the client session ID of the user.

13.4.3 Components of a Global Application Context

A global application context uses a package to manage its attributes and middle-tier application to manage the client session ID.

- The global application context. You use the CREATE CONTEXT SQL statement to create the global application context, and include the ACCESSED GLOBALLY clause in the statement. This statement names the application context and associates it with a PL/SQL procedure that is designed to set the application data context data. The global application context is created and stored in the database schema of the security administrator who creates it.
- A PL/SQL package to set the attributes. The package must contain a procedure that uses the <code>DBMS_SESSION.SET_CONTEXT</code> procedure to set the global application context. The <code>SET_CONTEXT</code> procedure provides parameters that enable you to create a global application context that fits any of the three user situations described in this section. You create, store, and run the PL/SQL package on the database server. Typically, it belongs in the schema of the security administrator who created it.
- A middle-tier application to get and set the client session ID. For nondatabase users, which require a client session ID to be authenticated, you can use the Oracle Call Interface (OCI) calls in the middle-tier application to retrieve and set their session data. You can also use the DBMS_SESSION.SET_IDENTIFIER procedure to set the client session ID. An advantage of creating a client session ID to store the nondatabase user's name is that you



can query the CLIENT_ID column of DBA_AUDIT_TRAIL, DBA_FGA_AUDIT_TRAIL, and DBA_COMMON_AUDIT_TRAIL data dictionary views to audit this user's activity.



Be aware that the <code>DBMS_APPLICATION_INFO.SET_CLIENT_INFO</code> setting can overwrite the value.

Related Topics

Use of the DBMS_SESSION PL/SQL Package to Set and Clear the Client Identifier
 The DBMS_SESSION PL/SQL package manages client identifiers on both the middle tier and
 the database itself.

13.4.4 Global Application Contexts in an Oracle Real Application Clusters Environment

In an Oracle RAC environment, whenever a global application context is loaded or changed, it is visible only to the existing active instances.

Be aware that setting a global application context value in an Oracle RAC environment has performance overhead of propagating the context value consistently to all Oracle RAC instances.

If you flush the global application context (using the ALTER SYSTEM FLUSH GLOBAL_CONTEXT SQL statement) in one Oracle RAC instance, then all the global application context is flushed in all other Oracle RAC instances as well.

13.4.5 Creating Global Application Contexts

The CREATE CONTEXT SQL statement creates the global application context, which is then located in the SYS schema.

- Ownership of the Global Application Context
 A global application context is owned by the SYS schema.
- Creating a Global Application Context
 As with local application contexts, the global application context is created and stored in the security administrator's database schema.

13.4.5.1 Ownership of the Global Application Context

A global application context is owned by the SYS schema.

The ownership of the global application context is as follows: Even though a user who has been granted the CREATE ANY CONTEXT and DROP ANY CONTEXT privileges can create and drop the global application context, it is owned by the SYS schema.

Oracle Database associates the context with the schema account that created it, but if you drop this user, the context still exists in the SYS schema. As user SYS, you can drop the application context.



13.4.5.2 Creating a Global Application Context

As with local application contexts, the global application context is created and stored in the security administrator's database schema.

You must have the CREATE ANY CONTEXT system privilege before you can create a global application context, and the DROP ANY CONTEXT privilege before you can drop the context with the DROP CONTEXT statement.

• To create a global application context, use the CREATE CONTEXT SQL statement to create the application context and include the ACCESSED GLOBALLY clause in the statement.

For example:

CREATE OR REPLACE CONTEXT global_hr_ctx USING hr_ctx_pkg ACCESSED GLOBALLY CONTAINER = ALL:

13.4.6 PL/SQL Package to Manage a Global Application Context

The DBMS SESSION PL/SQL package to manages global application contexts.

- About the Package That Manages the Global Application Context
 The package that is associated with a global application context uses the DBMS_SESSION package to set and clear the global application context values.
- How Editions Affects the Results of a Global Application Context PL/SQL Package
 Global application context packages, Oracle Virtual Private Database packages, and finegrained audit policies can be used across multiple editions.
- DBMS_SESSION.SET_CONTEXT username and client_id Parameters
 The DBMS_SESSION.SYS_CONTEXT procedure provides the client_id and username parameters, to be used for global application contexts.
- Sharing Global Application Context Values for All Database Users
 You can share global application values for all database users to give them access to data in the database.
- Example: Package to Manage Global Application Values for All Database Users
 The CREATE PACKAGE statement can manage global application values for all database users.
- Global Contexts for Database Users Who Move Between Applications
 A global application context can be used for database users who move between application, even when the applications have different access requirements.
- Global Application Context for Nondatabase Users
 When a nondatabase user starts a client session, the application server generates a client session ID.
- Example: Package to Manage Global Application Context Values for Nondatabase Users
 The CREATE PACKAGE statement can manage global application context values for nondatabase users.
- Clearing Session Data When the Session Closes
 The application context exists within memory, so when the user exits a session, either by switching to another session or ending the current session, you must clear the client identifier context value.



13.4.6.1 About the Package That Manages the Global Application Context

The package that is associated with a global application context uses the <code>DBMS_SESSION</code> package to set and clear the global application context values.

You must have the EXECUTE privilege for the DBMS_SESSION package before you use its procedures. Typically, you create and store this package in the database schema of a security administrator. The SYS schema owns the DBMS SESSION package.

Unlike PL/SQL packages used to set a local application context, you do not include a SYS_CONTEXT function to get the user session data. You do not need to include this function because the owner of the session, recorded in the USERENV context, is the same for every user who is connecting.

You can run the procedures within the PL/SQL package for a global application context at any time. You do not need to create logon and logoff triggers to execute the package procedures associated with the global application context. A common practice is to run the package procedures from within the database application. Additionally, for nondatabase users, you use middle-tier applications to get and set client session IDs.



Oracle Database PL/SQL Packages and Types Reference for detailed information about the DBMS SESSION package

13.4.6.2 How Editions Affects the Results of a Global Application Context PL/SQL Package

Global application context packages, Oracle Virtual Private Database packages, and finegrained audit policies can be used across multiple editions.

Follow these guidelines:

- If you want to have the PL/SQL package results be the same across all editions. To do so, create the package in the schema of a user who has not been editions enabled. To find users who are not editions enabled, you can query the DBA_USERS and USER_USERS data dictionary views. Remember that SYS, SYSTEM, and other default Oracle Database administrative accounts that are listed in the DBA_REGISTRY data dictionary view are not and cannot be editions enabled.
- If you want to have the PL/SQL package results depend on the current state of the edition in which the package is run. Here, the results may be different across all editions to which the package applies. In this case, create the package in the schema of a user who has been editions enabled. If the schema is editions enabled, then it is likely that there will be different actual copies of the package in different editions, where each copy has different behavior. This is useful for the following types of scenarios:
 - The package must use a new application context.
 - The package must encode input values using a different scheme.
 - The package must apply different validation rules for users logging in to the database.



For PL/SQL packages that set a global application context, use a single getter function to wrap the primitive SYS_CONTEXT calls that will read the key-value application context pairs. You can put this getter function in the same package as the application context setter procedure. This approach lets you tag the value for the application context key to reflect a relevant concept. For example, the tag can be the edition in which the setter function is actual. Or, it can be the current edition of the session that set the context, which you can find by using SYS_CONTEXT('USERENV', 'CURRENT_EDITION_NAME'). This tag can be any specific notion to which the setter function applies.

See Also:

Oracle Database Development Guide for detailed information about editions

13.4.6.3 DBMS_SESSION.SET_CONTEXT username and client_id Parameters

The DBMS_SESSION.SYS_CONTEXT procedure provides the client_id and username parameters, to be used for global application contexts.

Table 13-2 explains how the combination of these settings controls the type of global application context you can create.

Table 13-2 Setting the DBMS_SESSION.SET_CONTEXT username and client_id Parameters

Combination Settings	Result		
username set to NULL client id set to NULL	This combination enables all users to access the application context. See Sharing Global Application Context Values for All Database Users for more information.		
client_id set to Nobb	These settings are also used for database session-based application contexts. See Using Database Session-Based Application Contexts for more information.		
username set to a value	This combination enables an application context to be accessed by multiple session		
client_id set to NULL	as long as the username setting is the same throughout. Ensure that the user name specified is a valid database user. See Global Contexts for Database Users Who Move Between Applications for more information.		
username set to NULL	This combination enables an application to be accessed by multiple user sessions,		
client_id set to a value	as long as the client_id parameter is set to the same value throughout. This enables sessions of all users to see the application context values.		
username set to a value	This combination enables the following two scenarios:		
client_id set to a value	 Lightweight users. If the user does not have a database account, the username specified is a connection pool owner. The client_id setting is then associated with the nondatabase user who is logging in. 		
	 Database users. If the user is a database user, this combination can be used for stateless Web sessions. 		
	Setting the username parameter in the SET_CONTEXT procedure to USER calls the Oracle Database-supplied USER function. The USER function specifies the session owner from the application context retrieval process and ensures that only the user who set the application context can access the context. See <i>Oracle Database SQL Language Reference</i> for more information about the USER function.		
	See Global Application Context for Nondatabase Users for more information.		

13.4.6.4 Sharing Global Application Context Values for All Database Users

You can share global application values for all database users to give them access to data in the database.

• To share global application values for all database users, set the namespace, attribute, and value parameters in the SET CONTEXT procedure.

Related Topics

Example: Package to Manage Global Application Values for All Database Users
 The CREATE PACKAGE statement can manage global application values for all database users.

13.4.6.5 Example: Package to Manage Global Application Values for All Database Users

The CREATE PACKAGE statement can manage global application values for all database users.

Example 13-7 shows how to create a package that sets and clears a global application context for all database users.

Example 13-7 Package to Manage Global Application Values for All Database Users

```
CREATE OR REPLACE PACKAGE hr_ctx_pkg

AS

PROCEDURE set_hr_ctx(sec_level IN VARCHAR2);
PROCEDURE clear_hr_context;
END;

CREATE OR REPLACE PACKAGE BODY hr_ctx_pkg

AS

PROCEDURE set_hr_ctx(sec_level IN VARCHAR2)

AS

BEGIN

DBMS_SESSION.SET_CONTEXT(

namespace => 'global_hr_ctx',
attribute => 'job_role',
value => sec_level);
END set_hr_ctx;

PROCEDURE clear_hr_context

AS

BEGIN

DBMS_SESSION.CLEAR_CONTEXT('global_hr_ctx', 'job_role');
END clear_context;
END;

/
```

In this example:

DBMS_SESSION.SET_CONTEXT ... END set_hr_ctx uses the DBMS_SESSION.SET_CONTEXT procedure to set values for the namespace, attribute, and value parameters. The sec_level value is specified when the database application runs the hr_ctx_pkg.set_hr_ctx procedure.

The username and client_id values are not set, hence, they are NULL. This enables all users (database users) to have access to the values, which is appropriate for server-wide settings.

- namespace => 'global_hr_ctx' sets the namespace to global_hr_ctx, in the SET_CONTEXT procedure.
- attribute => 'job_role' creates the job_role attribute.
- value => sec_level sets the value for the job_role attribute to sec_level.
- PROCEDURE clear_hr_context creates the clear_hr_context procedure to clear the context values. See Clearing Session Data When the Session Closes for more information.

Typically, you execute this procedure within a database application. For example, if all users logging in are clerks, and you want to use "clerk" as a security level, you would embed a call within a database application similar to the following:

```
BEGIN
  hr_ctx_pkg.set_hr_ctx('clerk');
END;
/
```

If the procedure successfully completes, then you can check the application context values as follows:

You can clear the global application context values for all database users by running the following procedure:

```
BEGIN
hr_ctx_pkg.clear_hr_context;
END;
/
```

To check that the global context value is really cleared, the following SELECT statement should return no values:

```
SELECT SYS_CONTEXT('global_hr_ctx', 'job_role') job_role FROM DUAL;

JOB_ROLE
------
```

If Oracle Database returns error messages saying that you have insufficient privileges, then ensure that you have correctly created the global application context. You should also query the DBA_CONTEXT database view to ensure that your settings are correct, for example, that you are calling the procedure from the schema in which you created it.

If \mathtt{NULL} is returned, then you may have inadvertently set a client identifier. To clear the client identifier, run the following procedure:

```
EXEC DBMS_SESSION.CLEAR_IDENTIFIER;
```

13.4.6.6 Global Contexts for Database Users Who Move Between Applications

A global application context can be used for database users who move between application, even when the applications have different access requirements.

To do so, you must include the username parameter in the DBMS_SESSION.SET_CONTEXT procedure.

This parameter specifies that the same schema be used for all sessions.

You can use the following DBMS SESSION.SET CONTEXT parameters:

- namespace
- attribute
- งเป็น
- username

Oracle Database matches the username value so that the other application can recognize the application context. This enables the user to move between applications.

By omitting the client_id setting, its value is NULL, the default. This means that values can be seen by multiple sessions if the username setting is the same for a database user who maintains the same context in different applications. For example, you can have a suite of applications that control user access with Oracle Virtual Private Database policies, with each user restricted to a job role.

Example 13-8 demonstrates how to set the username parameter so that a specific user can move between applications. The use of the username parameter is indicated in **bold** typeface.

Example 13-8 Package for Global Application Context Values for Moving Between Applications

```
CREATE OR REPLACE PACKAGE hr ctx pkg
   PROCEDURE set hr ctx(sec level IN VARCHAR2, user name IN VARCHAR2);
    PROCEDURE clear hr context;
  CREATE OR REPLACE PACKAGE BODY hr ctx pkg
    PROCEDURE set hr ctx(sec level IN VARCHAR2, user_name IN VARCHAR2)
   AS
     BEGIN
     DBMS SESSION.SET CONTEXT (
      namespace => 'global hr ctx',
      attribute => 'job role',
      value => sec_level,
username => user_name);
      END set hr ctx;
   PROCEDURE clear_hr_context
    AS
     BEGIN
     DBMS SESSION.CLEAR CONTEXT('global hr ctx');
     END clear context;
 END;
```

Typically, you execute this procedure within a database application by embedding a call similar to the following example. Ensure that the value for the user_name parameter (scott in this case) is a valid database user name.

```
BEGIN
hr_ctx_pkg.set_hr_ctx('clerk', 'scott');
END;
```

A secure way to manage this type of global application context is within your applications, embed code to grant a secure application role to the user. This code should include EXECUTE

permissions on the trusted PL/SQL package that sets the application context. In other words, the application, not the user, will set the context for the user.

13.4.6.7 Global Application Context for Nondatabase Users

When a nondatabase user starts a client session, the application server generates a client session ID.

A nondatabase user is a user who is not known to the database, such as a Web application user.

Once this ID is set on the application server, it must be passed to the database server side. You can do this by using the <code>DBMS_SESSION.SET_IDENTIFIER</code> procedure to set the client session ID.

To set the context, you can set the client_id parameter in the DBMS_SESSION.SET_CONTEXT procedure, in a PL/SQL procedure on the server side. This enables you to manage the application context globally, yet each client sees only his or her assigned application context.

The client_id value is the key here to getting and setting the correct attributes for the global application context. Remember that the client identifier is controlled by the middle-tier application, and once set, it remains open until it is cleared.

A typical way to manage this type of application context is to place the <code>session_id</code> value (<code>client_identifier</code>) in a cookie, and send it to the end user's HTML page so that is returned on the next request. A lookup table in the application should also keep client identifiers so that they are prevented from being reused for other users and to implement an end-user session time out.

For nondatabase users, configure the following SET CONTEXT parameters:

- namespace
- attribute
- value
- username
- client id

Related Topics

- Tutorial: Creating a Global Application Context That Uses a Client Session ID
 This tutorial demonstrates how you can create a global application context that uses a client session ID.
- Step 2: Set the Client Session ID Using a Middle-Tier Application
 Next, you are ready to set the client session ID using a middle-tier application.
- Using Client Identifiers to Identify Application Users Unknown to the Database
 Client identifiers preserve user identity in middle tier systems; they also can be used
 independently of the global application context.

13.4.6.8 Example: Package to Manage Global Application Context Values for Nondatabase Users

The CREATE PACKAGE statement can manage global application context values for nondatabase users.



Example 13-9 shows how to create a package that manages this type of global application context.

Example 13-9 Package to Manage Global Application Context Values for Nondatabase Users

```
CREATE OR REPLACE PACKAGE hr ctx pkg
  PROCEDURE set session id(session id p IN NUMBER);
  PROCEDURE set hr ctx(sec level attr IN VARCHAR2,
     sec level val IN VARCHAR2);
  PROCEDURE clear hr session(session id p IN NUMBER);
  PROCEDURE clear hr context;
 CREATE OR REPLACE PACKAGE BODY hr ctx pkg
   session id global NUMBER;
 PROCEDURE set session id (session id p IN NUMBER)
  BEGIN
   session id global := session id p;
   DBMS SESSION.SET IDENTIFIER(session id p);
 END set session id;
  PROCEDURE set hr ctx(sec level attr IN VARCHAR2,
    sec level val IN VARCHAR2)
  BEGIN
   DBMS SESSION.SET CONTEXT(
    namespace => 'global_hr_ctx',
    attribute => sec level attr,
    value => sec level val,
    username => USER,
    client id => session id global);
  END set hr ctx;
 PROCEDURE clear hr session(session id p IN NUMBER)
  AS
  BEGIN
     DBMS SESSION.SET IDENTIFIER(session id p);
     DBMS SESSION.CLEAR IDENTIFIER;
  END clear hr session;
 PROCEDURE clear hr context
  DBMS SESSION.CLEAR CONTEXT('global hr ctx', session id global);
 END clear hr context;
END;
```

In this example:

- session_id_global NUMBER creates the session_id_global variable, which will hold the client session ID. The session_id_global variable is referenced throughout the package definition, including the procedure that creates the global application context attributes and assigns them values. This means that the global application context values will always be associated with this particular session ID.
- PROCEDURE set_session_id ... END set_session_id creates the set_session_id procedure, which writes the client session ID to the session id global variable.

- PROCEDURE set_hr_ctx ... END set_hr_ctx creates the set_hr_ctx procedure, which creates global application context attributes and enables you to assign values to these attributes. Within this procedure:
 - username => USER specifies the username value. This example sets it by calling the
 Oracle Database-supplied USER function, which adds the session owner from the
 context retrieval process. The USER function ensures that only the user who set the
 application context can access the context. See Oracle Database SQL Language
 Reference for more information about the USER function.

If you had specified ${\tt NULL}$ (the default for the username parameter), then any user can access the context.

Setting both the username and client_id values enables two scenarios. For lightweight users, set the username parameter to a connection pool owner (for example, APPS_USER), and then set client_id to the client session ID. If you want to use a stateless Web session, set the user_name parameter to the same database user who has logged in, and ensure that this user keeps the same client session ID. See DBMS_SESSION.SET_CONTEXT username and client_id Parameters for an explanation of how different username and client_id settings work.

- client_id => session_id_global specifies client_id value. This example sets it to the session_id_global variable. This associates the context settings defined here with a specific client session ID, that is, the one that is set when you run the set_session_id procedure. If you specify the client_id parameter default, NULL, then the global application context settings could be used by any session.
- PROCEDURE clear_hr_session ... END clear_hr_session creates the clear_hr_session procedure to clear the client session identifier. The AS clause sets it to ensure that you are clearing the correct session ID, that is, the one stored in variable session_id_p defined in the CREATE OR REPLACE PACKAGE BODY hr ctx pkg procedure.
- PROCEDURE clear_hr_context ... END clear_hr_context creates the clear_hr_context procedure, so that you can clear the context settings for the current user session, which were defined by the global_hr_ctx variable. See Clearing Session Data When the Session Closes for more information.

13.4.6.9 Clearing Session Data When the Session Closes

The application context exists within memory, so when the user exits a session, either by switching to another session or ending the current session, you must clear the client identifier context value.

This releases memory and prevents other users from accidentally using any left over values.

- To clear session data when a user exits a session (by switching or ending), use either of the following methods in the server-side PL/SQL package:
 - Clearing the client identifier when a user exits a session. Use the DBMS_SESSION.CLEAR_IDENTIFIER procedure. For example:

```
EXEC DBMS SESSION.CLEAR IDENTIFIER;
```

- Continuing the session but still clearing the context. If you want the session to continue, but you still need to clear the context, use one of the following procedures:
 - * DBMS_SESSION.CLEAR_CONTEXT clears the context for the current user. For example:

```
EXEC DBMS_SESSION.CLEAR_CONTEXT('my_ctx', 'my_client_id', 'my_attribute');
```



* DBMS_SESSION.CLEAR_ALL_CONTEXT clears the context values for all users, for example, when you need to shut down the application server. For example:

```
EXEC DBMS_SESSION.CLEAR_ALL_CONTEXT('my_ctx');
```

Global application context values are available until they are cleared, so you should use <code>DBMS_SESSION.CLEAR_CONTEXT</code> or <code>DBMS_SESSION.CLEAR_ALL_CONTEXT</code> to ensure that other sessions do not have access to these values. Be aware that any changes in the context value are reflected immediately and subsequent calls to access the value through the <code>SYS_CONTEXT</code> function will return the most recent value.

13.4.7 Embedding Calls in Middle-Tier Applications to Manage the Client Session ID

You can embed calls in middle-tier applications to manage client session IDs.

- About Managing Client Session IDs Using a Middle-Tier Application
 The application server generates the client session ID.
- Step 1: Retrieve the Client Session ID Using a Middle-Tier Application When a user starts a client session, the application server generates a client session ID.
- Step 2: Set the Client Session ID Using a Middle-Tier Application
 Next, you are ready to set the client session ID using a middle-tier application.
- Step 3: Clear the Session Data Using a Middle-Tier Application
 The application context exists entirely within memory.

13.4.7.1 About Managing Client Session IDs Using a Middle-Tier Application

The application server generates the client session ID.

From a middle-tier application, you can get, set, and clear the client session IDs. To do so, you can embed either Oracle Call Interface (OCI) calls or DBMS_SESSION PL/SQL package procedures into the middle-tier application code.

The application authenticates the user, sets the client identifier, and sets it in the current session. The PL/SQL package SET_CONTEXT sets the client_identifier value in the application context.

Related Topics

Global Application Context for Nondatabase Users
 When a nondatabase user starts a client session, the application server generates a client session ID.

13.4.7.2 Step 1: Retrieve the Client Session ID Using a Middle-Tier Application

When a user starts a client session, the application server generates a client session ID.

You can retrieve this ID for use in authenticating the user's access.

- To retrieve this client ID, use the OCIStmtExecute call with any of the following statements:
 - SELECT SYS_CONTEXT('userenv', 'client_identifier') FROM DUAL;
 - SELECT CLIENT IDENTIFIER from V\$SESSION;
 - SELECT value FROM session_context WHERE attribute='CLIENT_IDENTIFIER';



For example, to use the <code>OCIStmtExecute</code> call to retrieve a client session ID value:

In this example:

- oratext, OCIDefine, OCIStmt, and oratext create variables to store the client session ID, reference call for OCIDefine, the statement handle, and the SELECT statement to use.
- OCIStmtPrepar prepares the statement selcid for execution.
- OCIDefineByPos defines the output variable clientid for client session ID.
- OCIStmtExecute executes the statement in the selcid variable.
- printf prints the formatted output for the retrieved client session ID.

13.4.7.3 Step 2: Set the Client Session ID Using a Middle-Tier Application

Next, you are ready to set the client session ID using a middle-tier application.

- About Setting the Client Session ID Using a Middle-Tier Application
 After you use the OCIStmtExecute call to retrieve the client session ID, you are ready to set this ID.
- Setting the Client Session ID Using a Middle-Tier Application
 Oracle Call Interface or the DBMS_SESSION PL/SQL package can set the client session ID using a middle-tier application.
- Checking the Value of the Client Identifier
 For both OCIAttrSet and DBMS_SESSION.SET_IDENTIFIER, you can check the value of the client identifier.

13.4.7.3.1 About Setting the Client Session ID Using a Middle-Tier Application

After you use the <code>OCIStmtExecute</code> call to retrieve the client session ID, you are ready to set this ID.

The DBMS_SESSION.SET_CONTEXT procedure in the server-side PL/SQL package then sets this session ID and optionally, overwrites the application context values.

You must ensure that the middle-tier application code checks that the client session ID value (for example, the value written to user_id in the previous examples) matches the client_id

setting defined in the server-side <code>DBMS_SESSION.SET_CONTEXT</code> procedure. The sequence of calls on the application server side should be as follows:

- 1. Get the current client session ID. The session should already have this ID, but it is safer to ensure that it truly has the correct value.
- Clear the current client session ID. This prepares the application to service a request from a different end user.
- Set the new client session ID or the client session ID that has been assigned to the end user. This ensures that the session is using a different set of global application context values.

13.4.7.3.2 Setting the Client Session ID Using a Middle-Tier Application

Oracle Call Interface or the DBMS_SESSION PL/SQL package can set the client session ID using a middle-tier application.

- Use either of the following methods to set the client session ID on the application server side:
 - Oracle Call Interface. Set the OCI_ATTR_CLIENT_IDENTIFIER attribute in an OCIAttrSet OCI call. This attribute sets the client identifier in the session handle to track the end user identity.

The following example shows how to use <code>OCIAttrSet</code> with the <code>ATTR_CLIENT_IDENTIFIER</code> parameter. The <code>user_id</code> setting refers to a variable that stores the ID of the user who is logging on.

DBMS_SESSION package. Use the DBMS_SESSION.SET_IDENTIFIER procedure to set
the client identifier for the global application context. For example, assuming you are
storing the ID of the user logging on in a variable called user_id, you would enter the
following line into the middle-tier application code:

```
DBMS_SESSION.SET_IDENTIFIER(user_id);
```

Note:

When the application generates a session ID for use as a <code>CLIENT_IDENTIFIER</code>, then the session ID must be suitably random and protected over the network by encryption. If the session ID is not random, then a malicious user could guess the session ID and access the data of another user. If the session ID is not encrypted over the network, then a malicious user could retrieve the session ID and access the connection.

You can encrypt the session ID by using network data encryption. See Configuring Oracle Database Native Network Encryption and Data Integrity for more information.

13.4.7.3.3 Checking the Value of the Client Identifier

For both <code>OCIAttrSet</code> and <code>DBMS_SESSION.SET_IDENTIFIER</code>, you can check the value of the client identifier.

• To check the value of the client identifier, use one of the of the following approaches:

To check it using the SYS CONTEXT function:

```
SELECT SYS CONTEXT('userenv', 'client identifier') FROM DUAL;
```

To check it by querying the V\$SESSION view:

```
SELECT CLIENT IDENTIFIER from V$SESSION;
```

13.4.7.4 Step 3: Clear the Session Data Using a Middle-Tier Application

The application context exists entirely within memory.

When the user exits a session, you must clear the context for the client_identifier value. This releases memory and prevents other users from accidentally using any left over values

- To clear session data when a user exits a session, use either of the following methods in the middle-tier application code:
 - Clearing the client identifier when a user exits a session. Use the DBMS_SESSION.CLEAR_IDENTIFIER procedure. For example:

```
DBMS SESSION.CLEAR IDENTIFIER;
```

Continuing the session but still clearing the context. If you want the session to
continue, but you still need to clear the context, use the DBMS_SESSION.CLEAR_CONTEXT
or the DBMS_SESSION.CLEAR_ALL_CONTEXT procedure. For example:

```
DBMS_SESSION.CLEAR_CONTEXT(namespace, client_identifier, attribute);
```

The CLEAR_CONTEXT procedure clears the context for the current user. To clear the context values for all users, for example, when you need to shut down the application server, use the CLEAR ALL CONTEXT procedure.

Global application context values are available until they are cleared, so you should use CLEAR_CONTEXT or CLEAR_ALL_CONTEXT to ensure that other sessions do not have access to these values.

13.4.8 Tutorial: Creating a Global Application Context That Uses a Client Session ID

This tutorial demonstrates how you can create a global application context that uses a client session ID.

About This Tutorial

This tutorial shows how to create a global application context that uses a client session ID for a lightweight user application.

Step 1: Create User Accounts

A security administrator will manage the application context and its package, and a user account will own the connection pool.

- Step 2: Create the Global Application Context
 - Next, you are ready to create the global application context.
- Step 3: Create a Package for the Global Application Context
 The PL/SQL package will manage the global application context that you created.
- Step 4: Test the Newly Created Global Application Context
 At this stage, you are ready to explore how this global application context and session ID settings work.



- Step 5: Modify the Session ID and Test the Global Application Context Again
 Next, clear and then modify the session ID and test the global application context again.
- Step 6: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

13.4.8.1 About This Tutorial

This tutorial shows how to create a global application context that uses a client session ID for a lightweight user application.

It demonstrates how to control nondatabase user access by using a connection pool. If you are using a multitenant environment, then this tutorial applies to the current PDB only.

13.4.8.2 Step 1: Create User Accounts

A security administrator will manage the application context and its package, and a user account will own the connection pool.

1. Log on to SQL*Plus as SYS with the SYSDBA administrative privilege.

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

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

For example:

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

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

Create the local user account sysadmin_ctx, who will administer the global application context.

```
CREATE USER sysadmin_ctx IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION, CREATE ANY CONTEXT, CREATE PROCEDURE TO sysadmin_ctx;
GRANT EXECUTE ON DBMS_SESSION TO sysadmin_ctx;
```

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

Create the local database account apps user, who will own the connection pool.

```
CREATE USER apps_user IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION TO apps user;
```

Replace password with a password that is secure.

13.4.8.3 Step 2: Create the Global Application Context

Next, you are ready to create the global application context.

Log on as the security administrator sysadmin ctx.

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

2. Create the cust_ctx global application context.

```
CREATE CONTEXT global cust ctx USING cust ctx pkg ACCESSED GLOBALLY;
```

The <code>cust_ctx</code> context is created and associated with the schema of the security administrator <code>sysadmin_ctx</code>. However, the <code>sys</code> schema owns the application context.

13.4.8.4 Step 3: Create a Package for the Global Application Context

The PL/SQL package will manage the global application context that you created.

1. As sysadmin ctx, create the following PL/SQL package:

```
CREATE OR REPLACE PACKAGE cust ctx pkg
  PROCEDURE set_session_id(session_id_p IN NUMBER);
  PROCEDURE set_cust_ctx(sec_level_attr IN VARCHAR2,
    sec_level_val IN VARCHAR2);
  PROCEDURE clear hr session(session id p IN NUMBER);
  PROCEDURE clear hr context;
  END;
CREATE OR REPLACE PACKAGE BODY cust ctx pkg
  session id global NUMBER;
 PROCEDURE set session id(session id p IN NUMBER)
 BEGIN
  session id global := session id p;
  DBMS SESSION.SET IDENTIFIER(session id p);
 END set session id;
 PROCEDURE set cust ctx(sec level attr IN VARCHAR2, sec level val IN VARCHAR2)
  BEGIN
  DBMS SESSION.SET CONTEXT (
   namespace => 'global cust ctx',
   attribute => sec level attr,
   value => sec level val,
   username => USER, -- Retrieves the session user, in this case, apps_user
   client id => session id global);
  END set cust ctx;
  PROCEDURE clear hr session(session id p IN NUMBER)
  AS
     DBMS SESSION.SET IDENTIFIER(session id p);
     DBMS SESSION.CLEAR IDENTIFIER;
  END clear hr session;
 PROCEDURE clear hr context
 AS
 BEGIN
  DBMS SESSION.CLEAR CONTEXT('global_cust_ctx', session_id_global);
 END clear hr context;
 END;
```

For a detailed explanation of how this type of package works, see Example 13-9.

Grant EXECUTE privileges on the cust_ctx_pkg package to the connection pool owner, apps user.

```
GRANT EXECUTE ON cust ctx pkg TO apps user;
```

13.4.8.5 Step 4: Test the Newly Created Global Application Context

At this stage, you are ready to explore how this global application context and session ID settings work.

1. Log on to SQL*Plus as the connection pool owner, user apps user.

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

2. When the connection pool user logs on, the application sets the client session identifier as follows:

```
BEGIN
   sysadmin_ctx.cust_ctx_pkg.set_session_id(34256);
END;
//
```

- Test the value of the client session identifier.
 - a. Set the session ID:

```
EXEC sysadmin ctx.cust ctx pkg.set session id(34256);
```

b. Check the session ID:

```
SELECT SYS CONTEXT('userenv', 'client identifier') FROM DUAL;
```

The following output should appear:

4. Set the global application context as follows:

```
EXEC sysadmin_ctx.cust_ctx_pkg.set_cust_ctx('Category', 'Gold Partner');
EXEC sysadmin_ctx.cust_ctx_pkg.set_cust_ctx('Benefit Level', 'Highest');
```

(In a real-world scenario, the middle-tier application would set the global application context values, similar to how the client session identifier was set in Step 2.)

5. Enter the following SELECT SYS_CONTEXT statement to check that the settings were successful:

```
col category format a13
col benefit_level format a14

SELECT SYS_CONTEXT('global_cust_ctx', 'Category') category,
SYS_CONTEXT('global_cust_ctx', 'Benefit Level') benefit_level FROM DUAL;
```

The following output should appear:

What apps_user has done here, within the client session 34256, is set a global application context on behalf of a nondatabase user. This context sets the Category and Benefit Level DBMS_SESSION.SET_CONTEXT attributes to be Gold Partner and Highest, respectively. The context exists only for user apps_user with client ID 34256. When a nondatabase user logs in, behind the scenes, he or she is really logging on as the connection pool user apps_user. Hence, the Gold Partner and Highest context values are available to the nondatabase user.

Suppose the user had been a database user and could log in without using the intended application. (For example, the user logs in using SQL*Plus.) Because the user has not logged in through the connection pool user <code>apps_user</code>, the global application context appears empty to our errant user. This is because the context was created and set under the <code>apps_user</code> session. If the user runs the <code>SELECT SYS CONTEXT</code> statement, then the following output appears:

```
CATEGORY BENEFIT_LEVEL
```

13.4.8.6 Step 5: Modify the Session ID and Test the Global Application Context Again

Next, clear and then modify the session ID and test the global application context again.

1. As user apps user, clear the session ID.

```
EXEC sysadmin ctx.cust ctx pkg.clear hr session(34256);
```

2. Check the global application context settings again.

```
SELECT SYS_CONTEXT('global_cust_ctx', 'Category') category,
SYS_CONTEXT('global_cust_ctx', 'Benefit Level') benefit_level FROM DUAL;
CATEGORY BENEFIT_LEVEL
```

Because apps_user has cleared the session ID, the global application context settings are no longer available.

3. Restore the session ID to 34256, and then check the context values.

```
EXEC sysadmin_ctx.cust_ctx_pkg.set_session_id(34256);

SELECT SYS_CONTEXT('global_cust_ctx', 'Category') category,
SYS_CONTEXT('global_cust_ctx', 'Benefit Level') benefit level FROM DUAL;
```

The following output should appear:

As you can see, resetting the session ID to 34256 brings the application context values back again. To summarize, the global application context must be set only *once* for this user, but the client session ID must be set *each time* the user logs on.

4. Now try clearing and then checking the global application context values.

```
EXEC sysadmin_ctx.cust_ctx_pkg.clear_hr_context;

SELECT SYS_CONTEXT('global_cust_ctx', 'Category') category,
SYS_CONTEXT('global_cust_ctx', 'Benefit Level') benefit_level FROM DUAL;
```

The following output should appear:

```
CATEGORY BENEFIT_LEVEL
```

At this stage, the client session ID, 34256 is still in place, but the application context settings no longer exist. This enables you to continue the session for this user but without using the previously set application context values.

13.4.8.7 Step 6: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

1. Connect as SYS with the SYSDBA administrative privilege.

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

2. Drop the global application context.

```
DROP CONTEXT global_cust_ctx;
```

Remember that even though $sysadmin_ctx$ created the global application context, it is owned by the sys schema.

3. Drop the two sample users.

```
DROP USER sysadmin_ctx CASCADE;
DROP USER apps user;
```

13.4.9 Global Application Context Processes

A simple global application context uses a database user account create the user session; a global application context is for lightweight users.

- Simple Global Application Context Process
 In a simple global application context process, the application uses a database user to create a user session.
- Global Application Context Process for Lightweight Users
 You can set a global application contexts for lightweight users.

13.4.9.1 Simple Global Application Context Process

In a simple global application context process, the application uses a database user to create a user session.

The value for the context attribute of a simple global application context process can be retrieved from a SELECT statement.

Consider the application server, AppSvr, which has assigned the client identifier 12345 to client SCOTT. The AppSvr application uses the SCOTT user to create a session. (In other words, it is not a connection pool.) The value assigned to the context attribute can come from anywhere, for example, from running a SELECT statement on a table that holds the responsibility codes for users. When the application context is populated, it is stored in memory. As a result, any action that needs the responsibility code can access it quickly with a SYS_CONTEXT call, without the overhead of accessing a table. The only advantage of a global context over a local context in this case is if SCOTT were changing applications frequently and used the same context in each application.

The following steps show how the global application context process sets the client identifier for SCOTT:

The administrator creates a global context namespace by using the following statement:

```
CREATE OR REPLACE CONTEXT hr ctx USING hr.init ACCESSED GLOBALLY;
```



2. The administrator creates a PL/SQL package for the hr_ctx application context to indicate that, for this client identifier, there is an application context called responsibility with a value of 13 in the HR namespace.:

```
CREATE OR REPLACE PROCEDURE hr.init
AS
BEGIN

DBMS_SESSION.SET_CONTEXT(
   namespace => 'hr_ctx',
   attribute => 'responsibility',
   value => '13',
   username => 'SCOTT',
   client_id => '12345');
END;
/
```

This PL/SQL procedure is stored in the HR database schema, but typically it is stored in the schema of the security administrator.

3. The AppSvr application issues the following command to indicate the connecting client identity each time scott uses AppSvr to connect to the database:

```
EXEC DBMS SESSION.SET IDENTIFIER('12345');
```

- 4. When there is a SYS_CONTEXT('hr_ctx', 'responsibility') call within the database session, the database matches the client identifier, 12345, to the global context, and then returns the value 13.
- 5. When exiting this database session, AppSvr clears the client identifier by issuing the following procedure:

```
EXEC DBMS SESSION.CLEAR IDENTIFIER();
```

6. To release the memory used by the application context, AppSvr issues the following procedure:

```
DBMS_SESSION.CLEAR_CONTEXT('hr_ctx', '12345');
```

CLEAR_CONTEXT is needed when the user session is no longer active, either on an explicit logout, timeout, or other conditions determined by the AppSvr application.

Note:

After a client identifier in a session is cleared, it becomes a NULL value. This implies that subsequent SYS_CONTEXT calls only retrieve application contexts with NULL client identifiers, until the client identifier is set again using the SET IDENTIFIER interface.

13.4.9.2 Global Application Context Process for Lightweight Users

You can set a global application contexts for lightweight users.

You can configure this access so that when other users log in, they cannot access the global application context.

The following steps show the global application context process for a lightweight user application. The lightweight user, robert, is not known to the database through the application.

1. The administrator creates the global context namespace by using the following statement:

```
CREATE CONTEXT hr ctx USING hr.init ACCESSED GLOBALLY;
```

- 2. The HR application server, AppSvr, starts and then establishes multiple connections to the HR database as the appsmgr user.
- 3. User robert logs in to the HR application server.
- 4. AppSvr authenticates robert to the application.
- 5. AppSvr assigns a temporary session ID (or uses the application user ID), 12345, for this connection.
- The session ID is returned to the Web browser used by robert as part of a cookie or is maintained by AppSvr.
- 7. AppSvr initializes the application context for this client by calling the hr.init package, which issues the following statements:

```
DBMS_SESSION.SET_CONTEXT( 'hr_ctx', 'id', 'robert', 'APPSMGR', 12345 );
DBMS_SESSION.SET_CONTEXT( 'hr_ctx', 'dept', 'sales', 'APPSMGR', 12345 );
```

8. AppSvr assigns a database connection to this session and initializes the session by issuing the following statement:

```
DBMS SESSION.SET IDENTIFIER( 12345 );
```

All SYS_CONTEXT calls within this database session return application context values that belong only to the client session.

```
For example, SYS CONTEXT('hr','id') returns the value robert.
```

10. When finished with the session, AppSvr issues the following statement to clean up the client identity:

```
DBMS SESSION.CLEAR IDENTIFIER ( );
```

Even if another user logged in to the database, this user cannot access the global context set by AppSvr, because AppSvr specified that only the application with user APPSMGR logged in can see it. If AppSvr used the following, then any user session with client ID set to 12345 can see the global context:

```
DBMS_SESSION.SET_CONTEXT( 'hr_ctx', 'id', 'robert', NULL , 12345 );
DBMS_SESSION.SET_CONTEXT( 'hr_ctx', 'dept', 'sales', NULL , 12345 );
```

Setting USERNAME to NULL enables different users to share the same context.

Note:

Be aware of the security implication of different settings of the global context. Null in the user name means that any user can access the global context. A Null client ID in the global context means that a session with an uninitialized client ID can access the global context. To ensure that only the user who has logged on can access the session, specify USER instead of Null.

You can guery the client identifier set in the session as follows:

```
SELECT SYS CONTEXT ('USERENV', 'CLIENT IDENTIFIER') FROM DUAL;
```

The following output should appear:

A security administrator can see which sessions have the client identifier set by querying the V\$SESSION view for the CLIENT IDENTIFIER and USERNAME, for example:

```
COL client_identifier format a18 SELECT CLIENT IDENTIFIER, USERNAME from V$SESSION;
```

The following output should appear:

```
CLIENT_IDENTIFIER USERNAME
-----
12345 APPSMGR
```

To check the amount of global context area (in bytes) being used, use the following query:

```
SELECT SYS CONTEXT ('USERENV', 'GLOBAL CONTEXT MEMORY') FROM DUAL;
```

The following output should appear:

See Also:

For more information about using the CLIENT_IDENTIFIER predefined attribute of the USERENV application context:

- Use of the CLIENT_IDENTIFIER Attribute to Preserve User Identity
- Oracle Database SQL Language Reference
- Oracle Call Interface Programmer's Guide

13.5 Using Client Session-Based Application Contexts

A client session-based application context is stored in the User Global Area (UGA).

- About Client Session-Based Application Contexts
 Oracle Call Interface (OCI) functions can set and clear the User Global Area (UGA) user session information.
- Setting a Value in the CLIENTCONTEXT Namespace
 Oracle Call Interface (OCI) can set the CLIENTCONTEXT namespace.
- Retrieving the CLIENTCONTEXT Namespace
 You can use Oracle Call Interface to retrieve the CLIEINTCONTEXT namespace.
- Example: Retrieving a Client Session ID Value for Client Session-Based Contexts
 The OCI OCIStmtExecute call can retrieve client session ID values for client session-based contexts.
- Clearing a Setting in the CLIENTCONTEXT Namespace
 You can use Oracle Call Interface to clear the CLIENTCONTEXT namespace.

Clearing All Settings in the CLIENTCONTEXT Namespace
 You can use Oracle Call Interface (OCI) to clear the CLIENTCONTEXT namespace.

13.5.1 About Client Session-Based Application Contexts

Oracle Call Interface (OCI) functions can set and clear the User Global Area (UGA) user session information.

The advantage of this type of application context in a session-based application context is that an individual application can check for specific nondatabase user session data, rather than having the database perform this task. Another advantage is that the calls to set the application context value are included in the next call to the server, which improves performance.

However, be aware that application context security is compromised with a client session-based application context: any application user can set the client application context, and no check is performed in the database.

You configure the client session-based application context for the client application only. You do not configure any settings on the database server to which the client connects. Any application context settings in the database server do not affect the client session-based application context.

To configure a client session-based application context, use the <code>OCIAppCtxSet</code> OCI function. A client session-based application context uses the <code>CLIENTCONTEXT</code> namespace, updatable by any OCI client or by the existing <code>DBMS_SESSION</code> package for application context. Oracle Database performs no privilege or package security checks for this type.

The CLIENTCONTEXT namespace enables a single application transaction to both change the user context information and use the same user session handle to service the new user request. You can set or clear individual values for attributes in the CLIENTCONTEXT namespace, or clear all their values.

- An OCI client uses the OCIAppCtx function to set variable length data for the namespace, called OCISessionHandle. The OCI network single, round-trip transport sends all the information to the server in one round-trip. On the server side, you can query the application context information by using the SYS_CONTEXT SQL function on the namespace. For example:
- A JDBC client uses the oracle.jdbc.internal.OracleConnection function to achieve the same purposes.

Any user can set, clear, or collect the information in the CLIENTCONTEXT namespace, because it is not protected by package-based security.



Oracle Call Interface Programmer's Guide for more information about client application contexts



13.5.2 Setting a Value in the CLIENTCONTEXT Namespace

Oracle Call Interface (OCI) can set the CLIENTCONTEXT namespace.

 To set a value in the CLIENTCONTEXT namespace, use the OCIAppCTXSet command, in the following syntax:

In this specification:

- session handle represents the OCISessionHandle namespace.
- attribute_name is the name of the attribute. For example, responsibility, with a length
 of 14.
- attribute value is the value of the attribute. For example, manager, with a length of 7.



Oracle Call Interface Programmer's Guide for details about the OCIAppCtx function

13.5.3 Retrieving the CLIENTCONTEXT Namespace

You can use Oracle Call Interface to retrieve the CLIEINTCONTEXT namespace.

- To retrieve the CLIENTCONTEXT namespace, use the OCIStmtExecute call with either of the following statements:
 - SELECT SYS CONTEXT ('CLIENTCONTEXT', 'attribute-1') FROM DUAL;
 - SELECT VALUE FROM SESSION_CONTEXT WHERE NAMESPACE='CLIENTCONTEXT' AND ATTRIBUTE='attribute-1';

The attribute-1 value can be any attribute value that has already been set in the CLIENTCONTEXT namespace. Oracle Database only retrieves the set attribute; otherwise, it returns <code>NULL</code>. Typically, you set the attribute by using the <code>OCIAppCtxSet</code> call. In addition, you can embed a <code>DBMS SESSION.SET CONTEXT</code> call in the OCI code to set the attribute value.

13.5.4 Example: Retrieving a Client Session ID Value for Client Session-Based Contexts

The OCI ${\tt OCIStmtExecute}$ call can retrieve client session ID values for client session-based contexts.

Example 13-10 shows how to use the OCIStmtExecute call to retrieve a client session ID value.

Example 13-10 Retrieving a Client Session ID Value for Client Session-Based Contexts

```
oratext clientid[31];
OCIDefine *defnp1 = (OCIDefine *) 0;
OCIStmt *statementhndle;
oratext *selcid = (oratext *)"SELECT SYS_CONTEXT('CLIENTCONTEXT',
```



```
attribute) FROM DUAL";

OCIStmtPrepare(statementhndle, errhp, selcid, (ub4) strlen((char *) selcid),
   (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT);

OCIDefineByPos(statementhndle, &defnp1, errhp, 1, (dvoid *)clientid, 31,
   SQLT_STR, (dvoid *) 0, (ub2 *) 0, (ub2 *) 0, OCI_DEFAULT);

OCIStmtExecute(servhndle, statementhndle, errhp, (ub4) 1, (ub4) 0,
   (CONST OCISnapshot *) NULL, (OCISnapshot *) NULL, OCI_DEFAULT);

printf("CLIENT_IDENTIFIER = %s \n", clientid);
```

In this example:

- oratext, OCIDefine, OCIStmt, and oratext create variables to store the client session ID, reference call for OCIDefine, the statement handle, and the SELECT statement to use.
- OCIStmtPrepare prepares the statement selcid for execution.
- OCIDefineByPos defines the output variable clientid for client session ID.
- OCIStmtExecute executes the statement in the selcid variable.
- printf prints the formatted output for the retrieved client session ID.

13.5.5 Clearing a Setting in the CLIENTCONTEXT Namespace

You can use Oracle Call Interface to clear the CLIENTCONTEXT namespace.

- To clear a setting in CLIENTCONTEXT, set the value to NULL or to an empty string by using one of the following commands:
 - The following command sets the empty string to zero:

This following command sets the empty string to a blank value:

13.5.6 Clearing All Settings in the CLIENTCONTEXT Namespace

You can use Oracle Call Interface (OCI) to clear the CLIENTCONTEXT namespace.

To clear the namespace, use the OCIAppCtxClearAll command in the following form:



13.6 Application Context Data Dictionary Views

Oracle Database provides data dictionary views that provide information about application contexts.

Table 13-3 lists these data dictionary views.

Table 13-3 Data Dictionary Views That Display Information about Application Contexts

View	Description
ALL_CONTEXT	Describes all context namespaces in the current session for which attributes and values were specified using the <code>DBMS_SESSION.SET_CONTEXT</code> procedure. It lists the namespace and its associated schema and PL/SQL package.
ALL_POLICY_CONTEXTS	Describes the driving contexts defined for the synonyms, tables, and views accessible to the current user. (A driving context is a context used in a Virtual Private Database policy.)
DBA_CONTEXT	Provides all context namespace information in the database. Its columns are the same as those in the <code>ALL_CONTEXT</code> view, except that it includes the <code>TYPE</code> column. The <code>TYPE</code> column describes how the application context is accessed or initialized.
DBA_OBJECTS	Provides the names of existing application contexts. Query the <code>OBJECT_TYPE</code> column of the <code>DBA_OBJECTS</code> view, as follows:
	SELECT OBJECT_NAME FROM DBA_OBJECTS WHERE OBJECT_TYPE ='CONTEXT';
DBA_POLICY_CONTEXTS	Describes all driving contexts in the database that were added by the DBMS_RLS.ADD_POLICY_CONTEXT procedure. Its columns are the same as those in ALL_POLICY_CONTEXTS.
SESSION_CONTEXT	Describes the context attributes and their values set for the current session.
USER_POLICY_CONTEXTS	Describes the driving contexts defined for the synonyms, tables, and views owned by the current user. Its columns (except for <code>OBJECT_OWNER</code>) are the same as those in <code>ALL_POLICY_CONTEXTS</code> .
V\$CONTEXT	Lists set attributes in the current PDB session. Users do not have access to this view unless you grant the user the SELECT privilege on it.
V\$SESSION	Lists detailed information about each current PDB session. Users do not have access to this view unless you grant the user the SELECT privilege on it.



Tip:

In addition to these views, check the database trace file if you find errors when running applications that use application contexts. The <code>USER_DUMP_DEST</code> initialization parameter sets the directory location of the trace files. You can find the value of this parameter by issuing <code>SHOW PARAMETER USER_DUMP_DEST</code> in SQL*Plus.

Related Topics

- Oracle Database Reference
- Oracle Database SQL Tuning Guide



Using Oracle Virtual Private Database to Control Data Access

Oracle Virtual Private Database (VPD) enables you to filter users who access data.

- About Oracle Virtual Private Database
 Oracle Virtual Private Database (VPD) provides important benefits for filtering user access
 to data.
- Components of an Oracle Virtual Private Database Policy
 A VPD policy uses a function to generate the dynamic WHERE clause, and a policy to attach
 the function to objects to protect.
- Configuration of Oracle Virtual Private Database Policies
 The DBMS_RLS PL/SQL package can configure Oracle Virtual Private Database (VPD) policies.
- Tutorials: Creating Oracle Virtual Private Database Policies
 These tutorials show how to create a simple and a database session-based Oracle Virtual
 Private policy, and how to create policy groups.
- How Oracle Virtual Private Database Works with Other Oracle Features
 You should be aware of the impact of using Oracle Virtual Private Database with other
 Oracle features.
- Oracle Virtual Private Database Data Dictionary Views
 Oracle Database provides data dictionary views that list information about Oracle Virtual Private Database policies.

14.1 About Oracle Virtual Private Database

Oracle Virtual Private Database (VPD) provides important benefits for filtering user access to data.

- What Is Oracle Virtual Private Database?
 Oracle Virtual Private Database (VPD) creates security policies to control database access at the row and column level.
- Benefits of Using Oracle Virtual Private Database Policies
 Oracle Virtual Private Database policies provide the important benefits.
- Who Can Create Oracle Virtual Private Database Policies?
 The DBMS RLS PL/SQL package enables you to create VPD policies.
- Privileges to Run Oracle Virtual Private Database Policy Functions
 You should be aware of the correct privileges for running Oracle Virtual Private Database
 (VPD) policy functions.
- Oracle Virtual Private Database Use with an Application Context
 You can use application contexts with Oracle Virtual Private Database policies.
- Oracle Virtual Private Database in a Multitenant Environment
 You can create Virtual Private Database policies in an application root for use throughout any associated application PDBs.

14.1.1 What Is Oracle Virtual Private Database?

Oracle Virtual Private Database (VPD) creates security policies to control database access at the row and column level.

Essentially, Oracle Virtual Private Database adds a dynamic WHERE clause to a SQL statement that is issued against the table, view, or synonym to which an Oracle Virtual Private Database security policy was applied.

Oracle Virtual Private Database enforces security, to a fine level of granularity, directly on database tables, views, or synonyms. Because you attach security policies directly to these database objects, and the policies are automatically applied whenever a user accesses data, there is no way to bypass security.

When a user directly or indirectly accesses a table, view, or synonym that is protected with an Oracle Virtual Private Database policy, Oracle Database dynamically modifies the SQL statement of the user. This modification creates a WHERE condition (called a predicate) returned by a function implementing the security policy. Oracle Database modifies the statement dynamically, transparently to the user, using any condition that can be expressed in or returned by a function. You can apply Oracle Virtual Private Database policies to SELECT, INSERT, UPDATE, INDEX, and DELETE statements.

For example, suppose a user performs the following query:

```
SELECT * FROM OE.ORDERS;
```

The Oracle Virtual Private Database policy dynamically appends the statement with a WHERE clause. For example:

```
SELECT * FROM OE.ORDERS
WHERE SALES_REP ID = 159;
```

In this example, the user can only view orders by Sales Representative 159.

If you want to filter the user based on the session information of that user, such as the ID of the user, then you can create the WHERE clause to use an application context. For example:

```
SELECT * FROM OE.ORDERS
WHERE SALES_REP_ID = SYS_CONTEXT('USERENV', 'SESSION_USER');
```

Note the following:

- Oracle Database release 12c introduced Real Application Security (RAS) to supersede VPD. Oracle recommends that you use RAS for new projects that require row and column level access controls for their applications.
- Oracle Database does not protect tables and views that have VPD policies against the SYS
 user and against users who have an out-of-the-box database administrator role. The
 Oracle Database-supplied DBA role has privileges that can alter and remove VPD policies,
 and hence can access table and view data.
- Oracle Virtual Private Database does not support filtering for DDLs, such as TRUNCATE or ALTER TABLE statements.

Related Topics

Auditing of Oracle Virtual Private Database Predicates
 The unified audit trail automatically captures the predicates that are used in Oracle Virtual Private Database (VPD) policies.

14.1.2 Benefits of Using Oracle Virtual Private Database Policies

Oracle Virtual Private Database policies provide the important benefits.

- Security Policies Based on Database Objects Rather Than Applications
 Oracle Virtual Private Database provides benefits in security, simplicity, and flexibility.
- Control Over How Oracle Database Evaluates Policy Functions Running policy functions multiple times can affect performance.

14.1.2.1 Security Policies Based on Database Objects Rather Than Applications

Oracle Virtual Private Database provides benefits in security, simplicity, and flexibility.

Attaching Oracle Virtual Private Database security policies to database tables, views, or synonyms, rather than implementing access controls in all your applications, provides the following benefits:

- Security. Associating a policy with a database table, view, or synonym can solve a
 potentially serious application security problem. Suppose a user is authorized to use an
 application, and then drawing on the privileges associated with that application, wrongfully
 modifies the database by using an ad hoc query tool, such as SQL*Plus. By attaching
 security policies directly to tables, views, or synonyms, fine-grained access control ensures
 that the same security is in force, no matter how a user accesses the data.
- **Simplicity.** You add the security policy to a table, view, or synonym only once, rather than repeatedly adding it to each of your table-based, view-based, or synonym-based applications.
- Flexibility. You can have one security policy for SELECT statements, another for INSERT statements, and still others for UPDATE and DELETE statements. For example, you might want to enable Human Resources clerks to have SELECT privileges for all employee records in their division, but to update only salaries for those employees in their division whose last names begin with A through F. Furthermore, you can create multiple policies for each table, view, or synonym.

14.1.2.2 Control Over How Oracle Database Evaluates Policy Functions

Running policy functions multiple times can affect performance.

You can control the performance of policy functions by configuring how Oracle Database caches the Oracle Virtual Private Database predicates.

The following options are available:

- Evaluate the policy once for each query (static policies).
- Evaluate the policy only when an application context within the policy function changes (context-sensitive policies).
- Evaluate the policy each time it is run (dynamic policies).



Related Topics

Optimizing Performance by Using Oracle Virtual Private Database Policy Types
You can optimize performance by using the Oracle Virtual Private Database (VPD) the
dynamic, static, or shared policy types.

14.1.3 Who Can Create Oracle Virtual Private Database Policies?

The DBMS RLS PL/SQL package enables you to create VPD policies.

Users who have been granted the EXECUTE privilege on the DBMS_RLS PL/SQL package can create Oracle Virtual Private Database policies. As with all privileges, only grant this privilege to trusted users. You can find the privileges that a user has been granted by querying the DBA_SYS_PRIVS data dictionary view.

14.1.4 Privileges to Run Oracle Virtual Private Database Policy Functions

You should be aware of the correct privileges for running Oracle Virtual Private Database (VPD) policy functions.

For greater security, the Oracle Virtual Private Database policy function runs as if it had been declared with definer's rights.

Do not declare it as invoker's rights because this can confuse yourself and other users who maintain the code.



Oracle Database PL/SQL Language Reference for detailed information about definer's rights

14.1.5 Oracle Virtual Private Database Use with an Application Context

You can use application contexts with Oracle Virtual Private Database policies.

When you create an application context, it securely caches user information. Only the designated application package can set the cached environment. It cannot be changed by the user or outside the package. In addition, because the data is cached, performance is increased.

For example, suppose you want to base access to the <code>ORDERS_TAB</code> table on the customer ID number. Rather than querying the customer ID number for a logged-in user each time you need it, you could store the number in the application context. Then, the customer number is available in the session when you need it.

Application contexts are especially helpful if your security policy is based on multiple security attributes. For example, if a policy function bases a WHERE predicate on four attributes (such as employee number, cost center, position, spending limit), then multiple subqueries must execute to retrieve this information. Instead, if this data is available through an application context, then performance is much faster.

You can use an application context to return the correct security policy, enforced through a predicate. For example, consider an order entry application that enforces the following rules: customers only see their own orders, and clerks see all orders for all customers. These are two

different policies. You could define an application context with a position attribute, and this attribute could be accessed within the policy function to return the correct predicate, depending on the value of the attribute. Thus, you can enable a user in the clerk position to retrieve all orders, but a user in the customer position can see only those records associated with that particular user.

To design a fine-grained access control policy that returns a specific predicate for an attribute, you need to access the application context within the function that implements the policy. For example, suppose you want to limit customers to seeing only their own records. The user performs the following query:

```
SELECT * FROM orders tab
```

Fine-grained access control dynamically modifies this query to include the following WHERE predicate:

```
SELECT * FROM orders_tab
WHERE custno = SYS CONTEXT ('order entry', 'cust num');
```

Continuing with the preceding example, suppose you have 50,000 customers, and you do not want to have a different predicate returned for each customer. Customers all share the same WHERE predicate, which prescribes that they can only see their own orders. It is merely their customer numbers that are different.

Using application context, you can return one WHERE predicate within a policy function that applies to 50,000 customers. As a result, there is one shared cursor that executes differently for each customer, because the customer number is evaluated at execution time. This value is different for every customer. Use of application context in this case provides optimum performance, and at row-level security.

The SYS_CONTEXT function works much like a bind variable; only the SYS_CONTEXT arguments are constants.

Related Topics

Using Application Contexts to Retrieve User Information
 An application context stores user identification that can enable or prevent a user from accessing data in the database.

14.1.6 Oracle Virtual Private Database in a Multitenant Environment

You can create Virtual Private Database policies in an application root for use throughout any associated application PDBs.

The CDB restriction applies to shared context sensitive policies and views related to Virtual Private Database policies as well. You cannot create a Virtual Private Database policy for an entire multitenant environment.

With regard to application containers, you can create Virtual Private Database policies to protect application common objects by applying the common policy to all PDBs that belong to the application root. In other words, when you install an application in the application root, all the common Virtual Private Database policies that protect the common objects will be applied to and immediately enforced for all PDBs in the application container.

Note the following:

 You can only create the common Virtual Private Database policy and its associated PL/SQL function in the application root and only attach it to application common objects. If the function is not in the same location as the policy, then an error is raised at runtime.

- A Virtual Private Database policy that is applied to common objects is considered a common policy that will be automatically enforced in PDBs that belong to the application container when it accesses the application common objects from application PDBs.
- Application common Virtual Private Database policies can only protect application common objects.
- A Virtual Private Database policy that is applied to application common objects in the
 application root and is applied to all application PDBs is considered a common Virtual
 Private Database policy. A policy that is applied to a local database table and enforced in
 one PDB is considered a local Virtual Private Database policy.

For example, if policy VPD_P1 is applied to the application common table T1 in the application root, then it is a considered to be a common policy. It will be enforced in each application PDB. If a policy named VPD_P1 is applied to a local table called T1 in PDB1, then it is considered a local policy, which means that it affects only PDB1. If a policy called VPD_P1 is applied to a local table T1 in the application root, then it is still considered a local policy because it affects only the application root. This concept applies to other operations, such as enabling, disabling, and removing Virtual Private Database policies.

- Application common Virtual Private Database policies only protect application common objects, while local Virtual Private Database policies only protect local objects.
- If you are using application contexts, then ensure common database session-based application contexts and common global application context objects are used in the common Virtual Private Database configuration.
- Application container Virtual Private Database policies are stored in the application root.
 PDBs store only local policies. If you plug a PDB into the application container, then the common policies are not converted to local policies. Instead, Oracle Database loads them from the application root and enforces them in the local PDB when the policies access common objects in the local PDB.

14.2 Components of an Oracle Virtual Private Database Policy

A VPD policy uses a function to generate the dynamic WHERE clause, and a policy to attach the function to objects to protect.

- Function to Generate the Dynamic WHERE Clause
 The Oracle Virtual Private Database (VPD) function defines the restrictions that you want to enforce.
- Policies to Attach the Function to the Objects You Want to Protect
 The Oracle Virtual Private Database policy associates the VPD function with a table, view, or synonym.

14.2.1 Function to Generate the Dynamic WHERE Clause

The Oracle Virtual Private Database (VPD) function defines the restrictions that you want to enforce.

To generate the Oracle Virtual Private Database (VPD) dynamic WHERE clause (predicate), you must create a function (not a procedure) that defines these restrictions. This function is a definer's rights function. Oracle Database generates the predicate with the VPD policy function authorized by the owner but in the same current user session such that the PL/SQL global variables that are defined in the function will be used.



Usually, the security administrator creates this function in their own schema. For more complex behavior, such as including calls to other functions or adding checks to track failed logon attempts, create these functions within a package.

The function must have the following behavior:

- It must take as arguments a schema name and an object (table, view, or synonym) name as inputs. Define input parameters to hold this information, but do not specify the schema and object name themselves within the function. The policy that you create to attach the function to the objects that you want to protect, using the DBMS_RLS package, provides the names of the schema, and object to which the policy will apply. You must create the parameter for the schema first, followed by the parameter for the object.
- It must provide a return value for the WHERE clause predicate that will be generated. The return value for the WHERE clause is always a VARCHAR2 data type.
- It must generate a valid WHERE clause. This code can be as simple in that it applies to every user who logs in the database instance, but in most cases, you may want to design the WHERE clause to be different for each user, each group of users, or each application that accesses the objects you want to protect. For example, if a manager logs in, the WHERE clause can be specific to the rights of that particular manager. You can do this by incorporating an application context, which accesses user session information, into the WHERE clause generation code.

You can create Oracle Virtual Private Database functions that do not use an application context, but an application context creates a much stronger Oracle Virtual Private Database policy, by securely basing user access on the session attributes of that user, such as the user ID.

In addition, you can embed C or Java calls to access operating system information or to return where clauses from an operating system file or other source.

- It must not select from a table within the associated policy function. Although you can define a policy against a table, you cannot select that table from within the policy that was defined against the table.
- It must be a pure function. The VPD function must rely only on the application context and the arguments that are passed to the function to generate the WHERE clause. This function must not depend on the package variables.



If you plan to run the function across different editions, you can control the results of the function: whether the results are uniform across all editions, or specific to the edition in which the function is run.

Related Topics

- Policies to Attach the Function to the Objects You Want to Protect
 The Oracle Virtual Private Database policy associates the VPD function with a table, view, or synonym.
- Tutorial: Creating a Simple Oracle Virtual Private Database Policy
 This tutorial shows how to create a simple Oracle Virtual Private Database policy using the
 OE user account.



- Tutorial: Implementing a Session-Based Application Context Policy
 This tutorial demonstrates how to create an Oracle Virtual Private Database policy that uses a database session-based application context.
- Using Application Contexts to Retrieve User Information
 An application context stores user identification that can enable or prevent a user from accessing data in the database.
- How Editions Affects the Results of a Global Application Context PL/SQL Package
 Global application context packages, Oracle Virtual Private Database packages, and finegrained audit policies can be used across multiple editions.

14.2.2 Policies to Attach the Function to the Objects You Want to Protect

The Oracle Virtual Private Database policy associates the VPD function with a table, view, or synonym.

You create the policy by using the <code>DBMS_RLS</code> package. If you are not <code>SYS</code>, then you must be granted <code>EXECUTE</code> privileges to use the <code>DBMS_RLS</code> package. This package contains procedures that enable you to manage the policy and set fine-grained access control. For example, to attach the policy to a table, you use the <code>DBMS_RLS.ADD_POLICY</code> procedure. Within this setting, you set fine-grained access control, such as setting the policy to go into effect when a user issues a <code>SELECT</code> or <code>UPDATE</code> statement on the table or view.

The combination of creating the function and then applying it to a table or view is referred to as creating the Oracle Virtual Private Database policy.

Related Topics

- Configuration of Oracle Virtual Private Database Policies
 The DBMS_RLS PL/SQL package can configure Oracle Virtual Private Database (VPD) policies.
- Tutorials: Creating Oracle Virtual Private Database Policies
 These tutorials show how to create a simple and a database session-based Oracle Virtual Private policy, and how to create policy groups.

14.3 Configuration of Oracle Virtual Private Database Policies

The DBMS RLS PL/SQL package can configure Oracle Virtual Private Database (VPD) policies.

- About Oracle Virtual Private Database Policies
 The Oracle Virtual Private Database policy associates the VPD function with a database table, view, or synonym.
- Attaching a Policy to a Database Table, View, or Synonym
 The DBMS RLS PL/SQL package can attach a policy to a table, view, or synonym.
- Example: Attaching a Simple Oracle Virtual Private Database Policy to a Table

 The DBMS_RLS.ADD_POLICY procedure can attach an Oracle Virtual Private Database (VPD)

 policy to a table, view, or synomym.
- Enforcing Policies on Specific SQL Statement Types
 You can enforce Oracle Virtual Private Database policies for SELECT, INSERT, UPDATE,
 INDEX, and DELETE statements.
- Example: Specifying SQL Statement Types with DBMS_RLS.ADD_POLICY
 The DBMS_RLS.ADD_POLICY procedure statement_types parameter can specify the SELECT
 and INDEX statements for a policy.



- Control of the Display of Column Data with Policies
 - You can create policies that enforce row-level security when a security-relevant column is referenced in a query.
- Oracle Virtual Private Database Policy Groups
 An Oracle Virtual Private Database policy group is a named collection of VPD policies that can be applied to an application.
- Optimizing Performance by Using Oracle Virtual Private Database Policy Types
 You can optimize performance by using the Oracle Virtual Private Database (VPD) the
 dynamic, static, or shared policy types.

14.3.1 About Oracle Virtual Private Database Policies

The Oracle Virtual Private Database policy associates the VPD function with a database table, view, or synonym.

This function defines the actions of the Oracle Virtual Private Database WHERE clause. You must then associate this function with the database table to which the Oracle Virtual Private Database (VPD) action applies.

You can do this by configuring an Oracle Virtual Private Database policy. The policy itself is a mechanism for managing the Virtual Private Database function. The policy also enables you to add fine-grained access control, such as specifying the types of SQL statements or particular table columns the policy affects. When a user tries to access the data in this database object, the policy goes into effect automatically.

Table 14-1 lists the procedures in the DBMS RLS package.

Table 14-1 DBMS RLS Procedures

Procedure	Description
For Handling Individual Policies	-
DBMS_RLS.ADD_POLICY	Adds a policy to a table, view, or synonym
DBMS_RLS.ENABLE_POLICY	Enables (or disables) a policy you previously added to a table, view, or synonym
DBMS_RLS.ALTER_POLICY	Alters an existing policy to associate or disassociate attributes with the policy
DBMS_RLS.REFRESH_POLICY	Invalidates cursors associated with nonstatic policies
DBMS_RLS.DROP_POLICY	To drop a policy from a table, view, or synonym
For Handling Grouped Policies	-
DBMS_RLS.CREATE_POLICY_GROUP	Creates a policy group
DBMS_RLS.ALTER_GROUPED_POLICY	Alters a policy group
DBMS_RLS.DELETE_POLICY_GROUP	Drops a policy group
DBMS_RLS.ADD_GROUPED_POLICY	Adds a policy to the specified policy group
DBMS_RLS.ENABLE_GROUPED_POLICY	Enables a policy within a group
DBMS_RLS.REFRESH_GROUPED_POLICY	Parses again the SQL statements associated with a refreshed policy
DBMS_RLS.DISABLE_GROUPED_POLICY	Disables a policy within a group



Table 14-1 (C	ont.) DBMS	_RLS Procedures
---------------	------------	-----------------

Procedure	Description
DBMS_RLS.DROP_GROUPED_POLICY	Drops a policy that is a member of the specified group
For Handling Application Contexts	-
DBMS_RLS.ADD_POLICY_CONTEXT	Adds the context for the active application
DBMS_RLS.DROP_POLICY_CONTEXT	Drops the context for the application

Related Topics

- Components of an Oracle Virtual Private Database Policy
 A VPD policy uses a function to generate the dynamic WHERE clause, and a policy to attach
 the function to objects to protect.
- Using Application Contexts to Retrieve User Information
 An application context stores user identification that can enable or prevent a user from accessing data in the database.

14.3.2 Attaching a Policy to a Database Table, View, or Synonym

The DBMS RLS PL/SQL package can attach a policy to a table, view, or synonym.

To attach a policy to a database table, view, or synonym, use the DBMS_RLS.ADD_POLICY procedure.

You must specify the table, view, or synonym to which you are adding a policy, and a name for the policy. You can also specify other information, such as the types of statements the policy controls (SELECT, INSERT, UPDATE, DELETE, CREATE INDEX, or ALTER INDEX).

Follow these guidelines:

- If a view has been created as an extended data-linked object, then Oracle recommends that you apply the same VPD policy on this type of view as you would on the underlying objects of the view.
- Determine if the base object to which you want to add the VPD policy has dependent objects. If it does have dependent objects, then these objects will become invalid when the VPD policy is added to the base object, and these objects will be recompiled automatically when they are used.
 - Alternatively, you can proactively recompile them yourself by using an ALTER ... COMPILE statement. Be aware that invalidating dependent objects (by adding a VPD policy on their base object) and causing them to need to be recompiled can decrease performance in the overall system. Oracle recommends that you only add a VPD policy to an object that has dependent objects during off-peak hours or during a scheduled downtime.
- Be aware that the maximum number of policies that can be created for a single object is 255.

14.3.3 Example: Attaching a Simple Oracle Virtual Private Database Policy to a Table

The DBMS_RLS.ADD_POLICY procedure can attach an Oracle Virtual Private Database (VPD) policy to a table, view, or synomym.

Example 14-1 shows how to use <code>DBMS_RLS.ADD_POLICY</code> to attach an Oracle Virtual Private Database policy called <code>secure_update</code> to the <code>HR.EMPLOYEES</code> table. The function attached to the policy is <code>check updates</code>.

Example 14-1 Attaching a Simple Oracle Virtual Private Database Policy to a Table

```
BEGIN

DBMS_RLS.ADD_POLICY(
  object_schema => 'hr',
  object_name => 'employees',
  policy_name => 'secure_update',
  policy_function => 'check_updates',
```

If the function was created inside a package, include the package name. For example:

```
policy_function => 'pkg.check_updates',
...
```

Although you can define a policy against a table, you cannot select that table from within the policy that was defined against the table.

14.3.4 Enforcing Policies on Specific SQL Statement Types

You can enforce Oracle Virtual Private Database policies for SELECT, INSERT, UPDATE, INDEX, and DELETE statements.

• To specify a SQL statement type for the policy, use the statement_types parameter in the DBMS_RLS.ADD_POLICY procedure. If you want to specify more than one, separate each with a comma. Enclose the list in a pair of single quotation marks.

If you do not specify a statement type, then by default, Oracle Database specifies <code>SELECT</code>, <code>INSERT</code>, <code>UPDATE</code>, and <code>DELETE</code>, but not <code>INDEX</code>. You can enter any combination of these statement types.

When you specify the statement types parameter, be aware of the following functionality:

- The application code affected by the Virtual Private Database policy can include the MERGE INTO statement. However, in the Virtual Private Database policy, you must ensure that the statement_types parameter includes all three of the INSERT, UPDATE, and DELETE statements for the policy to succeed. Alternatively, you can omit the statement types parameter.
- Be aware that a user who has privileges to maintain an index can see all the row data, even if the user does not have full table access under a regular query such as SELECT. For example, a user can create a function-based index that contains a user-defined function with column values as its arguments. During index creation, Oracle Database passes column values of every row into the user function, making the row data available to the user who creates the index. You can enforce Oracle Virtual Private Database policies on index maintenance operations by specifying INDEX with the statement_types parameter.

14.3.5 Example: Specifying SQL Statement Types with DBMS RLS.ADD POLICY

The DBMS_RLS.ADD_POLICY procedure statement_types parameter can specify the SELECT and INDEX statements for a policy.

Example 14-2 shows an how this works.

Example 14-2 Specifying SQL Statement Types with DBMS_RLS.ADD_POLICY

```
BEGIN

DBMS_RLS.ADD_POLICY(
  object_schema => 'hr',
  object_name => 'employees',
  policy_name => 'secure_update',
  policy_function => 'check_updates',
   statement_types => 'SELECT,INDEX');
END;
//
```

14.3.6 Control of the Display of Column Data with Policies

You can create policies that enforce row-level security when a security-relevant column is referenced in a query.

- Policies for Column-Level Oracle Virtual Private Database
 Column-level policies enforce row-level security when a query references a security-relevant column.
- Example: Creating a Column-Level Oracle Virtual Private Database Policy
 The CREATE FUNCTION statement and the DBMS_RLS.ADD_POLICY procedure can configure a column-level Oracle Virtual Private Database policy.
- Display of Only the Column Rows Relevant to the Query
 Be default, column-level Oracle Virtual Private Database restricts the number of rows a
 query returns that references columns containing sensitive information.
- Column Masking to Display Sensitive Columns as NULL Values
 If a query references a sensitive column, then by default column-level Oracle Virtual
 Private Database restricts the number of rows returned.
- Example: Adding Column Masking to an Oracle Virtual Private Database Policy
 The DBMS_RLS.ADD_POLICY procedure can configure column-level Oracle Virtual Private
 Database column masking.

14.3.6.1 Policies for Column-Level Oracle Virtual Private Database

Column-level policies enforce row-level security when a query references a security-relevant column.

You can apply a column-level Oracle Virtual Private Database policy to tables and views, but not to synonyms. To apply the policy to a column, specify the security-relevant column by using the SEC_RELEVANT_COLS parameter of the DBMS_RLS.ADD_POLICY procedure. This parameter applies the security policy whenever the column is referenced, explicitly or implicitly, in a query.

For example, users who are not in a Human Resources department typically are allowed to view only their own Social Security numbers. A sales clerk initiates the following query:

```
SELECT fname, lname, ssn FROM emp;
```

The function implementing the security policy returns the predicate ssn='my_ssn'. Oracle Database rewrites the query and executes the following:

```
SELECT fname, lname, ssn FROM emp
WHERE ssn = 'my ssn';
```



14.3.6.2 Example: Creating a Column-Level Oracle Virtual Private Database Policy

The CREATE FUNCTION statement and the DBMS_RLS.ADD_POLICY procedure can configure a column-level Oracle Virtual Private Database policy.

Example 14-3 shows an Oracle Virtual Private Database policy in which sales department users cannot see the salaries of people outside the department (department number 30) of the sales department users. The relevant columns for this policy are sal and comm. First, the Oracle Virtual Private Database policy function is created, and then it is added by using the DBMS RLS PL/SQL package.

Example 14-3 Creating a Column-Level Oracle Virtual Private Database Policy

```
CREATE OR REPLACE FUNCTION hide_sal_comm (
v_schema IN VARCHAR2,
v_objname IN VARCHAR2)

RETURN VARCHAR2 AS
con VARCHAR2 (200);

BEGIN
con := 'deptno=30';
RETURN (con);
END hide_sal_comm;
```

Then you configure the policy with the DBMS RLS.ADD POLICY procedure as follows:

```
BEGIN

DBMS_RLS.ADD_POLICY (
  object_schema => 'scott',
  object_name => 'emp',
  policy_name => 'hide_sal_policy',
  policy_function => 'hide_sal_comm',
  sec_relevant_cols => 'sal,comm');

END;
```

14.3.6.3 Display of Only the Column Rows Relevant to the Query

Be default, column-level Oracle Virtual Private Database restricts the number of rows a query returns that references columns containing sensitive information.

You specify these security-relevant columns by using the SEC_RELEVANT_COLUMNS parameter of the DBMS_RLS.ADD_POLICY procedure, as shown in Example 14-3.

For example, consider sales department users with the SELECT privilege on the emp table, which is protected with the column-level Oracle Virtual Private Database policy created in Example 14-3. The user (for example, user SCOTT) runs the following query:

```
SELECT ENAME, d.dname, JOB, SAL, COMM
FROM emp e, dept d
WHERE d.deptno = e.deptno;
```

The database returns the following rows:

ENAME	DNAME	JOB	SAL	COMM
ALLEN	SALES	SALESMAN	1600	300
WARD	SALES	SALESMAN	1250	500
MARTIN	SALES	SALESMAN	1250	1400
BLAKE	SALES	MANAGER	2850	



TURNER	SALES	SALESMAN	1500	0
JAMES	SALES	CLERK	950	

6 rows selected.

The only rows that are displayed are those that the user has privileges to access all columns in the row.

14.3.6.4 Column Masking to Display Sensitive Columns as NULL Values

If a query references a sensitive column, then by default column-level Oracle Virtual Private Database restricts the number of rows returned.

With column-masking behavior, all rows display, even those that reference sensitive columns. However, the sensitive columns display as <code>NULL</code> values. To enable column-masking, set the <code>SEC_RELEVANT_COLS_opt</code> parameter of the <code>DBMS_RLS.ADD_POLICY</code> procedure.

For example, consider the results of the sales clerk query, described in the previous example. If column-masking is used, then instead of seeing only the row containing the details and Social Security number of the sales clerk, the clerk would see all rows from the \mathtt{emp} table, but the \mathtt{ssn} column values would be returned as \mathtt{NULL} . Note that this behavior is fundamentally different from all other types of Oracle Virtual Private Database policies, which return only a subset of rows.

In contrast to the default action of column-level Oracle Virtual Private Database, column-masking displays all rows, but returns sensitive column values as <code>NULL</code>. To include column-masking in your policy, set the <code>SEC_RELEVANT_COLS_OPT</code> parameter of the <code>DBMS_RLS.ADD_POLICY</code> procedure to <code>DBMS_RLS.ALL_ROWS</code>.

The following considerations apply to column masking:

- Column-masking applies only to SELECT statements.
- Column-masking conditions generated by the policy function must be simple Boolean expressions, unlike regular Oracle Virtual Private Database predicates.
- For applications that perform calculations, or do not expect NULL values, use standard column-level Oracle Virtual Private Database, specifying SEC_RELEVANT_COLS rather than the SEC RELEVANT COLS OPT column-masking option.
- Do not include columns of the object data type (including the XMLtype) in the sec_relevant_cols setting. This column type is not supported for the sec_relevant_cols setting.
- Column-masking used with UPDATE AS SELECT updates only the columns that users are allowed to see.
- For some queries, column-masking may prevent some rows from displaying. For example:

```
SELECT * FROM emp
WHERE sal = 10;
```

Because the column-masking option was set, this query may not return rows if the salary column returns a NULL value.



14.3.6.5 Example: Adding Column Masking to an Oracle Virtual Private Database Policy

The DBMS_RLS.ADD_POLICY procedure can configure column-level Oracle Virtual Private Database column masking.

Example 14-4 shows column-level Oracle Virtual Private Database column masking. It uses the same VPD policy as Example: Creating a Column-Level Oracle Virtual Private Database Policy, but with sec relevant cols opt specified as DBMS RLS.ALL ROWS.

Example 14-4 Adding Column Masking to an Oracle Virtual Private Database Policy

Assume that a sales department user with SELECT privilege on the emp table (such as user SCOTT) runs the following query:

```
SELECT ENAME, d.dname, job, sal, comm
FROM emp e, dept d
WHERE d.deptno = e.deptno;
```

The database returns all rows specified in the query, but with certain values masked because of the Oracle Virtual Private Database policy:

ENAME	DNAME	JOB	SAL	COMM
CLARK	ACCOUNTING	MANAGER		
KING	ACCOUNTING	PRESIDENT		
MILLER	ACCOUNTING	CLERK		
JONES	RESEARCH	MANAGER		
FORD	RESEARCH	ANALYST		
ADAMS	RESEARCH	CLERK		
SMITH	RESEARCH	CLERK		
SCOTT	RESEARCH	ANALYST		
WARD	SALES	SALESMAN	1250	500
TURNER	SALES	SALESMAN	1500	0
ALLEN	SALES	SALESMAN	1600	300
JAMES	SALES	CLERK	950	
BLAKE	SALES	MANAGER	2850	
MARTIN	SALES	SALESMAN	1250	1400

14 rows selected.

The column-masking returned all rows requested by the sales user query, but made the sal and comm columns NULL for employees outside the sales department.

14.3.7 Oracle Virtual Private Database Policy Groups

An Oracle Virtual Private Database policy group is a named collection of VPD policies that can be applied to an application.

- About Oracle Virtual Private Database Policy Groups
 You can group multiple security policies together, and apply them to an application.
- Creation of a New Oracle Virtual Private Database Policy Group
 The DBMS RLS.ADD GROUPED POLICY procedure adds a VPD policy to a VPD policy group.
- Default Policy Group with the SYS_DEFAULT Policy Group
 Within a group of security policies, you can designate one security policy to be the default
 security policy.
- Multiple Policies for Each Table, View, or Synonym
 You can establish several policies for the same table, view, or synonym.
- Validation of the Application Used to Connect to the Database
 The package implementing the driving context must correctly validate the application that is being used to connect to the database.

14.3.7.1 About Oracle Virtual Private Database Policy Groups

You can group multiple security policies together, and apply them to an application.

A policy group is a set of security policies that belong to an application. You can designate an application context (known as a *driving context* or *policy context*) to indicate the policy group in effect. Then, when a user accesses the table, view, or synonym column, Oracle Database looks up the driving context to determine the policy group in effect. It enforces all the associated policies that belong to the policy group.

Policy groups are useful for situations where multiple applications with multiple security policies share the same table, view, or synonym. This enables you to identify those policies that should be in effect when the table, view, or synonym is accessed.

For example, in a hosting environment, Company A can host the BENEFIT table for Company B and Company C. The table is accessed by two different applications, Human Resources and Finance, with two different security policies. The Human Resources application authorizes users based on ranking in the company, and the Finance application authorizes users based on department. Integrating these two policies into the BENEFIT table requires joint development of policies between the two companies, which is not a feasible option. By defining an application context to drive the enforcement of a particular set of policies to the base objects, each application can implement a private set of security policies.

To do this, you organize security policies into groups. By referring to the application context, Oracle Database determines which group of policies should be in effect at run time. The server enforces all the policies that belong to that policy group.

14.3.7.2 Creation of a New Oracle Virtual Private Database Policy Group

The DBMS RLS.ADD GROUPED POLICY procedure adds a VPD policy to a VPD policy group.

To specify which policies will be effective, you can add a driving context using the <code>DBMS_RLS.ADD_POLICY_CONTEXT</code> procedure. If the driving context returns an unknown policy group, then an error is returned.

If the driving context is not defined, then Oracle Database runs all policies. Likewise, if the driving context is \mathtt{NULL} , then policies from all policy groups are enforced. An application accessing the data cannot bypass the security setup module (which sets up application context) to avoid any applicable policies.

You can apply multiple driving contexts to the same table, view, or synonym, and each of them will be processed individually. This enables you to configure multiple active sets of policies to be enforced.

Consider, for example, a hosting company that hosts Benefits and Financial applications, which share some database objects. Both applications are striped for hosting using a SUBSCRIBER policy in the SYS_DEFAULT policy group. Data access is partitioned first by subscriber ID, then by whether the user is accessing the Benefits or Financial applications (determined by a driving context). Suppose that Company A, which uses the hosting services, wants to apply a custom policy that relates only to its own data access. You could add an additional driving context (such as COMPANY A SPECIAL) to ensure that the additional, special policy group is applied for data access for Company A only. You would not apply this under the SUBSCRIBER policy, because the policy relates only to Company A, and it is more efficient to segregate the basic hosting policy from other policies.

14.3.7.3 Default Policy Group with the SYS DEFAULT Policy Group

Within a group of security policies, you can designate one security policy to be the default security policy.

This is useful in situations where you partition security policies by application, so that they will be always be in effect. Default security policies enable developers to base security enforcement under all conditions, while partitioning security policies by application (using security groups) enables layering of additional, application-specific security on top of default security policies. To implement default security policies, you add the policy to the SYS_DEFAULT policy group.

Policies defined in this group for a particular table, view, or synonym are run with the policy group specified by the driving context. As described earlier, a driving context is an application context that indicates the policy group in effect. The SYS_DEFAULT policy group may or may not contain policies. You cannot to drop the SYS_DEFAULT policy group. If you do, then Oracle Database displays an error.

If, to the SYS_DEFAULT policy group, you add policies associated with two or more objects, then each object will have a separate SYS_DEFAULT policy group associated with it. For example, the emp table in the scott schema has one SYS_DEFAULT policy group, and the dept table in the scott schema has a different SYS_DEFAULT policy group associated with it. Think of them as being organized in the tree structure as follows:

```
SYS_DEFAULT
  - policy1 (scott/emp)
  - policy3 (scott/emp)
SYS_DEFAULT
  - policy2 (scott/dept)
```

You can create policy groups with identical names. When you select a particular policy group, its associated schema and object name are displayed in the property sheet on the right side of the screen.

14.3.7.4 Multiple Policies for Each Table, View, or Synonym

You can establish several policies for the same table, view, or synonym.

Suppose, for example, you have a base application for Order Entry, and each division of your company has its own rules for data access. You can add a division-specific policy function to a table without having to rewrite the policy function of the base application.

All policies applied to a table are enforced with AND syntax. If you have three policies applied to the CUSTOMERS table, then each policy is applied to the table. You can use policy groups and an application context to partition fine-grained access control enforcement so that different policies apply, depending upon which application is accessing data. This eliminates the requirement for

development groups to collaborate on policies, and simplifies application development. You can also have a default policy group that is always applicable (for example, to enforce data separated by subscriber in a hosting environment).

14.3.7.5 Validation of the Application Used to Connect to the Database

The package implementing the driving context must correctly validate the application that is being used to connect to the database.

Although Oracle Database checks the call stack to ensure that the package implementing the driving context sets context attributes, inadequate validation can still occur within the package. For example, in applications where database users or enterprise users are known to the database, the user needs the EXECUTE privilege on the package that sets the driving context. Consider a user who knows that the BENEFITS application enables more liberal access than the HR application. The setctx procedure (which sets the correct policy group within the driving context) does not perform any validation to determine which application is actually connecting. That is, the procedure does not check either the IP address of the incoming connection (for a three-tier system) or the proxy user attribute of the user session.

This user could pass to the driving context package an argument setting the context to the more liberal BENEFITS policy group, and then access the HR application instead. Because the setctx does no further validation of the application, this user bypasses the more restrictive HR security policy.

By contrast, if you implement proxy authentication with Oracle Virtual Private Database, then you can determine the identity of the middle tier (and the application) that is connecting to the database on behalf of a user. The correct policy will be applied for each application to mediate data access.

For example, a developer using the proxy authentication feature could determine that the application (the middle tier) connecting to the database is <code>HRAPPSERVER</code>. The package that implements the driving context can thus verify whether the <code>proxy_user</code> in the user session is <code>HRAPPSERVER</code>. If so, then it can set the driving context to use the <code>HR policy</code> group. If <code>proxy_user</code> is not <code>HRAPPSERVER</code>, then it can deny access.

In this case, the following query is executed:

```
SELECT * FROM apps.benefit;
```

Oracle Database picks up policies from the default policy group ($SYS_DEFAULT$) and active namespace HR. The query is internally rewritten as follows:

```
SELECT * FROM apps.benefit
WHERE company = SYS_CONTEXT('ID','MY_COMPANY')
AND SYS CONTEXT('ID','TITLE') = 'MANAGER';
```

14.3.8 Optimizing Performance by Using Oracle Virtual Private Database Policy Types

You can optimize performance by using the Oracle Virtual Private Database (VPD) the dynamic, static, or shared policy types.

About Oracle Virtual Private Database Policy Types
 Specifying a policy type for your policies can optimize performance each the Oracle Virtual Private Database policy runs.

- Dynamic Policy Type to Automatically Rerun Policy Functions
 The DYNAMIC policy type runs the policy function each time a user accesses the Virtual Private Database-protected database objects.
- Example: Creating a DYNAMIC Policy with DBMS_RLS.ADD_POLICY
 The DBMS_RLS.ADD_POLICY procedure can create a dynamic Oracle Virtual Private Database policy.
- Static Policy to Prevent Policy Functions from Rerunning for Each Query
 The static policy type enforces the same predicate for all users in the instance.
- Example: Creating a Static Policy with DBMS_RLS.ADD_POLICY
 The DBMS_RLS.ADD_POLICY procedure can create a static Oracle Virtual Private Database
 (VPD) policy.
- Example: Shared Static Policy to Share a Policy with Multiple Objects
 The DBMS_RLS.ADD_POLICY procedure can create a shared static Oracle Virtual Private
 Database policy to share the policy with multiple objects.
- When to Use Static and Shared Static Policies
 Static policies are ideal when every query requires the same predicate and fast performance is essential, such as hosting environments.
- Context-Sensitive Policy for Application Context Attributes That Change
 Context-sensitive policies are useful when different predicates must be applied depending on which predicate executes the query.
- Example: Creating a Context-Sensitive Policy with DBMS_RLS.ADD_POLICY
 The DBMS_RLS.ADD_POLICY procedure can create an Oracle Virtual Private Database
 context-sensitive policy.
- Example: Refreshing Cached Statements for a VPD Context-Sensitive Policy
 The DBMS_RLS.REFRESH_POLICY statement can refresh cached statements for Oracle
 Virtual Private Database context-sensitive policies.
- Example: Altering an Existing Context-Sensitive Policy
 The DBMS_RLS.ALTER_POLICY procedure can modify an Oracle Virtual Private Database policy.
- Example: Using a Shared Context Sensitive Policy to Share a Policy with Multiple Objects The DBMS_RLS.ADD_POLICY procedure can create a shared context-sensitive Oracle Virtual Private Database to share a policy that has multiple objects.
- When to Use Context-Sensitive and Shared Context-Sensitive Policies
 Use context-sensitive policies when a predicate does not need to change for a user
 session, but the policy must enforce multiple predicates for different users or groups.
- Summary of the Five Oracle Virtual Private Database Policy Types
 Oracle Virtual Private Database provides five policy types, based on user needs such as hosting environments.

14.3.8.1 About Oracle Virtual Private Database Policy Types

Specifying a policy type for your policies can optimize performance each the Oracle Virtual Private Database policy runs.

Policy types control how Oracle Database caches Oracle Virtual Private Database policy predicates. Consider setting a policy type for your policies, because the execution of policy functions can use a significant amount of system resources. Minimizing the number of times that a policy function can run optimizes database performance.



You can choose from five policy types: DYNAMIC, STATIC, SHARED_STATIC, CONTEXT_SENSITIVE, and SHARED_CONTEXT_SENSITIVE. These enable you to precisely specify how often a policy predicate should change. To specify the policy type, set the policy_type parameter of the DBMS_RLS.ADD POLICY procedure.

14.3.8.2 Dynamic Policy Type to Automatically Rerun Policy Functions

The DYNAMIC policy type runs the policy function each time a user accesses the Virtual Private Database-protected database objects.

If you do not specify a policy type in the <code>DBMS_RLS.ADD_POLICY</code> procedure, then, by default, your policy will be dynamic. You can specifically configure a policy to be dynamic by setting the <code>policy_type</code> parameter of the <code>DBMS_RLS.ADD_POLICY</code> procedure to <code>DYNAMIC</code>.

This policy type does not optimize database performance as the static and context sensitive policy types do. However, Oracle recommends that before you set policies as either static or context-sensitive, you should first test them as <code>DYNAMIC</code> policy types, which run every time. Testing policy functions as <code>DYNAMIC</code> policies first enables you to observe how the policy function affects each query, because nothing is cached. This ensures that the functions work properly before you enable them as static or context-sensitive policy types to optimize performance.

You can use the <code>DBMS_UTILITY.GET_TIME</code> function to measure the start and end times for a statement to execute. For example:

Related Topics

Auditing Functions, Procedures, Packages, and Triggers
 You can audit functions, procedures, PL/SQL packages, and triggers.

14.3.8.3 Example: Creating a DYNAMIC Policy with DBMS_RLS.ADD_POLICY

The DBMS_RLS.ADD_POLICY procedure can create a dynamic Oracle Virtual Private Database policy.

Example 14-5 shows how to create the DYNAMIC policy type.

Example 14-5 Creating a DYNAMIC Policy with DBMS_RLS.ADD_POLICY

```
BEGIN

DBMS_RLS.ADD_POLICY(
  object_schema => 'hr',
  object_name => 'employees',
  policy_name => 'secure_update',
  policy_function => 'hide_fin',
  policy_type => dbms_rls.DYNAMIC);
END;
//
```

14.3.8.4 Static Policy to Prevent Policy Functions from Rerunning for Each Query

The static policy type enforces the same predicate for all users in the instance.

Oracle Database stores static policy predicates in SGA, so policy functions do not rerun for each query. This results in faster performance.

You can enable static policies by setting the <code>policy_type</code> parameter of the <code>DBMS_RLS.ADD_POLICY</code> procedure to either <code>STATIC</code> or <code>SHARED_STATIC</code>, depending on whether or not you want the policy to be shared across multiple objects.

Each execution of the same cursor could produce a different row set for the same predicate, because the predicate may filter the data differently based on attributes such as SYS_CONTEXT or SYSDATE.

For example, suppose you enable a policy as either a STATIC or SHARED_STATIC policy type, which appends the following predicate to all queries made against policy protected database objects:

```
WHERE dept = SYS CONTEXT ('hr app', 'deptno')
```

Although the predicate does not change for each query, it applies to the query based on session attributes of the <code>SYS_CONTEXT</code>. In the case of the preceding example, the predicate returns only those rows where the department number matches the <code>deptno</code> attribute of the <code>SYS_CONTEXT</code>, which is the department number of the user who is querying the policy-protected database object.



When using shared static policies, ensure that the policy predicate does not contain attributes that are specific to a particular database object, such as a column name.

Related Topics

Auditing Functions, Procedures, Packages, and Triggers
 You can audit functions, procedures, PL/SQL packages, and triggers.

14.3.8.5 Example: Creating a Static Policy with DBMS RLS.ADD POLICY

The DBMS_RLS.ADD_POLICY procedure can create a static Oracle Virtual Private Database (VPD) policy.

Example 14-6 shows how to create the STATIC policy type.

Example 14-6 Creating a Static Policy with DBMS_RLS.ADD_POLICY

```
BEGIN

DBMS_RLS.ADD_POLICY(
  object_schema => 'hr',
  object_name => 'employees',
  policy_name => 'secure_update',
  policy_function => 'hide_fin',
  policy_type => DBMS_RLS.STATIC);
END;
//
```

14.3.8.6 Example: Shared Static Policy to Share a Policy with Multiple Objects

The DBMS_RLS.ADD_POLICY procedure can create a shared static Oracle Virtual Private Database policy to share the policy with multiple objects.

If, for example, you wanted to apply the static policy that was created earlier to a second table in the HR schema that may contain financial data that you want to hide, you could use the SHARED STATIC setting for both tables.

Example 14-7 shows how to set the SHARED_STATIC policy type for two tables that share the same policy.

Example 14-7 Creating a Shared Static Policy to Share a Policy with Multiple Objects

BEGIN DBMS_RLS.ADD_POLICY(object_schema => 'hr', object_name => 'employees', policy_name => 'secure_update', policy_function => 'hide_fin', policy_type => dbms_rls.SHARED_STATIC); END; /--2. Create a policy for the second table, fin_data: BEGIN DBMS_RLS.ADD_POLICY(object_schema => 'hr', object_name => 'fin_data', policy_name => 'secure_update', policy_function => 'hide_fin', policy_type => dbms_rls.SHARED_STATIC);

-- 1. Create a policy for the first table, employees:

Related Topics

END;

 Example: Creating a Static Policy with DBMS_RLS.ADD_POLICY
 The DBMS_RLS.ADD_POLICY procedure can create a static Oracle Virtual Private Database (VPD) policy.

14.3.8.7 When to Use Static and Shared Static Policies

Static policies are ideal when every query requires the same predicate and fast performance is essential, such as hosting environments.

For these situations when the policy function appends the same predicate to every query, rerunning the policy function each time adds unnecessary overhead to the system. For example, consider a data warehouse that contains market research data for customer

organizations that are competitors. The warehouse must enforce the policy that each organization can see only their own market research, which is expressed by the following predicate:

```
WHERE subscriber_id = SYS_CONTEXT('customer', 'cust_num')
```

Using SYS_CONTEXT for the application context enables the database to dynamically change the rows that are returned. You do not need to rerun the function, so the predicate can be cached in the SGA, thus conserving system resources and improving performance.

14.3.8.8 Context-Sensitive Policy for Application Context Attributes That Change

Context-sensitive policies are useful when different predicates must be applied depending on which predicate executes the query.

For example, consider the case where managers should have the predicate <code>WHERE group</code> set to <code>managers</code>, and employees should have the predicate <code>WHERE empno_ctx</code> set to <code>emp_id</code>. A context-sensitive policy will enable you to present only the information that the managers must see when the managers log in, and only the information that the employees must see when they log in. The policy uses application contexts to determine which predicate to use.

In contrast to static policies, context-sensitive policies do not always cache the predicate. With context-sensitive policies, the database assumes that the predicate will change after statement parse time. But if there is no change in the local application context, then Oracle Database does not rerun the policy function within the user session. If there is a change in any attribute of any application context during the user session, then by default the database re-executes the policy function to ensure that it captures all changes to the predicate since the initial parsing. This results in unnecessary re-executions of the policy function if none of the associated attributes have changed. You can restrict the evaluation to a specific application context by including both the namespace and attribute parameters.

If you plan to use the namespace and attribute parameters in your policy, then follow these quidelines:

- Ensure that you specify both namespace and attribute parameters, not just one.
- Ensure that your policy has the policy_type argument set to DBMS_RLS.CONTEXT_SENSITIVE or SHARED_CONTEXT_SENSITIVE. You cannot use the namespace and attribute parameters in static or dynamic policies.

If there are no attributes associated with the Virtual Private Database policy function, then Oracle Database evaluates the context-sensitive function for any application context changes.

Shared context-sensitive policies operate in the same way as regular context-sensitive policies, except they can be shared across multiple database objects. For this policy type, all objects can share the policy function from the UGA, where the predicate is cached until the local session context changes.

Related Topics

- Example: Using a Shared Context Sensitive Policy to Share a Policy with Multiple Objects The DBMS_RLS.ADD_POLICY procedure can create a shared context-sensitive Oracle Virtual Private Database to share a policy that has multiple objects.
- Tutorial: Implementing a Session-Based Application Context Policy
 This tutorial demonstrates how to create an Oracle Virtual Private Database policy that uses a database session-based application context.
- Tutorial: Implementing an Oracle Virtual Private Database Policy Group
 This tutorial demonstrates how to create an Oracle Virtual Private Database policy group.

14.3.8.9 Example: Creating a Context-Sensitive Policy with DBMS RLS.ADD POLICY

The DBMS_RLS.ADD_POLICY procedure can create an Oracle Virtual Private Database context-sensitive policy.

Example 14-8shows how to create a CONTEXT_SENSITIVE policy in which the policy is evaluated only for changes to the empno ctx namespace and emp id attribute.

Example 14-8 Creating a Context-Sensitive Policy with DBMS_RLS.ADD_POLICY

```
BEGIN

DBMS_RLS.ADD_POLICY(
  object_schema => 'hr',
  object_name => 'employees',
  policy_name => 'secure_update',
  policy_function => 'hide_fin',
  policy_type => dbms_rls.CONTEXT_SENSITIVE,
  namespace => 'empno_ctx',
  attribute => 'emp_id');
END;
//
```

14.3.8.10 Example: Refreshing Cached Statements for a VPD Context-Sensitive Policy

The DBMS_RLS.REFRESH_POLICY statement can refresh cached statements for Oracle Virtual Private Database context-sensitive policies.

Example 14-9 shows you can manually refresh all the cached statements that are associated with a Virtual Private Database context-sensitive policy by running the DBMS RLS.REFRESH POLICY procedure.

Example 14-9 Refreshing Cached Statements for a VPD Context-Sensitive Policy

```
BEGIN

DBMS_RLS.REFRESH_POLICY(
  object_schema => 'hr',
  object_name => 'employees',
  policy_name => 'secure_update');
END;
//
```

14.3.8.11 Example: Altering an Existing Context-Sensitive Policy

The DBMS RLS.ALTER POLICY procedure can modify an Oracle Virtual Private Database policy.

Example 14-10 shows how you can use the <code>DBMS_RLS.ALTER_POLICY</code> statement to alter an existing context-sensitive policy so that the <code>order_update_pol</code> policy function is executed only if the relevant context attributes change.

Example 14-10 Altering an Existing Context-Sensitive Policy

```
BEGIN

DBMS_RLS.ALTER_POLICY(
  object_schema => 'oe',
  object_name => 'orders',
  policy_name => 'order_update_pol',
```

14.3.8.12 Example: Using a Shared Context Sensitive Policy to Share a Policy with Multiple Objects

The DBMS_RLS.ADD_POLICY procedure can create a shared context-sensitive Oracle Virtual Private Database to share a policy that has multiple objects.

Example 14-11 shows how to create two shared context sensitive policies that share a policy with multiple tables, and how to restrict the evaluation only for changes to the <code>empno_ctx</code> namespace and <code>emp id</code> attribute.

Example 14-11 Shared Context-Sensitive Policy with DBMS_RLS.ADD_POLICY

```
-- 1. Create a policy for the first table, employees:
BEGIN
 DBMS RLS.ADD POLICY(
  object schema => 'hr',
  object_name => 'employees',
policy_name => 'secure_update',
  policy function => 'hide fin',
  policy_type => dbms_rls.SHARED_CONTEXT_SENSITIVE,
namespace => 'empno_ctx',
attribute => 'emp_id');
END;
--2. Create a policy for the second table, fin data:
 DBMS RLS.ADD POLICY(
  object_schema => 'hr',
  object_name => 'fin_data',
policy_name => 'secure_update',
  policy function => 'hide fin',
  policy_type => dbms_rls.SHARED_CONTEXT_SENSITIVE,
  namespace => 'empno_ctx',
attribute => 'emp_id');
END;
```

Note the following:

- When using shared context-sensitive policies, ensure that the policy predicate does not contain attributes that are specific to a particular database object, such as a column name.
- To manually refresh all the cached statements that are associated with a Virtual Private
 Database shared context-sensitive policy, run the DBMS_RLS.REFRESH_GROUPED_POLICY
 procedure.

14.3.8.13 When to Use Context-Sensitive and Shared Context-Sensitive Policies

Use context-sensitive policies when a predicate does not need to change for a user session, but the policy must enforce multiple predicates for different users or groups.

For example, consider a sales_history table with a single policy. This policy states that analysts can see only their own products and regional employees can see only their own region. In this case, the database must rerun the policy function each time the type of user

changes. The performance gain is realized when a user can log in and issue several DML statements against the protected object without causing the server to rerun the policy function.



For session pooling where multiple clients share a database session, the middle tier must reset the context during client switches.

14.3.8.14 Summary of the Five Oracle Virtual Private Database Policy Types

Oracle Virtual Private Database provides five policy types, based on user needs such as hosting environments.

Table 14-2 summarizes the types of policy types available.

Table 14-2 DBMS_RLS.ADD_POLICY Policy Types

Policy Types	When the Policy Function Executes	Usage Example	Shared Across Multiple Objects ?
DYNAMIC	Policy function re-executes every time a policy-protected database object is accessed.	Applications where policy predicates must be generated for each query, such as time-dependent policies where users are denied access to database objects at certain times during the day	No
STATIC	Once, then the predicate is cached in the $\ensuremath{SGA^1}$	View replacement	No
SHARED_STA	Same as STATIC	Hosting environments, such as data warehouses where the same predicate must be applied to multiple database objects	Yes
CONTEXT_SE NSITIVE	 At statement parse time At statement execution time when the local application context changed since the last use of the cursor 	Three-tier, session pooling applications where policies enforce two or more predicates for different users or groups	No
SHARED_CON TEXT_SENSI TIVE	First time the object is reference in a database session. Predicates are cached in the private session memory UGA so policy functions can be shared among objects.	Same as CONTEXT_SENSITIVE, but multiple objects can share the policy function from the session UGA	Yes

Each execution of the same cursor could produce a different row set for the same predicate because the predicate may filter the data differently based on attributes such as SYS CONTEXT or SYSDATE.

14.4 Tutorials: Creating Oracle Virtual Private Database Policies

These tutorials show how to create a simple and a database session-based Oracle Virtual Private policy, and how to create policy groups.

Tutorial: Creating a Simple Oracle Virtual Private Database Policy
 This tutorial shows how to create a simple Oracle Virtual Private Database policy using the
 OE user account.

- Tutorial: Implementing a Session-Based Application Context Policy
 This tutorial demonstrates how to create an Oracle Virtual Private Database policy that uses a database session-based application context.
- Tutorial: Implementing an Oracle Virtual Private Database Policy Group
 This tutorial demonstrates how to create an Oracle Virtual Private Database policy group.

14.4.1 Tutorial: Creating a Simple Oracle Virtual Private Database Policy

This tutorial shows how to create a simple Oracle Virtual Private Database policy using the $\odot E$ user account.

About This Tutorial

This tutorial shows how to create a VPD policy that limits access to orders created by Sales Representative 159 in the OE.ORDERS table.

- Step 1: Ensure That the OE User Account Is Active First, you must ensure that OE user account is active.
- Step 2: Create a Policy Function
 Next, you are ready to create a policy function.
- Step 3: Create the Oracle Virtual Private Database Policy
 After you create the policy function, you are ready to associate it with a VPD policy.
- Step 4: Test the Policy
 After you create the Oracle Virtual Private Database policy, it goes into effect immediately.
- Step 5: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

14.4.1.1 About This Tutorial

This tutorial shows how to create a VPD policy that limits access to orders created by Sales Representative 159 in the OE.ORDERS table.

In essence, the policy translates the following statement:

```
SELECT * FROM OE.ORDERS;
```

To the following statement:

```
SELECT * FROM OE.ORDERS WHERE SALES_REP_ID = 159;
```



If you are using a multitenant environment, then this tutorial applies to the current PDB only.

14.4.1.2 Step 1: Ensure That the OE User Account Is Active

First, you must ensure that OE user account is active.

1. Log on to SQL*Plus as user SYS with the SYSDBA administrative privilege.

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



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

For example:

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

To find the available PDBs, run the show pdbs command. To check the current PDB, run the show con name command.

3. Query the DBA USERS data dictionary view to find the account status of OE.

```
SELECT USERNAME, ACCOUNT STATUS FROM DBA USERS WHERE USERNAME = 'OE';
```

The status should be OPEN. If the DBA_USERS view lists user OE as locked and expired, then enter the following statement to unlock the OE account and create a new password:

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

Follow the guidelines in Minimum Requirements for Passwords to replace *password* with a password that is secure. For greater security, do not reuse the same password that was used in previous releases of Oracle Database.

14.4.1.3 Step 2: Create a Policy Function

Next, you are ready to create a policy function.

As user SYS, create the following function, which will append the WHERE SALES_REP_ID = 159 clause to any SELECT statement on the OE.ORDERS table. (You can copy and paste this text by positioning the cursor at the start of CREATE OR REPLACE in the first line.)

```
CREATE OR REPLACE FUNCTION auth_orders(
    schema_var IN VARCHAR2,
    table_var IN VARCHAR2
)

RETURN VARCHAR2
IS
    return_val VARCHAR2 (400);

BEGIN
    return_val := 'SALES_REP_ID = 159';

RETURN return_val;

END auth_orders;
```

In this example:

- schema_var and table_var create input parameters to specify to store the schema name, OE, and table name, ORDERS. First, define the parameter for the schema, and then define the parameter for the object, in this case, a table. Always create them in this order. The Virtual Private Database policy you create will need these parameters to specify the OE.ORDERS table.
- RETURN VARCHAR2 returns the string that will be used for the WHERE predicate clause. Remember that return value is always a VARCHAR2 data type.
- IS ... RETURN return_val encompasses the creation of the WHERE SALES_REP_ID = 159 predicate.

14.4.1.4 Step 3: Create the Oracle Virtual Private Database Policy

After you create the policy function, you are ready to associate it with a VPD policy.

Create the following policy by using the ADD POLICY procedure in the DBMS RLS package.

```
BEGIN

DBMS_RLS.ADD_POLICY (
   object_schema => 'oe',
   object_name => 'orders',
   policy_name => 'orders_policy',
   function_schema => 'sys',
   policy_function => 'auth_orders',
   statement_types => 'select'
   );
END;
//
```

In this example:

- object schema => 'oe' specifies the schema that you want to protect, that is, OE.
- object_name => 'orders' specifies the object within the schema to protect, that is, the ORDERS table.
- policy_name => 'orders_policy' names this policy orders_policy.
- function_schema => 'sys' specifies the schema in which the auth_orders function
 was created. In this example, auth_orders was created in the SYS schema. But
 typically, it should be created in the schema of a security administrator.
- policy_function => 'auth_orders' specifies a function to enforce the policy. Here, you specify the auth_orders function that you created in Step 2: Create a Policy Function.
- statement_types => 'select' specifies the operations to which the policy applies. In this example, the policy applies to all SELECT statements that the user may perform.

14.4.1.5 Step 4: Test the Policy

After you create the Oracle Virtual Private Database policy, it goes into effect immediately.

The next time a user, including the owner of the schema, performs a SELECT on OE.ORDERS, only the orders by Sales Representative 159 will be accessed.

1. Connect as user OE.

```
CONNECT oe -- Or, CONNECT OE@hrpdb
Enter password: password
```

2. Enter the following SELECT statement:

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

The following output should appear:

```
COUNT (*)
```



The policy is in effect for user OE: As you can see, only 7 of the 105 rows in the orders table are returned.

But users with administrative privileges still have access to all the rows in the table.

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

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

4. Enter the following SELECT statement:

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

The following output should appear:

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

14.4.1.6 Step 5: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

1. As user SYS, remove the function and policy as follows:

```
DROP FUNCTION auth_orders;
EXEC DBMS_RLS.DROP_POLICY('OE','ORDERS','ORDERS_POLICY');
```

2. If you need to lock and expire the OE account, then enter the following statement:

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

14.4.2 Tutorial: Implementing a Session-Based Application Context Policy

This tutorial demonstrates how to create an Oracle Virtual Private Database policy that uses a database session-based application context.

About This Tutorial

This tutorial shows how to use a database session-based application context to implement a policy in which customers see only their own orders.

- Step 1: Create User Accounts and Sample Tables
 First, create user accounts and the sample tables.
- Step 2: Create a Database Session-Based Application Context
 Next, you are ready to create the database session-based application context.
- Step 3: Create a PL/SQL Package to Set the Application Context
 After you create the application context, you are ready to create a package to set the context.
- Step 4: Create a Logon Trigger to Run the Application Context PL/SQL Package
 The logon trigger runs the PL/SQL package procedure so that the next time a user logs on, the application context is set.
- Step 5: Test the Logon Trigger
 The logon trigger sets the application context for the user when the trigger runs the sysadmin_vpd.orders_ctx_pkg.set_custnum procedure.
- Step 6: Create a PL/SQL Policy Function to Limit User Access to Their Orders
 The next step is to create a PL/SQL function to control the display of the user's query.

- Step 7: Create the New Security Policy
 Finally, you are ready to create the VPD security policy.
- Step 8: Test the New Policy
 Now that you have created all the components, you are ready to test the policy.
- Step 9: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

14.4.2.1 About This Tutorial

This tutorial shows how to use a database session-based application context to implement a policy in which customers see only their own orders.

If you are using a multitenant environment, then this tutorial applies to the current PDB only.

In this tutorial, you create the following layers of security:

- 1. When a user logs on, a database session-based application context checks whether the user is a customer. If a user is not a customer, the user still can log on, but this user cannot access the orders entry table you will create for this example.
- 2. If the user is a customer, he or she can log on. After the customer has logged on, an Oracle Virtual Private Database policy restricts this user to see only his or her orders.
- As a further restriction, the Oracle Virtual Private Database policy prevents users from adding, modifying, or removing orders.

14.4.2.2 Step 1: Create User Accounts and Sample Tables

First, create user accounts and the sample tables.

1. Start SQL*Plus and log on as a user who has administrative privileges.

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

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

For example:

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

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

Create the following administrative user, who will administer the Oracle Virtual Private Database policy.

The following SQL statements create this user and then grant the user the necessary privileges for completing this tutorial.

```
CREATE USER sysadmin_vpd IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION, CREATE ANY CONTEXT, CREATE PROCEDURE, CREATE TRIGGER, ADMINISTER DATABASE
TRIGGER TO sysadmin_vpd;
GRANT EXECUTE ON DBMS_SESSION TO sysadmin_vpd;
GRANT EXECUTE ON DBMS RLS TO sysadmin vpd;
```

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

4. Create the following local users:

```
CREATE USER tbrooke IDENTIFIED BY password CONTAINER = CURRENT;
CREATE USER owoods IDENTIFIED BY password CONTAINER = CURRENT;
GRANT CREATE SESSION TO tbrooke, owoods;
```

Replace password with a password that is secure.

5. Check the account status of the sample user SCOTT, who you will use for this tutorial:

```
SELECT USERNAME, ACCOUNT STATUS FROM DBA USERS WHERE USERNAME = 'SCOTT';
```

The status should be OPEN. If the DBA_USERS view lists user SCOTT as locked and expired, then enter the following statement to unlock the SCOTT account and create a new password for him:

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

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure. For greater security, do not reuse the same password that was used in previous releases of Oracle Database.

6. Connect as user SCOTT.

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

7. Create and populate the customers table.

When you enter the user email IDs, enter them in upper-case letters. Later on, when you create the application context PL/SQL package, the <code>SESSION_USER</code> parameter of the <code>SYS_CONTEXT</code> function expects the user names to be in upper case. Otherwise, you will be unable to set the application context for the user.

8. User sysadmin_vpd will need SELECT privileges for the customers table, so as user SCOTT, grant him this privilege.

```
GRANT READ ON customers TO sysadmin vpd;
```

9. Create and populate the orders tab table.

```
CREATE TABLE orders_tab (
  cust_no NUMBER(4),
  order_no NUMBER(4));

INSERT INTO orders_tab VALUES (1234, 9876);
INSERT INTO orders_tab VALUES (5678, 5432);
INSERT INTO orders_tab VALUES (5678, 4592);
```

10. Users throoke and owoods need to query the orders_tab table, so grant them the READ object privilege.

```
GRANT READ ON orders_tab TO tbrooke, owoods;
```

At this stage, the two sample customers, tbrooke and owoods, have a record of purchases in the orders_tab order entry table, and if they tried right now, they can see all the orders in this table.

14.4.2.3 Step 2: Create a Database Session-Based Application Context

Next, you are ready to create the database session-based application context.

1. Connect as user sysadmin vpd.

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

2. Enter the following statement:

```
CREATE OR REPLACE CONTEXT orders_ctx USING orders_ctx_pkg;
```

This statement creates the orders_ctx application context. Remember that even though user sysadmin_vpd has created this context and it is associated with the sysadmin_vpd schema, the SYS schema owns the application context.

14.4.2.4 Step 3: Create a PL/SQL Package to Set the Application Context

After you create the application context, you are ready to create a package to set the context.

 As user sysadmin_vpd, create the following PL/SQL package, which will set the database session-based application context when the customers tbrooke and owoods log onto their accounts.

```
CREATE OR REPLACE PACKAGE orders_ctx_pkg IS
    PROCEDURE set_custnum;
END;
/
CREATE OR REPLACE PACKAGE BODY orders_ctx_pkg IS
    PROCEDURE set_custnum
    AS
        custnum NUMBER;
BEGIN
        SELECT cust_no INTO custnum FROM SCOTT.CUSTOMERS
            WHERE cust_email = SYS_CONTEXT('USERENV', 'SESSION_USER');
        DBMS_SESSION.SET_CONTEXT('orders_ctx', 'cust_no', custnum);
        EXCEPTION
        WHEN NO_DATA_FOUND THEN NULL;
        END set_custnum;
END;
//
```

In this example:

- custnum NUMBER creates the custnum variable, which will hold the customer ID.
- SELECT cust_no INTO custnum performs a SELECT statement to copy the customer ID that is stored in the cust_no column data from the scott.customers table into the custnum variable.
- WHERE cust_email = SYS_CONTEXT('USERENV', 'SESSION_USER') uses a WHERE clause to find all the customer IDs that match the user name of the user who is logging on

- DBMS_SESSION.SET_CONTEXT ('orders_ctx', 'cust_no', custnum) sets the
 orders_ctx application context values by creating the cust_no attribute and then
 setting it to the value stored in the custnum variable.
- EXCEPTION ... WHEN adds a WHEN NO_DATA_FOUND system exception to catch any no data found errors that may result from the SELECT statement in the SELECT cust_no INTO custnum ... statement.

To summarize, the <code>sysadmin_vpd.set_custnum</code> procedure identifies whether or not the session user is a registered customer by attempting to select the user's customer ID into the <code>custnum</code> variable. If the user is a registered customer, then Oracle Database sets an application context value for this user. The policy function uses the context value to control the access a user has to data in the <code>orders tab</code> table.

14.4.2.5 Step 4: Create a Logon Trigger to Run the Application Context PL/SQL Package

The logon trigger runs the PL/SQL package procedure so that the next time a user logs on, the application context is set.

As user sysadmin vpd, create the following logon trigger:

```
CREATE TRIGGER set_custno_ctx_trig AFTER LOGON ON DATABASE
BEGIN
   sysadmin_vpd.orders_ctx_pkg.set_custnum;
END;
//
```

Related Topics

Logon Triggers to Run a Database Session Application Context Package
 Users must run database session application context package after when they log in to the database instance.

14.4.2.6 Step 5: Test the Logon Trigger

The logon trigger sets the application context for the user when the trigger runs the sysadmin_vpd.orders_ctx_pkg.set_custnum procedure.

Connect as user tbrooke.

```
CONNECT throoke -- For a CDB, connect to the PDB, e.g., @hrpdb Enter password: password
```

2. Execute the following query:

```
SELECT SYS_CONTEXT('orders_ctx', 'cust_no') custnum FROM DUAL;
```

The following output should appear:



14.4.2.7 Step 6: Create a PL/SQL Policy Function to Limit User Access to Their Orders

The next step is to create a PL/SQL function to control the display of the user's query.

When the user who has logged in performs a $SELECT * FROM scott.orders_tab$ query, the function should cause the output to display only the orders of that user.

1. Connect as user sysadmin vpd.

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

2. Create the following function:

This function creates and returns a WHERE predicate that translates to "where the orders displayed belong to the user who has logged in." It then appends this WHERE predicate to any queries this user may run against the <code>scott.orders_tab</code> table. Next, you are ready to create an Oracle Virtual Private Database policy that applies this function to the <code>orders_tab</code> table.

14.4.2.8 Step 7: Create the New Security Policy

Finally, you are ready to create the VPD security policy.

 As user sysadmin_vpd, use the DBMS_RLS.ADD_POLICY procedure to create the policy as follows:

This statement creates a policy named orders_policy and applies it to the orders_tab table, which customers will query for their orders, in the SCOTT schema. The get_user_orders function implements the policy, which is stored in the sysadmin_vpd schema. The policy further restricts users to issuing SELECT statements only. The namespace and attribute parameters specify the application context that you created earlier.

14.4.2.9 Step 8: Test the New Policy

Now that you have created all the components, you are ready to test the policy.

1. Connect as user tbrooke.

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

User tbrooke can log on because he has passed the requirements you defined in the application context.

2. As user tbrooke, access your purchases.

```
SELECT * FROM scott.orders tab;
```

The following output should appear:

User throoke has passed the second test. He can access his own orders in the scott.orders tab table.

3. Connect as user owoods, and then access your purchases.

```
CONNECT owoods -- For a CDB, connect to the PDB, e.g., @hrpdb
Enter password: password
SELECT * FROM scott.orders tab
```

The following output should appear:

ORDER_NO	CUST_NO
5432	5678
4592	5678

As with user tbrooke, user owoods can log on and see a listing of his own orders.

Note the following:

- You can create several predicates based on the position of a user. For example, a sales
 representative would be able to see records only for his customers, and an order entry
 clerk would be able to see any customer order. You could expand the custnum_sec
 function to return different predicates based on the user position context value.
- The use of an application context in a fine-grained access control package effectively gives you a bind variable in a parsed statement. For example:

```
SELECT * FROM scott.orders_tab
WHERE cust_no = SYS_CONTEXT('order_entry', 'cust_num');
```

This is fully parsed and optimized, but the evaluation of the <code>cust_num</code> attribute value of the user for the <code>order_entry</code> context takes place at run-time. This means that you get the benefit of an optimized statement that executes differently for each user who issues the statement.





You can improve the performance of the function in this tutorial by indexing <code>cust_no</code>.

 You can set context attributes based on data from a database table or tables, or from a directory server using Lightweight Directory Access Protocol (LDAP).



Oracle Database PL/SQL Language Reference for more information about triggers

Compare this tutorial, which uses an application context within the dynamically generated predicate, with About Oracle Virtual Private Database Policies, which uses a subquery in the predicate.

14.4.2.10 Step 9: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

Connect as user SCOTT.

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

2. Remove the orders tab and customers tables.

```
DROP TABLE orders_tab;
DROP TABLE customers;
```

3. Connect as user SYS, connecting with AS SYSDBA.

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

4. Run the following statements to drop the components for this tutorial:

```
DROP CONTEXT orders_ctx;
DROP USER sysadmin_vpd CASCADE;
DROP USER tbrooke;
DROP USER owoods;
```

14.4.3 Tutorial: Implementing an Oracle Virtual Private Database Policy Group

This tutorial demonstrates how to create an Oracle Virtual Private Database policy group.

- About This Tutorial
 - This tutorial shows how you can use Oracle Virtual Private Database (VPD) to create a policy group.
- Step 1: Create User Accounts and Other Components for This Tutorial
 First, you must create user accounts and tables for this tutorial, and grant the appropriate
 privileges.

Step 2: Create the Two Policy Groups

Next, you must create a policy group for each of the two nondatabase users, provider_a and provider b.

Step 3: Create PL/SQL Functions to Control the Policy Groups

A policy group must have a function that defines how the application can control data access for users.

Step 4: Create the Driving Application Context

The application context determines which policy the nondatabase user who is the logging on should use.

Step 5: Add the PL/SQL Functions to the Policy Groups

Now that you have created the necessary functions, you are ready to associate them with their appropriate policy groups.

Step 6: Test the Policy Groups

Now you are ready to test the two policy groups.

Step 7: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

14.4.3.1 About This Tutorial

This tutorial shows how you can use Oracle Virtual Private Database (VPD) to create a policy group.

Oracle Virtual Private Database Policy Groups describes how you can group a set of policies for use in an application. When a nondatabase user logs onto the application, Oracle Database grants the user access based on the policies defined within the appropriate policy group.

For column-level access control, every column or set of hidden columns is controlled by one policy. In this tutorial, you must hide two sets of columns. So, you must create two policies, one for each set of columns that you want to hide. You only want one policy for each user; the driving application context separates the policies for you.



If you are using a multitenant environment, then this tutorial applies to the current PDB only.

14.4.3.2 Step 1: Create User Accounts and Other Components for This Tutorial

First, you must create user accounts and tables for this tutorial, and grant the appropriate privileges.

1. Log on as user SYS with the SYSDBA administrative privilege.

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

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

For example:

CONNECT SYS@hrpdb AS SYSDBA Enter password: password

To find the available PDBs, run the show pdbs command. To check the current PDB, run the show con name command.

3. Create the following local users:

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

4. Grant the following additional privilege to user sysadmin pg:

```
GRANT EXECUTE ON DBMS RLS TO sysadmin pg;
```

5. Log on as user OE.

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

If the OE account is locked and expired, then reconnect as user SYS with the SYSDBA administrative privilege and enter the following statement to unlock the account and give it s new password:

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

Replace *password* with a password that is secure. For greater security, do not reuse the same password that was used in previous releases of Oracle Database.

6. Create the product code names table:

```
CREATE TABLE product_code_names(
group_a varchar2(32),
year_a varchar2(32),
group_b varchar2(32),
year b varchar2(32));
```

7. Insert some values into the product code names table:

```
INSERT INTO product_code_names values('Biffo','2008','Beffo','2004');
INSERT INTO product_code_names values('Hortensia','2008','Bunko','2008');
INSERT INTO product_code_names values('Boppo','2006','Hortensia','2003');
COMMIT;
```

8. Grant the apps user user SELECT privileges on the product code names table.

```
GRANT SELECT ON product_code_names TO apps_user;
```

14.4.3.3 Step 2: Create the Two Policy Groups

Next, you must create a policy group for each of the two nondatabase users, provider_a and provider b.

1. Connect as user sysadmin pq.

2. Create the provider a group policy group, to be used by user provider a:

```
BEGIN
DBMS RLS.CREATE POLICY GROUP(
```

```
object_schema => 'oe',
object_name => 'product_code_names',
policy_group => 'provider_a_group');
END;
//
```

3. Create the provider b group policy group, to be used by user provider b:

```
BEGIN
DBMS_RLS.CREATE_POLICY_GROUP(
object_schema => 'oe',
object_name => 'product_code_names',
policy_group => 'provider_b_group');
END;
//
```

14.4.3.4 Step 3: Create PL/SQL Functions to Control the Policy Groups

A policy group must have a function that defines how the application can control data access for users.

The function that you will create for this policy group applies to users $provider_a$ and $provider_b$.

 Create the vpd_function_provider_a function, which restricts the data accessed by user provider a.

```
CREATE OR REPLACE FUNCTION vpd_function_provider_a
  (schema in varchar2, tab in varchar2) return varchar2 as
  predicate varchar2(8) default NULL;
  BEGIN
  IF LOWER(SYS_CONTEXT('USERENV','CLIENT_IDENTIFIER')) = 'provider_a'
   THEN predicate := '1=2';
  ELSE NULL;
  END IF;
  RETURN predicate;
END;
//
```

This function checks that the user logging in is really user provider_a. If this is true, then only the data in the product_code_names table columns group_a and year_a will be visible to provider_a. Data in columns group_b and year_b will not appear for provider_a. This works as follows: Setting predicate := '1=2' hides the relevant columns. In Step 5: Add the PL/SQL Functions to the Policy Groups, you specify these columns in the SEC_RELEVANT_COLS parameter.

Create the vpd_function_provider_b, function, which restricts the data accessed by user provider_a.

```
CREATE OR REPLACE FUNCTION vpd_function_provider_b
  (schema in varchar2, tab in varchar2) return varchar2 as
  predicate varchar2(8) default NULL;
  BEGIN
    IF LOWER(SYS_CONTEXT('USERENV','CLIENT_IDENTIFIER')) = 'provider_b'
    THEN predicate := '1=2';
    ELSE NULL;
    END IF;
    RETURN predicate;
END;
/
```

Similar to the <code>vpd_function_provider_a</code> function, this function checks that the user logging in is really user <code>provider_b</code>. If this is true, then only the data in the columns <code>group_b</code> and <code>year_b</code> will be visible to <code>provider_b</code>, with data in the <code>group_a</code> and <code>year_a</code> not appearing for <code>provider_b</code>. Similar to the <code>vpd_function_provider_a</code> function, <code>predicate := '1=2'</code> hides the relevant columns specified <code>Step 5: Add the PL/SQL Functions</code> to the <code>Policy Groups</code> in the <code>SEC_RELEVANT_COLS parameter</code>.

Related Topics

Function to Generate the Dynamic WHERE Clause
 The Oracle Virtual Private Database (VPD) function defines the restrictions that you want to enforce.

14.4.3.5 Step 4: Create the Driving Application Context

The application context determines which policy the nondatabase user who is the logging on should use.

1. As user sysadmin pg, create the driving application context as follows:

```
CREATE OR REPLACE CONTEXT provider ctx USING provider package;
```

2. Create the PL/SQL provider package package for the application context.

```
CREATE OR REPLACE PACKAGE provider_package IS

PROCEDURE set_provider_context (policy_group varchar2 default NULL);

END;

/

CREATE OR REPLACE PACKAGE BODY provider_package AS

PROCEDURE set_provider_context (policy_group varchar2 default NULL) IS

BEGIN

CASE LOWER(SYS_CONTEXT('USERENV', 'CLIENT_IDENTIFIER'))

WHEN 'provider_a' THEN

DBMS_SESSION.SET_CONTEXT('provider_ctx', 'policy_group', 'PROVIDER_A_GROUP');

WHEN 'provider_b' THEN

DBMS_SESSION.SET_CONTEXT('provider_ctx', 'policy_group', 'PROVIDER_B_GROUP');

END CASE;

END set_provider_context;

END;

/
```

3. Associate the provider_ctx application context with the product_code_names table, and then provide a name.

4. Grant the apps user account the EXECUTE privilege for the provider package package.

```
GRANT EXECUTE ON provider package TO apps user;
```

14.4.3.6 Step 5: Add the PL/SQL Functions to the Policy Groups

Now that you have created the necessary functions, you are ready to associate them with their appropriate policy groups.

1. Add the vpd function provider a function to the provider a group policy group.

The group_b and year_b columns specified in the sec_relevant_cols parameter are hidden from user provider a.

2. Add the vpd function provider b function to the provider b group policy group.

The group_a and year_a columns specified in the sec_relevant_cols parameter are hidden from user provider b.

14.4.3.7 Step 6: Test the Policy Groups

Now you are ready to test the two policy groups.

 Connect as user apps_user and then enter the following statements to ensure that the output you will create later on is nicely formatted.

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

col group_a format a16

col group_b format a16;

col year_a format a16;

col year_b format a16;
```

2. Set the session identifier to provider a.

```
EXEC DBMS SESSION.SET IDENTIFIER('provider a');
```

Here, the application sets the identifier. Setting the identifier to provider_a sets the apps_user user to a user who should only see the products available to products in the provider a group policy group.

3. Run the provider package to set the policy group based on the context.

```
EXEC sysadmin_pg.provider_package.set_provider_context;
```

At this stage, you can check the application context was set, as follows:

```
SELECT SYS CONTEXT('USERENV', 'CLIENT IDENTIFIER') AS END USER FROM DUAL;
```

The following output should appear:

```
END_USER
-----
provider a
```

4. Enter the following SELECT statement:

```
SELECT * FROM oe.product_code_names;
```

The following output should appear:

GROUP_A	YEAR_A	GROUP_B	YEAR_B
Biffo	2008		
Hortensia	2008		
Ворро	2006		

5. Set the client identifier to provider b and then enter the following statements:

```
EXEC DBMS_SESSION.SET_IDENTIFIER('provider_b');
EXEC sysadmin_pg.provider_package.set_provider_context;
SELECT * FROM oe.product_code_names;
```

The following output should appear:

GROUP_A	YEAR_A	GROUP_B	YEAR_B
		Beffo	2004
		Bunko	2008
		Hortensia	2003

14.4.3.8 Step 7: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

1. Connect as user OE.

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

2. Drop the product code names table.

```
DROP TABLE product_code_names;
```

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

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

4. Drop the application context and users for this tutorial.

DROP CONTEXT provider_ctx; DROP USER sysadmin_pg cascade; DROP USER apps_user;

14.5 How Oracle Virtual Private Database Works with Other Oracle Features

You should be aware of the impact of using Oracle Virtual Private Database with other Oracle features.

- Oracle Virtual Private Database Policies with Editions
 You should be aware of how to use Oracle VPD with editions.
- SELECT FOR UPDATE Statement in User Queries on VPD-Protected Tables
 As a general rule, users should not include the FOR UPDATE clause when querying Virtual Private Database-protected tables.
- Oracle Virtual Private Database Policies and Outer or ANSI Joins
 Oracle Virtual Private Database rewrites SQL by using dynamic views.
- Oracle Virtual Private Database Security Policies and Applications
 An Oracle Virtual Private Database security policy is applied within the database itself, rather than within an application.
- Automatic Reparsing for Fine-Grained Access Control Policies Functions
 Queries against objects enabled with fine-grained access control run the policy function so
 that the most current predicate is used for each policy.
- Oracle Virtual Private Database Policies and Flashback Queries
 Operations on the database use the most recently committed data available.
- Oracle Virtual Private Database and Oracle Label Security
 You can use Oracle Virtual Private Database with Oracle Label Security, but when you do, you should be aware of security exceptions.
- Export of Data Using the EXPDP Utility access_method Parameter
 Be aware if you try to export data from objects that have VPD policies defined on them.
- User Models and Oracle Virtual Private Database
 You can use Oracle Virtual Private Database in several types of user models.

14.5.1 Oracle Virtual Private Database Policies with Editions

You should be aware of how to use Oracle VPD with editions.

If you are preparing an application for edition-based redefinition, and you cover each table that the application uses with an editioning view, then you must move the Virtual Private Database polices that protect these tables to the editioning view.

When an editioned object has a Virtual Private Database policy, then it applies in all editions in which the object is visible. When an editioned object is actualized, any VPD policies that are attached to it are newly attached to the new actual occurrence. When you newly apply a VPD policy to an inherited editioned object, this action will actualize it.





Oracle Database Development Guide for detailed information about editions

14.5.2 SELECT FOR UPDATE Statement in User Queries on VPD-Protected Tables

As a general rule, users should not include the FOR UPDATE clause when querying Virtual Private Database-protected tables.

The Virtual Private Database technology depends on rewriting the user's query against an inline view that includes the VPD predicate generated by the VPD policy function. Because of this, the same limitations on views also apply to VPD-protected tables. If a user's query against a VPD-protected table includes the FOR UPDATE clause in a SELECT statement, in most cases, the query may not work. However, the user's query may work in some situations if the inline view generated by VPD is sufficiently simple.



Oracle Database SQL Language Reference for more information about the restrictions of the FOR UPDATE clause in the SELECT statement

14.5.3 Oracle Virtual Private Database Policies and Outer or ANSI Joins

Oracle Virtual Private Database rewrites SQL by using dynamic views.

For SQL that contains outer join or ANSI operations, some views may not merge and some indexes may not be used. This problem is a known optimization limitation. To remedy this problem, rewrite the SQL to not use outer joins or ANSI operations.

14.5.4 Oracle Virtual Private Database Security Policies and Applications

An Oracle Virtual Private Database security policy is applied within the database itself, rather than within an application.

Hence, a user trying to access data by using a different application cannot bypass the Oracle Virtual Private Database security policy. Another advantage of creating the security policy in the database is that you maintain it in one central place, rather than maintaining individual security policies in multiple applications. Oracle Virtual Private Database provides stronger security than application-based security, at a lower cost of ownership.

You may want to enforce different security policies depending on the application that is accessing data. Consider a situation in which two applications, Order Entry and Inventory, both access the orders table. You may want to have the Inventory application use a policy that limits access based on type of product. At the same time, you may want to have the Order Entry application use a policy that limits access based on customer number.

In this case, you must partition the use of fine-grained access by application. Otherwise, both policies would be automatically concatenated together, which may not be the result that you want. You can specify two or more policy groups, and a driving application context that

determines which policy group is in effect for a given transaction. You can also designate default policies that always apply to data access. In a hosted application, for example, data access should be limited by subscriber ID.

Related Topics

Tutorial: Implementing an Oracle Virtual Private Database Policy Group
 This tutorial demonstrates how to create an Oracle Virtual Private Database policy group.

14.5.5 Automatic Reparsing for Fine-Grained Access Control Policies Functions

Queries against objects enabled with fine-grained access control run the policy function so that the most current predicate is used for each policy.

For example, in the case of a time-based policy function, in which queries are only allowed between 8:00 a.m. and 5:00 p.m., a cursor execution parsed at noon runs the policy function at that time, ensuring that the policy is consulted again for the query. Even if the curser was parsed at 9 a.m., when it runs later on (for example, at noon), then the Virtual Private Database policy function runs again to ensure that the execution of the cursor is still permitted at the current time (noon). This ensures that the security check it must perform is the most recent.

Automatic re-execution of the Virtual Private Database policy function does not occur when you set the DBMS_RLS.ADD_POLICY setting STATIC_POLICY to TRUE while adding the policy. This setting causes the policy function to return the same predicate.

14.5.6 Oracle Virtual Private Database Policies and Flashback Queries

Operations on the database use the most recently committed data available.

The flashback query feature enables you to query the database at some point in the past.

To write an application that uses flashback query, you can use the AS OF clause in SQL queries to specify either a time or a system change number (SCN), and then query against the committed data from the specified time. You can also use the DBMS_FLASHBACK PL/SQL package, which requires more code, but enables you to perform multiple operations, all of which refer to the same point in time.

However, if you use flashback query against a database object that is protected with Oracle Virtual Private Database policies, then the current policies are applied to the old data. Applying the current Oracle Virtual Private Database policies to flashback query data is more secure because it reflects the most current business policy.

See Also:

- Oracle Database Development Guide for more information about the flashback query feature and how to write applications that use it
- Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS FLASHBACK PL/SQL package



14.5.7 Oracle Virtual Private Database and Oracle Label Security

You can use Oracle Virtual Private Database with Oracle Label Security, but when you do, you should be aware of security exceptions.

- Using Oracle Virtual Private Database to Enforce Oracle Label Security Policies
 Oracle Virtual Private Database policies provide column or row-level access control based on Oracle Label Security user authorizations.
- Oracle Virtual Private Database and Oracle Label Security Exceptions
 Be aware of the security exceptions when you use Oracle Virtual Private Database and Oracle Label Security.

14.5.7.1 Using Oracle Virtual Private Database to Enforce Oracle Label Security Policies

Oracle Virtual Private Database policies provide column or row-level access control based on Oracle Label Security user authorizations.

In general, you must perform the following steps:

- 1. When you create the Oracle Label Security policy, do not apply the policy to the table that you want to protect. (The Virtual Private Database policy that you create handles this for you.) In the SA_SYSDBA.CREATE_POLICY procedure, set the default_options parameter to NO CONTROL.
- 2. Create the Oracle Label Security label components and authorize users as you normally would
- 3. When you create the Oracle Virtual Private Database policy, do the following:
 - In the PL/SQL function you create for the policy, use the Oracle Label Security DOMINATES function to compare the authorization of the user with the label that you created in Step 2. The DOMINATES function determines if the user authorization is equal to, or if it is more sensitive than, the label used in the comparison. If the user authorization passes, then the user is granted access to the column. Otherwise, the user is denied access.
 - In the Virtual Private Database policy definition, apply this function to the table that you want to protect. In the <code>DBMS_RLS.ADD_POLICY</code> procedure, use the sensitive column (<code>SEC_RELEVANT_COLS</code> parameter) and column masking (<code>SEC_RELEVANT_COLS_OPT</code> parameter) functionality to show or hide columns based on Oracle Label Security user authorizations.

See Also:

Oracle Label Security Administrator's Guide for more information about the dominance functions

14.5.7.2 Oracle Virtual Private Database and Oracle Label Security Exceptions

Be aware of the security exceptions when you use Oracle Virtual Private Database and Oracle Label Security.



These security exceptions are as follows:

- When you are exporting data, Oracle Virtual Private Database and Oracle Label
 Security policies are not enforced during a direct path export operation. In a direct
 path export operation, Oracle Database reads data from disk into the buffer cache and
 transfers rows directly to the Export client.
- You cannot apply Oracle Virtual Private Database policies and Oracle Label Security policies to objects in the SYS schema. The SYS user and users making a DBA-privileged connection to the database (for example, CONNECT/AS SYSDBA) do not have Oracle Virtual Private Database or Oracle Label Security policies applied to their actions. The database user SYS is thus always exempt from Oracle Virtual Private Database or Oracle Label Security enforcement, regardless of the export mode, application, or utility used to extract data from the database.

However, you can audit SYSDBA actions by enabling auditing upon installation and specifying that this audit trail be stored in a secure location in the operating system. You can also closely monitor the SYS user by using Oracle Database Vault.

• Database users who were granted the EXEMPT ACCESS POLICY privilege, either directly or through a database role, are exempt from Oracle Virtual Private Database enforcements. The system privilege EXEMPT ACCESS POLICY allows a user to be exempted from all fine-grained access control policies on any SELECT or DML operation (INSERT, UPDATE, and DELETE). This provides ease of use for administrative activities, such as installation and import and export of the database, through a non-sys schema.

However, the following policy enforcement options remain in effect even when EXEMPT ACCESS POLICY is granted:

- INSERT_CONTROL, UPDATE_CONTROL, DELETE_CONTROL, WRITE_CONTROL, LABEL_UPDATE,
 and LABEL DEFAULT
- If the Oracle Label Security policy specifies the ALL_CONTROL option, then all
 enforcement controls are applied except READ CONTROL and CHECK CONTROL.

Because EXEMPT ACCESS POLICY negates the effect of fine-grained access control, you should only grant this privilege to users who have legitimate reasons for bypassing fine-grained access control enforcement. Do not grant this privilege using the WITH ADMIN OPTION. If you do, users could pass the EXEMPT ACCESS POLICY privilege to other users, and thus propagate the ability to bypass fine-grained access control.

Note:

- The EXEMPT ACCESS POLICY privilege does not affect the enforcement of object privileges such as SELECT, INSERT, UPDATE, and DELETE. These privileges are enforced even if a user was granted the EXEMPT ACCESS POLICY privilege.
- The SYS_CONTEXT values that Oracle Virtual Private Database uses are not propagated to secondary databases for failover.

See Also:

Oracle Database Utilities for more information about direct path export operations



14.5.8 Export of Data Using the EXPDP Utility access_method Parameter

Be aware if you try to export data from objects that have VPD policies defined on them.

If you try to use the Oracle Data Pump Export (EXPDP) utility with the access_method parameter set to direct_path to export data from a schema that contains an object that has a Virtual Private Database policy defined on it, then an ORA-31696 error message may appear and the export operation will fail.

The error message is as follows:

ORA-31696: unable to export/import TABLE_DATA:"schema.table" using client specified DIRECT_PATH method

This problem occurs when you perform a schema-level export or a full database export, which requires the <code>EXP_FULL_DATABASE</code> role. To perform an export with VPD policies in place using the <code>access_method=direct_path</code> parameter, the exporting user must be granted the system privilege <code>EXEMPT ACCESS POLICY</code>. <code>EXEMPT ACCESS POLICY</code> bypasses Virtual Private Database policies. Note that the <code>EXP_FULL_DATABASE</code> role does not include the <code>EXEMPT ACCESS POLICY</code> system privilege.

To find the underlying problem, try the EXPDP invocation again, but do not set the access_method parameter to direct_path. Instead, use either automatic or external_table. The underlying problem could be a permissions problem, for example:

ORA-39181: Only partial table data may be exported due to fine grain access control on "schema name". "object name"

14.5.9 User Models and Oracle Virtual Private Database

You can use Oracle Virtual Private Database in several types of user models.

These user models are as follows:

- Application users who are also database users. Oracle Database enables applications to enforce fine-grained access control for each user, regardless of whether that user is a database user or an application user unknown to the database. When application users are also database users, Oracle Virtual Private Database enforcement works as follows: users connect to the database, and then the application sets up application contexts for each session. (You can use the default USERENV application context namespace, which provides many parameters for retrieve different types of user session data.) As each session is initiated under a different user name, it can enforce different fine-grained access control conditions for each user.
- **Proxy authentication using OCI or JDBC/OCI.** Proxy authentication permits different fine-grained access control for each user, because each session (OCI or JDBC/OCI) is a distinct database session with its own application context.
- Proxy authentication integrated with Enterprise User Security. If you have integrated
 proxy authentication by using Enterprise User Security, you can retrieve user roles and
 other attributes from Oracle Internet Directory to enforce Oracle Virtual Private Database
 policies. (In addition, globally initialized application context can also be retrieved from the
 directory.)
- Users connecting as One Big Application User. Applications connecting to the database as a single user on behalf of all users can have fine-grained access control for each user. The user for that single session is often called *One Big Application User*. Within the context of that session, however, an application developer can create a global



application context attribute to represent the individual application user (for example, REALUSER). Although all database sessions and audit records are created for One Big Application User, the attributes for each session can vary, depending on who the end user is. This model works best for applications with a limited number of users and no reuse of sessions. The scope of roles and database auditing is diminished because each session is created as the same database user.

• Web-based applications. Web-based applications typically have hundreds of users. Even when there are persistent connections to the database, supporting data retrieval for many user requests, these connections are not specific to particular Web-based users. Instead, Web-based applications typically set up and reuse connections, to provide scalability, rather than having different sessions for each user. For example, when Web users Jane and Ajit connect to a middle tier application, it may establish a single database session that it uses on behalf of both users. Typically, neither Jane nor Ajit is known to the database. The application is responsible for switching the user name on the connection, so that, at any given time, it is either Jane or Ajit using the session.

Oracle Virtual Private Database helps with connection pooling by allowing multiple connections to access more than one global application context. This ability makes it unnecessary to establish a separate application context for each distinct user session.

Table 14-3 summarizes how Oracle Virtual Private Database applies to user models.

Table 14-3 Oracle Virtual Private Database in Different User Models

User Model Scenario	Individual Database Connection	Separate Application Context per User	Single Database Connection	Application Must Switch User Name
Application users are also database users	Yes	Yes	No	No
Proxy authentication using OCI or JDBC/OCI	Yes	Yes	No	No
Proxy authentication integrated with Enterprise User Security ¹	No	No	Yes	Yes
One Big Application User	No	No ²	No	Yes ²
Web-based applications	No	No	Yes	Yes

¹ User roles and other attributes, including globally initialized application context, can be retrieved from Oracle Internet Directory to enforce Oracle Virtual Private Database.

Related Topics

Global Application Contexts

You can use a global application context to access application values across database sessions, including an Oracle Real Application Clusters environment.

14.6 Oracle Virtual Private Database Data Dictionary Views

Oracle Database provides data dictionary views that list information about Oracle Virtual Private Database policies.

Table 14-4 lists Virtual Private Database-specific views



Application developers can create a global application context attribute representing individual application users (for example, REALUSER), which can then be used for controlling each session attributes, or for auditing.

Table 14-4 Data Dictionary Views That Display Information about VPD Policies

View	Description
ALL_POLICIES	Describes all Oracle Virtual Private Database security policies for objects accessible to the current user.
ALL_POLICY_ATTRIBUTES	Describes all the application context namespaces, attributes, and Virtual Private Database policy associations where the logged in user is the owner of the VPD policy or the VPD policy belongs to PUBLIC.
ALL_POLICY_CONTEXTS	Describes the driving contexts defined for the synonyms, tables, and views accessible to the current user. A driving context is an application context used in an Oracle Virtual Private Database policy.
ALL_POLICY_GROUPS	Describes the Oracle Virtual Private Database policy groups defined for the synonyms, tables, and views accessible to the current user
ALL_SEC_RELEVANT_COLS	Describes the security relevant columns of the security policies for the tables and views accessible to the current user
DBA_POLICIES	Describes all Oracle Virtual Private Database security policies in the database.
DBA_POLICY_ATTRIBUTES	Describes all the application context namespaces, attributes, and Virtual Private Database policy associations for context-sensitive and shared context-sensitive Virtual Private Database policies
DBA_POLICY_GROUPS	Describes all policy groups in the database.
DBA_POLICY_CONTEXTS	Describes all driving contexts in the database. Its columns are the same as those in <code>ALL_POLICY_CONTEXTS</code> .
DBA_SEC_RELEVANT_COLS	Describes the security relevant columns of all security policies in the database
UNIFIED_AUDIT_TRAIL	Captures the VPD predicates in the ${\tt RLS_INFO}$ column, for unified auditing and fine-grained auditing
USER_POLICIES	Describes all Oracle Virtual Private Database security policies associated with objects owned by the current user. This view does not display the <code>OBJECT_OWNER</code> column.
USER_POLICY_ATTRIBUTES	Describes all the application context namespaces, attributes, and Virtual Private Database policy associations where the owner of the Virtual Private Database policy is the current user
USER_POLICY_CONTEXTS	Describes the driving contexts defined for the synonyms, tables, and views owned by the current user. Its columns (except for <code>OBJECT_OWNER</code>) are the same as those in <code>ALL_POLICY_CONTEXTS</code> .
USER_SEC_RELEVANT_COLS	Describes the security relevant columns of the security policies for the tables and views owned by the current user. Its columns (except for <code>OBJECT_OWNER</code>) are the same as those in <code>ALL_SEC_RELEVANT_COLS</code> .
USER_POLICY_GROUPS	Describes the policy groups defined for the synonyms, tables, and views owned by the current user. This view does not display the <code>OBJECT_OWNER</code> column.
V\$VPD_POLICY	For the current PDB, displays all the fine-grained security policies and predicates associated with the cursors currently in the library cache. This view is useful for finding the policies that were applied to a SQL statement.





Tip:

In addition to these views, check the database trace file if you find errors in application that use Virtual Private Database policies. The ${\tt USER}\ {\tt DUMP}\ {\tt DEST}$ initialization parameter specifies the current location of the trace files. You can find the value of this parameter by issuing show parameter user dump dest in SQL*Plus.

Related Topics

- Oracle Database Reference
- Oracle Database SQL Tuning Guide



Using Transparent Sensitive Data Protection

Transparent sensitive data protection enables you to find all table columns in a database that hold sensitive data.

- About Transparent Sensitive Data Protection
 Transparent sensitive data protection is a way to find and classify table columns that hold sensitive information.
- General Steps for Using Transparent Sensitive Data Protection
 To use TSDP with Oracle Data Redaction, you must follow a set of general steps.
- Use Cases for Transparent Sensitive Data Protection Policies Transparent sensitive data protection has several benefits.
- Privileges Required for Using Transparent Sensitive Data Protection
 To use transparent sensitive data protection, you must have the EXECUTE privilege for several PL/SQL packages.
- How a Multitenant Environment Affects Transparent Sensitive Data Protection
 In a multitenant environment, you can apply Transparent Sensitive Data Protection policies
 to the current PDB or current application PDB only.
- Creating Transparent Sensitive Data Protection Policies
 You must create a sensitive type, find the sensitive columns to be protected, and then
 import these columns from ADM into your database.
- Altering Transparent Sensitive Data Protection Policies
 The DBMS_TSDP_PROTECT.ALTER_POLICY procedure can alter a TSDP policy.
- Disabling Transparent Sensitive Data Protection Policies
 The DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN procedure disables one or all TSDP policies.
- Dropping Transparent Sensitive Data Protection Policies
 You can drop an entire TSDP policy or a condition-enable-options combination from the
 policy.
- Using the Predefined REDACT_AUDIT Policy to Mask Bind Values
 The predefined REDACT_AUDIT policy masks bind values, which can appear in trace files when an event is set.
- Transparent Sensitive Data Protection Policies with Data Redaction
 Oracle Data Redaction features work with transparent sensitive data protection policies.
- Using Transparent Sensitive Data Protection Policies with Oracle VPD Policies
 You can combine protections from TSDP and Oracle Virtual Private Database into one
 policy.
- Using Transparent Sensitive Data Protection Policies with Unified Auditing
 The transparent sensitive data protection and unified auditing procedures can combine the protections of these two features.
- Using Transparent Sensitive Data Protection Policies with Fine-Grained Auditing
 The transparent sensitive data protection and fine-grained auditing procedures can
 combine the protections of these two features.

- Using Transparent Sensitive Data Protection Policies with TDE Column Encryption
 The TSDP procedures and Transparent Data Encryption column encryption statements
 can combine the protections of these two features.
- Transparent Sensitive Data Protection Data Dictionary Views
 Oracle Database provides data dictionary views that list information about transparent sensitive data protection policies.

15.1 About Transparent Sensitive Data Protection

Transparent sensitive data protection is a way to find and classify table columns that hold sensitive information.

This feature enables you to quickly find the table columns in a database that hold sensitive data, classify this data, and then create a policy that protects this data as a whole for a given class. Examples of this type of sensitive data are credit card numbers or Social Security numbers.

The TSDP policy then protects the sensitive data in these table columns by using either Oracle Data Redaction or Oracle Virtual Private Database settings. The TSDP policy applies at the column level of the table that you want to protect, targeting a specific column data type, such as all NUMBER data types of columns that contain credit card information. You can create a uniform TSDP policy for all of the data that you classify, and then modify this policy as necessary, as compliance regulations change. Optionally, you can export the TSDP policies for use in other databases.

The benefits of TSDP policies are enormous: You easily can create and apply TSDP policies throughout a large organization with numerous databases. This helps auditors greatly by enabling them to estimate the protection for the data that the TSDP policies target. TSDP is particularly useful for government environments, in which you may have a lot of data with similar security restrictions and you must apply a policy to all of this data consistently. The policy could be to redact it, encrypt it, control access to it, audit access to it, and mask it in the audit trail. Without TSDP, you would have to configure every redaction policy, column-level encryption configuration, and Virtual Private Database policy column by column.

15.2 General Steps for Using Transparent Sensitive Data Protection

To use TSDP with Oracle Data Redaction, you must follow a set of general steps.

- 1. Create a sensitive type to classify the types of columns that you want to protect.
 - For example, you can create a sensitive type for classify all Social Security numbers or credit card numbers. To create the sensitive type, either use the DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE PL/SQL procedure or use an Enterprise Manager Cloud Control Application Data Model. To add multiple sensitive types in one operation from an Application Data Model, you can use the DBMS_TSDP_MANAGE.IMPORT_SENSITIVE_TYPES procedure.
- Identify a list of sensitive columns that are associated with the sensitive types.

To determine and generate this list, you can use either of the following methods:

The DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN procedure individually identifies sensitive columns.



- An Oracle Enterprise Manager Cloud Control Application Data Model enables you to identify a group of sensitive columns. It then prepares this list of sensitive columns in XML format, which you then import into your database.
- 3. If you used an Application Data Model for Step 2, then import the list of sensitive columns from the Application Data Model into your database by using the DBMS TSDP MANAGE.IMPORT DISCOVERY RESULT procedure.
- 4. Create the TSDP policy by using the DBMS_TSDP_PROTECT.ADD_POLICY procedure within an anonymous block that defines the Data Redaction or Virtual Private Database settings that you want to use.
- Associate the TSDP policy with one or more sensitive types by using the DBMS TSDP PROTECT.ASSOCIATE POLICY procedure.
- 6. Enable the TSDP policy protections by using the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_SOURCE, DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN, or the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE procedure.
- Optionally, export the TSDP policy to other databases by using Oracle Data Pump to perform a full database export. (You cannot individually export TSDP policies.)

15.3 Use Cases for Transparent Sensitive Data Protection Policies

Transparent sensitive data protection has several benefits.

These benefits are as follows:

- You configure the sensitive data protection once, and then deploy this protection as necessary. You can configure transparent sensitive data protection policies to designate how a class of data (for example, credit card columns) must be protected without actually having to specify the target data. In other words, when you create the transparent sensitive data protection policy, you do not need to include references to the actual target columns that you want to protect. The transparent sensitive data protection policy finds these target columns based on a list of sensitive columns in the database and the policy's associations with the specified sensitive types. This can be useful when you add more sensitive data to your databases after you have created the transparent sensitive data protection policies. After you create the policy, you can enable protection for the sensitive data in a single step (for example, enable protection based on the entire source database). The sensitive type of the new data and the sensitive type and policy associations determine how the sensitive data is protected. In this way, as new sensitive data is added, you do not need to configure its protection, as long as it meets the current transparent sensitive data protection policy's requirements.
- You can manage protection of multiple sensitive columns. You can enable or disable protection for multiple sensitive columns based on a suitable attribute (such as the source database of the identification, the sensitive type itself, or a specific schema, table, or column). This granularity provides a high level of control over data security. The design of this feature enables you to manage data security based on specific compliance needs for large data sets that fall under the purview of these compliance regulations. You can configure data security based on a specific category rather than for each individual column. For example, you can configure protection for credit card numbers or Social Security numbers, but you do not need to configure protection for each and every column in the database that contains this data.



You can protect the sensitive columns identified using the Oracle Enterprise
Manager Cloud Control Application Data Modeling (ADM) feature. You can use the
Cloud Control ADM feature to create sensitive types and discover a list of sensitive
columns. Then you can import this list of sensitive columns and their corresponding
sensitive types into your database. From there, you can create and manage transparent
sensitive data protection policies using this information.

15.4 Privileges Required for Using Transparent Sensitive Data Protection

To use transparent sensitive data protection, you must have the EXECUTE privilege for several PL/SQL packages.

These privileges are as follows:

- DBMS_TSDP_MANAGE, which enables you to import and manage sensitive columns and sensitive types into your database. The procedures in this package execute with invoker's rights. Typically, an application database administrator will be granted privileges for this package.
- DBMS_TSDP_PROTECT, which you use to create the TSDP policy. The procedures in this
 package execute with invoker's rights. Typically, a security database administrator will be
 granted privileges for this package.
- DBMS_REDACT, if you plan to create Data Redaction policies. Typically, a security database administrator will be granted privileges for this package.
- DBMS_RLS, if you plan to incorporate Oracle Virtual Private Database functionality into your TSDP policies. Typically, a security database administrator will be granted privileges for this package.

For better separation of duty, these packages are designed so that either an application database administrator has control over one area of the TSDP policy creation (as in the case of the DBMS_TSDP_MANAGE package) or a security database administrator (for the DBMS_TSDP_PROTECT, DBMS_REDACT, and DBMS_RLS packages).

15.5 How a Multitenant Environment Affects Transparent Sensitive Data Protection

In a multitenant environment, you can apply Transparent Sensitive Data Protection policies to the current PDB or current application PDB only.

If you are using Enterprise Manager Cloud Control Application Data Model, then you can find sensitive columns that belong to both local and common application objects (that is, common objects that are visible and accessible in the current PDB) inside the PDB. This enables you to use a TSDP policy to protect both local objects to the PDB and common objects that are accessible from the PDB.

In an application root:

- For application containers in general:
 - When you create scripts for application install, upgrade, patch, or uninstall operations, you can include SQL statements within the ALTER PLUGGABLE DATABASE app_name BEGIN INSTALL and ALTER PLUGGABLE DATABASE app_name END INSTALL blocks to perform various operations. If you include TSDP statements within these blocks, then



the TSDP statements will fail. You can, however, include TSDP statements outside these blocks in the script.

- In the application root:
 - You can perform TSDP operations in both application common objects and application root local objects.
 - A TSDP policy that is defined in the application root container behaves as if it is a local
 policy to the application root. That is, the policy is effective only in the application root
 container.

In an application PDB:

- The security policies that protect an application PDB apply to TSDP operations that are performed on local application objects.
- The security policies that protect an application PDB apply to TSDP operations that are
 performed on application common objects that are accessed from the PDB. However,
 access to the application common object outside the application PDB is not governed by
 the security policy that protects the application PDB.

You can find a listing of TSDP policies and the security features that are associated with them by querying the <code>DBA_TSDP_POLICY_FEATURE</code> data dictionary views. To find all PDBs, query the <code>DBA_PDBS</code> view.

15.6 Creating Transparent Sensitive Data Protection Policies

You must create a sensitive type, find the sensitive columns to be protected, and then import these columns from ADM into your database.

- Step 1: Create a Sensitive Type
 The sensitive type is a class of data that you designate as sensitive.
- Step 2: Identify the Sensitive Columns to Protect
 After you define a sensitive column, you are ready to identify the columns to protect.
- Step 3: Import the Sensitive Columns List from ADM into Your Database
 Next, you are ready to import the sensitive columns list from ADM into your database.
- Step 4: Create the Transparent Sensitive Data Protection Policy
 After you have created the list of sensitive columns and imported this list into your database, you can create the transparent sensitive data protection policy.
- Step 5: Associate the Policy with a Sensitive Type
 The DBMS_TSDP_PROTECT.ASSOCIATE_POLICY procedure associates a TSDP policy with a sensitive type.
- Step 6: Enable the Transparent Sensitive Data Protection Policy
 You can enable the TSDP policy for the current database in a protected source, a specific
 table column, or a specific column type.
- Step 7: Optionally, Export the Policy to Other Databases
 You can export or import the policy to or from another database.

15.6.1 Step 1: Create a Sensitive Type

The sensitive type is a class of data that you designate as sensitive.

For example, you can create a credit card type sensitive type for all credit card numbers.



 To create a sensitive type, either create it from an Enterprise Manager Cloud Control Application Data Model or use the DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE PL/SQL procedure.

To drop a sensitive type, you can use the <code>DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE</code> procedure.

For example, to create the sensitive type credit card num type:

```
BEGIN

DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
   sensitive_type => 'credit_card_num_type',
   user_comment => 'Type for credit card columns using a number data type');
END;
//
```

In this example:

- sensitive_type: Create a name that describes the sensitive type that you want to capture.
 This value is case sensitive, so when you reference it later on, ensure that you use the case in which you created it. You can find existing sensitive types by querying the DBA SENSITIVE COLUMN TYPES data dictionary view.
- user comment: Optionally, enter a description for the sensitive type.

See Also:

- Oracle Database Testing Guide for detailed information about Application Data Models
- Oracle Database PL/SQL Packages and Types Reference information about the DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE PL/SQL procedure

15.6.2 Step 2: Identify the Sensitive Columns to Protect

After you define a sensitive column, you are ready to identify the columns to protect.

To identify the columns to protect, based on the sensitive type that you defined, you either can use an Enterprise Manager Cloud Control Application Data Model to identify these columns, or you can use the <code>DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN</code> procedure.

To remove the column from the list of sensitive columns for the database, you can use the DBMS_TSDP_MANAGE.DROP_SENSITIVE_COLUMN procedure.

1. Find the sensitive type that you want to use.

For example:

```
SELECT NAME FROM DBA_SENSITIVE_COLUMN_TYPES;

NAME

credit_card_num_type
```

2. Execute the DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN procedure to associate the sensitive type with a table column. Ensure that you enter the sensitive_type parameter using the case in which you used to create the sensitive type.

For example:

15.6.3 Step 3: Import the Sensitive Columns List from ADM into Your Database

Next, you are ready to import the sensitive columns list from ADM into your database.

 If you had used an Application Data Model to create the list of sensitive columns, then import this list into your database by running the DBMS TSDP MANAGE.IMPORT DISCOVERY RESULT procedure.

If you had used the <code>DBMS_TSDP_MANAGE.ADD_SENSITIVE_COLUMN</code> procedure to identify these columns, then you can bypass this step.

For example, to import the Cloud Control Application Data Model into the current database:

```
BEGIN
  DBMS_TSDP_MANAGE.IMPORT_DISCOVERY_RESULT (
  discovery_result => xml_adm_result,
  discovery_source => 'ADM_Demo');
END;
/
```

In this example:

- discovery_result refers to the list of sensitive columns and their associated sensitive types. This list is in XML format.
- discover_source refers to the name of the Application Data Model that contains the list of sensitive columns referred by the discovery_result setting. You can find a list of the Application Data Models from the Data Discovery and Modeling page in Enterprise Manager Cloud Control. (To access this page, from the Enterprise menu, select Quality Management, and then Data Discovery and Modeling. You can find a list of the sensitive columns and their associated types in the Sensitive Columns tab.)

15.6.4 Step 4: Create the Transparent Sensitive Data Protection Policy

After you have created the list of sensitive columns and imported this list into your database, you can create the transparent sensitive data protection policy.

- About Creating the Transparent Sensitive Data Protection Policy
 The DBMS_TSDP_PROTECT.ADD_POLICY procedure creates the transparent sensitive data protection policy.
- Creating the Transparent Sensitive Data Protection Policy
 You can create a transparent sensitive data protection policy that uses a partial number
 data type-based Data Redaction policy.

- Setting the Oracle Data Redaction or Virtual Private Database Feature Options
 The TSDP feature options describe the Oracle Data Redaction or Virtual Private Database
 settings to use for the transparent sensitive data protection policy.
- Setting Conditions for the Transparent Sensitive Data Protection Policy
 Optionally, you can specify conditions for the transparent sensitive data protection policy.
- Specifying the DBMS_TSDP_PROTECT.ADD_POLICY Procedure
 The DBMS_TSDP_PROTECT.ADD_POLICY procedure names the TSDP policy and executes the
 FEATURE OPTIONS and POLICY CONDITIONS settings.

15.6.4.1 About Creating the Transparent Sensitive Data Protection Policy

The DBMS_TSDP_PROTECT.ADD_POLICY procedure creates the transparent sensitive data protection policy.

After you have identified the sensitive columns, and if you had used an Application Data Model to create the list of sensitive columns, and imported this list into your database, you are ready to create the transparent sensitive data protection policy. To create the transparent sensitive data protection policy, you must configure it for the Virtual Private Database or Oracle Data Redaction settings that you want to use, and then apply these settings to a transparent sensitive data protection policy defined by DBMS TSDP PROTECT.ADD POLICY.

You can create the policy by defining an anonymous block that has the following components:

- If you are using Oracle Data Redaction for your policy, a specification of the type of Data Redaction that you want to use, such as partial Data Redaction
- If you are using Oracle Virtual Private Database for your policy, a specification of the VPD settings that you want to use
- Conditions to test when the policy is enabled. For example, the data type of the column which should be satisfied before the policy can be enabled.
- A named transparent sensitive data protection policy to tie these components together, by using the DBMS_TSDP_PROTECT.ADD POLICY procedure

After you create the sensitive type, it resides in the SYS schema.

Related Topics

Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection
 This tutorial demonstrates how to incorporate Oracle Virtual Private Database protection
 with a transparent sensitive data protection policy.

15.6.4.2 Creating the Transparent Sensitive Data Protection Policy

You can create a transparent sensitive data protection policy that uses a partial number data type-based Data Redaction policy.

Example 15-1 shows how to create this type of policy.

• To create the policy, use the DBMS_TSDP_PROTECT.ADD_POLICY procedure, as shown in Example 15-1.

Example 15-1 Creating a Transparent Sensitive Data Protection Policy

```
DECLARE
  redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
```



```
redact_feature_options ('expression') :=
    'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''APPUSER''';
redact_feature_options ('function_type') := 'DBMS_REDACT.PARTIAL';
redact_feature_options ('function_parameters') := '0,1,6';
policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
policy_conditions(DBMS_TSDP_PROTECT.LENGTH) := '16';
DBMS_TSDP_PROTECT.ADD_POLICY ('redact_partial_cc',
    DBMS_TSDP_PROTECT.REDACT,redact_feature_options,
    policy_conditions);
END;
//
```

In this example:

- redact_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS creates the variable redact_feature_options, which uses the FEATURE_OPTIONS data type. See Setting the Oracle Data Redaction or Virtual Private Database Feature Options for more information.
- policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS creates the variable policy_conditions, which uses the POLICY_CONDITIONS data type. See Setting Conditions for the Transparent Sensitive Data Protection Policy for more information.
- redact_feature_options lines (3) write the Data Redaction policy settings to the redact_feature_option variable. This example applies the Data Redaction policy to the user APPUSER and defines the policy as a partial data redaction for number data types. See Oracle Database Advanced Security Guide for information about how the function parameters parameter works for this case.
- policy_conditions lines (2) write the TSDP policy conditions to the policy_conditions variable (that is, the data type and length) for the protected NUMBER data type column.
- DBMS_TSDP_PROTECT.ADD_POLICY executes the DBMS_TSDP_PROTECT.ADD_POLICY procedure,
 which creates the redact_partial_cc TSDP policy. See Specifying the
 DBMS_TSDP_PROTECT.ADD_POLICY Procedure for more information.

If you want to see an example of a similar policy for VPD, see Step 4: Create and Enable a Transparent Sensitive Data Protection Policy.

15.6.4.3 Setting the Oracle Data Redaction or Virtual Private Database Feature Options

The TSDP feature options describe the Oracle Data Redaction or Virtual Private Database settings to use for the transparent sensitive data protection policy.

• For Data Redaction, define the feature options by using the name redact_feature_options variable and for the type, you must use the type DBMS_TSDP_PROTECT.FEATURE_OPTIONS, which is an associative array of the data type VARCHAR2 (TSDP_PARAM_MAX). Initialize these options with the parameter-value pairs that correspond with the DBMS_REDACT.ADD_POLICY parameters.

For example, to specify a TSDP policy that uses partial Data Redaction, Example 15-1 shows the following parameter-value pair:

```
redact feature options ('function type') := 'DBMS REDACT.PARTIAL';
```

For a partial Data Redaction policy that uses a number data type for the protected column, Example 15-1 specifies the following additional parameter-value pairs:

```
redact_feature_options ('expression') := 'expression';
redact_feature_options ('function_parameters') := 'values';
```

Similarly, for Virtual Private Database, you use the <code>vpd_feature_options</code> variable to define the VPD feature options. For example:

```
vpd feature options ('statement types') := 'SELECT, INSERT, UPDATE, DELETE';
```

See Also:

- Oracle Database Advanced Security Guide for more information about Data Redaction policy creation parameters
- DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies for more information about available VPD parameters

15.6.4.4 Setting Conditions for the Transparent Sensitive Data Protection Policy

Optionally, you can specify conditions for the transparent sensitive data protection policy.

However, if you choose to omit conditions, you still must include the following line in the DECLARE variables. (In this case, the default value for policy_conditions is an empty associative array.)

```
policy conditions SYS.DBMS TSDP PROTECT.POLICY CONDITIONS;
```

• To define the conditions, use the name <code>policy_conditions</code> for the variable and for the type, use type <code>DBMS_TSDP_PROTECT.POLICY_CONDITIONS</code>, which is an associative array of the data type <code>VARCHAR2(TSDP_PARAM_MAX)</code>. Ensure that no two conditions are satisfied by a single target sensitive column. The target column's properties should satisfy all the condition properties for the corresponding <code>DBMS_TSDP_PROTECT.FEATURE_OPTIONS</code> settings to be applied on the column

Example 15-1 shows the policy conditions as follows:

```
policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
policy_conditions(DBMS_TSDP_PROTECT.LENGTH) := '16';
```

Optionally, you can specify one or more of the following keys for the POLICY_CONDITIONS settings:

- DBMS TSDP PROTECT.DATATYPE enables you to specify a data type.
- DBMS_TSDP_PROTECT.LENGTH enables you to specify a data type length for the DBMS_TSDP_PROTECT.DATATYPE key.
- DBMS_TSDP_PROTECT.PARENT_SCHEMA enables you to restrict the policy to a specific schema. If you omit this setting, then the policy applies to all schemas in the database.
- DBMS_TSDP_PROTECT.PARENT_TABLE enables you to restrict the policy to a table specified by the DBMS_TSDP_PROTECT.PARENT_SCHEMA key. If you omit this setting, then the policy applies to all tables within the specified schema.



15.6.4.5 Specifying the DBMS TSDP PROTECT.ADD POLICY Procedure

The DBMS_TSDP_PROTECT.ADD_POLICY procedure names the TSDP policy and executes the FEATURE OPTIONS and POLICY CONDITIONS settings.

In the policy, the <code>redact_feature_options</code> and the <code>policy_conditions</code> settings work together: When the policy is enabled (using any of the <code>DBMS_TSDP_PROTECT.ENABLE_PROTECTION*</code> procedures) on the target object, then the <code>redact_feature_options</code> settings apply only if the corresponding <code>policy_condition</code> settings are satisfied. You must enter the following parameters:

- To specify a procedure that names the transparent sensitive data protection policy and executes the necessary settings, include the following parameters:
 - policy_name creates a name for the TSDP policy. The name that you enter is stored in the database using the case sensitivity that you used when you created it. For example, if you had entered redact_partial_cc, then the database stores it as redact_partial_cc, not redact_partial_cc.
 - security_feature refers to the security feature the TSDP policy will use. Enter DBMS TSDP PROTECT.REDACT to specify Oracle Data Redaction.
 - policy_enable_options refers to the variable that you defined for the DBMS TSDP PROTECT.FEATURE OPTIONS type.
 - policy_apply_condition refers to the variable that you defined for the DBMS TSDP PROTECT.POLICY CONDITIONS type.

Example 15-1 shows the policy set as follows:

```
DBMS_TSDP_PROTECT.ADD_POLICY('redact_partial_cc', DBMS_TSDP_PROTECT.REDACT, redact_feature_options, policy_conditions);
```

15.6.5 Step 5: Associate the Policy with a Sensitive Type

The DBMS_TSDP_PROTECT.ASSOCIATE_POLICY procedure associates a TSDP policy with a sensitive type.

1. Find the sensitive type that you want to use.

For example, to find a list of all sensitive types:

Run the DBMS_TSDP_PROTECT.ASSOCIATE_POLICY procedure to associate the policy with a sensitive column type.

For example:



15.6.6 Step 6: Enable the Transparent Sensitive Data Protection Policy

You can enable the TSDP policy for the current database in a protected source, a specific table column, or a specific column type.

- Enabling Protection for the Current Database in a Protected Source
 You can enable transparent sensitive data protection for the current database in a protected source.
- Enabling Protection for a Specific Table Column
 You can enable transparent sensitive data protection for a specific column in a table.
- Enabling Protection for a Specific Column Type
 You can enable transparent sensitive data protection for a specific column type, such as all columns that use the VARCHAR2 data type.

15.6.6.1 Enabling Protection for the Current Database in a Protected Source

You can enable transparent sensitive data protection for the current database in a protected source.

If you must disable the protection, then you can run the DBMS TSDP PROTECT.DISABLE PROTECTION SOURCE procedure.

Run the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_SOURCE procedure to enable this type of
protection.

For example, to enable transparent sensitive data protection policies for the <code>orders_db</code> database.

```
BEGIN

DBMS_TSDP_PROTECT.ENABLE_PROTECTION_SOURCE(
  discovery_source => 'orders_db');
END;
//
```

15.6.6.2 Enabling Protection for a Specific Table Column

You can enable transparent sensitive data protection for a specific column in a table.

Remember that you can enable only one policy per table. If you must disable the protection, then you can run the <code>DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN</code> procedure.

Run the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN procedure to enable this type of
protection.

For example, to enable the transparent sensitive data protection policy <code>redact_partial_cc</code> for a specific table column:

```
BEGIN
DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN(
    schema_name => 'OE',
```

If an ORA-45622: warnings generated during policy enforcement error appears, then check the configuration of the policy. In this example, the <code>redact_partial_cc</code> policy is enabled on a column if this column is of the <code>NUMBER</code> data type and has a length of 16. Even though the <code>OE.CUST_CC.CREDIT_CARD</code> column is associated with the <code>redact_partial_cc</code> policy, the policy is not enabled if this column fails to satisfy the conditions (data type and length).

15.6.6.3 Enabling Protection for a Specific Column Type

You can enable transparent sensitive data protection for a specific column type, such as all columns that use the VARCHAR2 data type.

If you must disable the protection, then you can run the DBMS TSDP PROTECT.DISABLE PROTECTION TYPE procedure.

Run the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE procedure to enable this type of
protection.

For example, to enable transparent sensitive data protection for all columns that use the credit card num type sensitive type:

```
BEGIN

DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE(
   sensitive_type => 'credit_card_num_type');
END;
//
```

15.6.7 Step 7: Optionally, Export the Policy to Other Databases

You can export or import the policy to or from another database.

 To export or import the TSDP policy to or from another database, use Oracle Data Pump to perform a full export or import of the database that contains the policy.

Remember that the export and import operations apply to the entire database, not just the transparent sensitive data protection policy.

See Also:

- Oracle Database Utilities for information about using Oracle Data Pump
- Oracle Database Vault Administrator's Guide for information about using Oracle Data Pump in an Oracle Database Vault environment



15.7 Altering Transparent Sensitive Data Protection Policies

The DBMS_TSDP_PROTECT.ALTER_POLICY procedure can alter a TSDP policy.

When you alter a transparent data protection policy, you must define how the Data Redaction settings must change, and then apply these changes to the transparent sensitive data protection policy itself.

You can find a list of existing policies and their protection definitions by querying the DBA TSDP POLICY FEATURE data dictionary view.

• To alter a transparent sensitive data protection policy, use the DBMS TSDP PROTECT.ALTER POLICY procedure.

For example, to alter an existing transparent sensitive data protection policy:

```
DECLARE
  redact_feature_options SYS.DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
  policy_conditions SYS.DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
  BEGIN
  redact_feature_options ('expression') :=
    'SYS_CONTEXT(''USERENV'',''SESSION_ USER'') = ''APPUSER''';
  redact_feature_options ('function_type') := 'DBMS_REDACT.PARTIAL';
  redact_feature_options ('function_parameters') := '9,1,6';
  policy_conditions(DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
  policy_conditions(DBMS_TSDP_PROTECT.LENGTH) := '22';
  DBMS_TSDP_PROTECT.ALTER_POLICY ('redact_partial_cc',
    redact_feature_options, policy_conditions);
END;
//
```

In this example:

- redact_feature_options SYS.DBMS_TSDP_PROTECT.FEATURE_OPTIONS creates the variable redact feature options, which uses the FEATURE OPTIONS data type.
- policy_conditions SYS.DBMS_TSDP_PROTECT.POLICY_CONDITIONS creates the variable policy conditions, which uses the POLICY CONDITIONS data type.
- redact_feature_options ... redact_feature_optionswrites the Data Redaction policy settings to the redact_feature_option variable. This example applies the Data Redaction policy to the user APPUSER, defines the policy as a partial data redaction for number data types. See Oracle Database Advanced Security Guide for information about how the function parameters parameter works for this case.
- policy_conditions ... policy_conditions writes the TSDP policy conditions to the policy_conditions variable (that is, the data type and length) for the protected NUMBER data type column.
- DBMS_TSDP_PROTECT.ALTER_POLICY ... executes the DBMS_TSDP_PROTECT.ALTER_POLICY procedure, which alters the redact_partial_cc TSDP policy to use the definitions set in the redact_feature_options and policy_conditions variables.

15.8 Disabling Transparent Sensitive Data Protection Policies

The DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN procedure disables one or all TSDP policies.

1. Query the DBA_TSDP_POLICY_PROTECTION data dictionary view to find the protected columns and their associated transparent sensitive data protection policies.

For example:

2. Run the DBMS_TSDP_PROTECT.DISABLE_PROTECTION_COLUMN procedure.

For example, to disable the <code>redact_partial_cc</code> policy on the <code>CREDIT_CARD</code> column of the <code>CUST_CC</code> table:

You can use the % wildcard in this procedure to specify multiple items. For example, to disable protection for any columns that begin with CREDIT, you could enter the following:

To disable all transparent sensitive data protection policies for a table, you can omit the policy parameter. For example:

15.9 Dropping Transparent Sensitive Data Protection Policies

You can drop an entire TSDP policy or a condition-enable-options combination from the policy.

If the policy only has one condition-enable-options combination, then Oracle Database drops the entire policy. You do not need to disable a policy before dropping it, but you do need to drop its associated sensitive column first, then its sensitive type.

 Query the POLICY_NAME column of the DBA_TSDP_POLICY_FEATURE data dictionary view to find the policy that you want to drop.

```
SELECT POLICY NAME FROM DBA TSDP POLICY FEATURE;
```



```
POLICY_NAME
-----
redact_partial_cc
```

Remember that you must be granted the <code>SELECT_CATALOG_ROLE</code> role to query the transparent sensitive data protection data dictionary views.

2. Find the sensitive column that is associated with this policy.

For example:

3. Drop this sensitive column.

For example:

4. Find the sensitive type that is associated with this policy.

For example:

5. Drop this sensitive type.

For example:

```
BEGIN
   DBMS_TSDP_MANAGE.DROP_SENSITIVE_TYPE ( sensitive_type =>
'credit_card_num_type');END;
//
```

6. Run the DBMS TSDP PROTECT. DROP POLICY procedure to drop the policy.

For example, to completely drop the policy:

```
BEGIN
DBMS_TSDP_PROTECT.DROP_POLICY(
   policy_name => 'redact_partial_cc');
END;
/
```

To drop the default condition-enable options combination from the policy:

```
DECLARE
    policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
    DBMS TSDP PROTECT.DROP POLICY ('redact partial cc', policy conditions);
```



```
END;
```

To drop the default condition-enable options combination from the policy based on a specific condition:

```
DECLARE
    policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
    policy_conditions (DBMS_TSDP_PROTECT.DATATYPE) := 'NUMBER';
    DBMS_TSDP_PROTECT.DROP_POLICY ('redact_partial_cc', policy_conditions);
END;
//
```

15.10 Using the Predefined REDACT_AUDIT Policy to Mask Bind Values

The predefined REDACT_AUDIT policy masks bind values, which can appear in trace files when an event is set.

About the REDACT AUDIT Policy

The predefined REDACT_AUDIT transparent sensitive data protection policy masks bind values.

Variables Associated with Sensitive Columns

Bind variables affect the use of sensitive columns with conditions, SELECT items, and INSERT or UPDATE operations.

How Bind Variables on Sensitive Columns Behave with Views

A bind variable that appears in a query on a view is considered sensitive if the view column references a sensitive column.

Disabling the REDACT_AUDIT Policy

By default, the REDACT AUDIT policy is enabled for all sensitive columns.

Enabling the REDACT_AUDIT Policy

You can enable the REDACT_AUDIT policy for a specific sensitive column or for all columns in the database.

15.10.1 About the REDACT AUDIT Policy

The predefined REDACT AUDIT transparent sensitive data protection policy masks bind values.

The bind values of the bind variables that are used in SQL statements can appear in audit records when auditing is configured. Similarly, bind values can appear in trace files when the appropriate event is set. Bind values can also appear when you query the V\$SQL_BIND_DATA dynamic view.

The REDACT_AUDIT transparent sensitive data protection policy displays the data as an asterisk (*) in audit records, trace files, and in V\$SQL_BIND_DATA view queries. By default the REDACT_AUDIT policy is associated with every sensitive type in the database. When you identify a column as sensitive, by default, the REDACT_AUDIT policy is enabled for it.

You can disable and enable the REDACT AUDIT policy, but you cannot alter or drop it.



15.10.2 Variables Associated with Sensitive Columns

Bind variables affect the use of sensitive columns with conditions, SELECT items, and INSERT or UPDATE operations.

- About Variables Associated with Sensitive Columns
 You can associate variables with sensitive columns in TSDP policies.
- Bind Variables and Sensitive Columns in the Expressions of Conditions
 You can include sensitive columns in SQL queries that have WHERE clauses.
- A Bind Variable and a Sensitive Column Appearing in the Same SELECT Item
 If a column in a SELECT item is sensitive, then all the binds in the SELECT item are
 considered sensitive.
- Bind Variables in Expressions Assigned to Sensitive Columns in INSERT or UPDATE Operations

You can assign multiple bind variables to different columns in one INSERT or UPDATE statement.

15.10.2.1 About Variables Associated with Sensitive Columns

You can associate variables with sensitive columns in TSDP policies.

A bind variable can be considered to be sensitive or "associated" with a sensitive column if the bind variable occurs in the same comparison condition as a sensitive column, if it occurs in a SELECT statement alongside a sensitive column, or if it occurs in an INSERT or UPDATE operation that involves a sensitive column.

15.10.2.2 Bind Variables and Sensitive Columns in the Expressions of Conditions

You can include sensitive columns in SQL queries that have WHERE clauses.

A SQL query that contains a WHERE clause can include sensitive columns and bind variables for use with comparison operators such as =, IS, IS NOT, LIKE, BETWEEN, and IN, as well as in subqueries.

In the following comparison query, the bind value in VAR1 is masked because VAR1 and the sensitive column SALARY appear in the expression that is compared using the comparison condition >.

```
SELECT EMPLOYEE ID FROM HR.EMPLOYEES WHERE SALARY > :VAR1;
```

In the next query, the bind values in VAR1 and VAR2 are masked because VAR1, VAR2, and the sensitive column SALARY appear in the expression that uses the comparison equality condition =.

```
SELECT EMPLOYEE ID FROM HR.EMPLOYEES WHERE SALARY + : VAR1 = TO NUMBER(:VAR2, '9G999D99');
```

For floating point conditions, the sensitive column and the bind variable appear in the expression that is evaluated. In the following example, the bind value in VAR1 is masked because VAR1 and the sensitive column SALARY appear in the expression for the IS NOT NAN condition.

```
SELECT COUNT() FROM HR.EMPLOYEES WHERE (SALARY * : VAR1) IS NOT NAN;
```



In pattern matching conditions, the sensitive column and the bind variable appear as arguments. In the following example, the bind value in VAR1 is masked because VAR1 and the sensitive column LAST NAME are the arguments for the LIKE condition.

```
SELECT LAST_NAME FROM HR.EMPLOYEES WHERE LAST_NAME LIKE :VAR1;
```

For Between conditions, the sensitive column and the bind variable appear in the expressions that are arguments. In the following example, bind values in VAR1 and VAR2 are masked because VAR1, VAR2, and SALARY appear in expressions that are arguments to the Between condition.

```
SELECT EMPLOYEE ID FROM HR.EMPLOYEES WHERE SALARY BETWEEN : VAR1 AND : VAR2;
```

In the next example, the sensitive column and the bind variable are the arguments of the IN condition. Here, the bind values in VAR1 and VAR2 are masked because VAR1, VAR2, and the sensitive column SALARY appear as arguments to the IN condition.

```
SELECT COUNT() FROM HR.EMPLOYEES WHERE SALARY IN (:VAR1,:VAR2);
```

When a condition has a nested subquery as an argument, the bind variables and sensitive columns that appear in the nested subquery are not considered to be associated with the condition. In the following query, the sensitive column SALARY and the subquery are expressions for the greater-than condition >.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES WHERE SALARY > (SELECT SALARY FROM HR.EMPLOYEES WHERE MANAGER_ID = :VAR1);
```

However, variable VAR1 is associated with column MANAGER_ID as variable VAR1 and MANAGER_ID appears in expressions being compared using the condition =. Because MANAGER_ID is not a sensitive column, variable VAR1 is not considered sensitive. The variable VAR1 is not considered to be associated with the sensitive column SALARY.

In the case of the logical conditions, model conditions, multiset conditions, XML conditions, compound conditions, IS OF type conditions, and EXISTS conditions, there can be no cases where a bind variable and a sensitive column are associated with each other. This is due to the structure or the nature of these conditions.

15.10.2.3 A Bind Variable and a Sensitive Column Appearing in the Same SELECT Item

If a column in a SELECT item is sensitive, then all the binds in the SELECT item are considered sensitive.

For example, assume that HR.EMPLOYEES.SALARY and HR.EMPLOYEES.COMMISSION_PCT are sensitive columns. In the following query, the bind variable VAR1 is considered sensitive because it appears in the same SELECT item as the sensitive column SALARY, so its bind value is masked.

```
SELECT (SALARY * : VAR1) AS BONUS AS FROM HR.EMPLOYEES WHERE EMPLOYEE ID = :VAR2;
```

In the next example, the bind variable <code>VAR1</code> is considered sensitive because it appears in the same <code>SELECT</code> item as <code>SALARY</code>. <code>VAR2</code> is considered sensitive because it appears in the same <code>SELECT</code> item as the sensitive column <code>COMMISSION</code> <code>PCT</code>.

```
SELECT (SALARY * :VAR1), (COMMISSION_PCT * :VAR2), (EMPNO + :VAR3) AS BONUS AS FROM PAYROLL.ACCOUNT;
```



15.10.2.4 Bind Variables in Expressions Assigned to Sensitive Columns in INSERT or UPDATE Operations

You can assign multiple bind variables to different columns in one INSERT or UPDATE statement.

Consider the following INSERT statement:

```
INSERT INTO PAYROLL.ACCOUNT (ACCOUNT NUM, SALARY) VALUES (:VAR1 * :VAR2 , :VAR3);
```

In this INSERT statement, the following takes place:

- The bind variables VAR1 and VAR2 appear in the expression (:VAR1 * :VAR2), which is
 assigned to the sensitive column ACCOUNT NUM.
- The bind variable VAR3 is assigned to sensitive column SALARY.

Consider the following UPDATE statement:

```
UPDATE PAYROLL.ACCOUNT SET ACCOUNT NUM = :VAR1, SALARY = :VAR2;
```

In this update statement, the following takes place:

- The bind variable VAR1 is assigned to sensitive column ACCOUNT NUM.
- The bind variable VAR2 is assigned to sensitive column SALARY.

15.10.3 How Bind Variables on Sensitive Columns Behave with Views

A bind variable that appears in a query on a view is considered sensitive if the view column references a sensitive column.

For example, suppose you identify the SALARY column in the HR.EMPLOYEES table as sensitive. Then you create the view EMPLOYEES_VIEW as follows:

```
CREATE OR REPLACE VIEW HR.EMPLOYEES VIEW AS SELECT * FROM HR.EMPLOYEES;
```

When a user references the SALARY column from this view in a SQL statement, any bind variable that has been associated with the SALARY column is considered sensitive and its bind value then masked.

```
SELECT EMPLOYEE_ID FROM HR.EMPLOYEES_VIEW WHERE SALARY = :VAR1;
```

In this case, the bind variable VAR1 is masked because it is associated with the HR.EMPLOYEES_VIEW.SALARY column, which references the sensitive column HR.EMPLOYEES.SALARY.

15.10.4 Disabling the REDACT_AUDIT Policy

By default, the REDACT AUDIT policy is enabled for all sensitive columns.

You can disable it for a specific sensitive column or all sensitive columns, and when needed, re-enable it. Remember that you cannot alter or delete the \mathtt{REDACT} AUDIT policy.

 To disable the REDACT_AUDIT policy, use the DBMS TSDP PROTECT.DISABLE PROTECTION COLUMN procedure.

For example, to disable the REDACT AUDIT policy for the SALARY column of HR.EMPLOYEES:



The following example shows how to disable the REDACT_AUDIT policy for all sensitive columns in the current database.

15.10.5 Enabling the REDACT_AUDIT Policy

You can enable the REDACT_AUDIT policy for a specific sensitive column or for all columns in the database.

• To enable the REDACT_AUDIT policy, use the DBMS TSDP PROTECT.ENABLE PROTECTION COLUMN procedure.

For example, to re-enable the REDACT AUDIT policy for the SALARY column of HR.EMPLOYEES:

The following example shows how to enable the $\texttt{REDACT_AUDIT}$ policy for all sensitive columns in the current database.

```
BEGIN
DBMS_TSDP_PROTECT.ENABLE_PROTECTION_COLUMN(
  policy => 'REDACT_AUDIT');
END;
//
```

15.11 Transparent Sensitive Data Protection Policies with Data Redaction

Oracle Data Redaction features work with transparent sensitive data protection policies.

The Data Redaction function types, function parameters, and expressions can be used in the TSDP policy definition. For example, you can set the enable the TSDP policy to use FULL or PARTIAL data redaction. This chapter uses Data Redaction for examples of managing TSDP policies.

See Also:

- Creating Transparent Sensitive Data Protection Policies for an example of how to create TSDP policies that use Data Redaction function types
- Oracle Database Advanced Security Guide for details about Oracle Data Redaction

15.12 Using Transparent Sensitive Data Protection Policies with Oracle VPD Policies

You can combine protections from TSDP and Oracle Virtual Private Database into one policy.

- About Using TSDP Policies with Oracle Virtual Private Database Policies
 To incorporate Oracle Virtual Private Database protection with transparent sensitive data protection policies, you must use the DBMS TSDP PROTECT and DBMS RLS packages.
- DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies
 Oracle Database provides a set of parameters for fine-tuning the behavior of TSDP
 policies.
- Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection
 This tutorial demonstrates how to incorporate Oracle Virtual Private Database protection
 with a transparent sensitive data protection policy.

15.12.1 About Using TSDP Policies with Oracle Virtual Private Database Policies

To incorporate Oracle Virtual Private Database protection with transparent sensitive data protection policies, you must use the DBMS TSDP PROTECT and DBMS RLS packages.

This feature works as follows:

- 1. You create a VPD policy function with a suitable predicate. Later on, when you create the TSDP policy, you will refer to this VPD policy function by using the policy_function setting of the DBMS_RLS.ADD_POLICY procedure for the feature_options parameter of the DBMS_TSDP_PROTECT.ADD_POLICY procedure.
- 2. You create a TSDP policy with the necessary VPD settings similar to the VPD policy function.
 - The TSDP policy uses parameter settings from the <code>DBMS_RLS.ADD_POLICY</code> procedure to provide VPD protection. Table 15-1 lists these parameters. Be aware that parameters from the <code>DBMS_RLS.ADD_GROUPED_POLICY</code> policy are not supported.
- 3. You associate the TSDP policy with the necessary sensitive types by using the DBMS_TSDP_PROTECT.ASSOCIATE_POLICY procedure.
- **4.** You then enable TSDP protection by using any of the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_* procedures.
- 5. You enable the TSDP policy. At this point, Oracle Database creates an internal VPD policy that uses the function that you created in Step 1.



The name of the internal policy begins with <code>ORA\$VPD</code> followed by an identifier (for example, <code>ORA\$VPD_6J6L3RSJSN2VAN0XF</code>). You can find this policy by querying the <code>POLICY_NAME</code> column of the <code>DBA_POLICIES</code> data dictionary view.

- When users query the table, the output for the column is based on both the VPD protections and the TSDP policy that are now in place.
- 7. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database automatically drops the internal VPD policy, because it is no longer needed. If you reenable the TSDP policy, then the internal VPD policy is recreated.

Related Topics

Function to Generate the Dynamic WHERE Clause
 The Oracle Virtual Private Database (VPD) function defines the restrictions that you want to enforce.

15.12.2 DBMS_RLS.ADD_POLICY Parameters That Are Used for TSDP Policies

Oracle Database provides a set of parameters for fine-tuning the behavior of TSDP policies.

Table 15-1 describes the <code>DBMS_RLS.ADD_POLICY</code> parameters that are permissible in the <code>FEATURE_OPTIONS</code> parameter when you use the <code>DBMS_TSDP_PROTECT.ADD_POLICY</code> or <code>DBMS_TSDP_PROTECT.ALTER_POLICY</code> procedure.

Table 15-1 DBMS_RLS.ADD_POLICY Parameters Used for TSDP Policies

Parameter	Description	Default
function_schema	Schema of the policy function (current default schema, if NULL). If no function_schema is specified, then the current user's schema is assumed.	NULL
policy_function	Name of a function that generates a predicate for the policy. If the function is defined within a package, then you must include the name of the package (for example, my_package.my_function).	NULL
statement_types	Statement types to which the policy applies. It can be any combination of INDEX, SELECT, INSERT, UPDATE, or DELETE. The default is to apply to most of these types except INDEX.	NULL
update_check	Optional argument for INSERT or UPDATE statement types. Setting update_check to TRUE sets Oracle Database to check the policy against the value after an INSERT or UPDATE operation.	FALSE
	The check applies only to the security relevant columns that are included in the policy definition. In other words, the INSERT or UPDATE operation will fail only if the security relevant column that is defined in the policy is added or updated in the INSERT or UPDATE statement.	
static_policy	If you set this value to TRUE, then Oracle Database assumes that the policy function for the static policy produces the same predicate string for anyone accessing the object, except for SYS or the privileged user who has the EXEMPT ACCESS POLICY privilege.	FALSE



Parameter	Description	Default	
policy_type	Default is NULL, which means policy_type is decided by the value of the static_policy parameter. Specifying any of these policy types overrides the value of static_policy.	NULL	
long_predicate	Default is FALSE, which means the policy function can return a predicate with a length of up to 4000 bytes. TRUE means the predicate text string length can be up to 32K bytes. Policies existing before the availability of the long predicate parameter retain a 32K limit.		
sec_relevant_cols_opt	If you specify this parameter, then transparent sensitive data protection inputs the sensitive column on which the protection is enabled to the sec_relevant_cols parameter of the DBMS_RLS.ADD_POLICY procedure.		
	Allowed values are for sec_relevant_cols_opt are as follows:		
	 NULL enables the filtering defined with sec_relevant_cols to take effect. 		
	DBMS_RLS.ALL_ROWS displays all rows, but with sensitive column values, which are filtered by the sec_relevant_cols parameter, they display as NULL.		

Table 15-1 (Cont.) DBMS_RLS.ADD_POLICY Parameters Used for TSDP Policies

Related Topics

Attaching a Policy to a Database Table, View, or Synonym
 The DBMS RLS PL/SQL package can attach a policy to a table, view, or synonym.

15.12.3 Tutorial: Creating a TSDP Policy That Uses Virtual Private Database Protection

This tutorial demonstrates how to incorporate Oracle Virtual Private Database protection with a transparent sensitive data protection policy.

- Step 1: Create the hr_appuser User Account
 First, you must create a sample user account and then grant this user the appropriate
 privileges.
- Step 2: Identify the Sensitive Columns
 As the sample user tsdp_admin, you are ready to identify sensitive columns to protect.
- Step 3: Create an Oracle Virtual Private Database Function
 TSDP will associate the Oracle VPD policy function with the VPD policy that is
 automatically created when the TSDP policy is enabled.
- Step 4: Create and Enable a Transparent Sensitive Data Protection Policy
 After you have created the VPD policy function, you can associate it with a transparent
 sensitive data protection policy.
- Step 5: Test the Transparent Sensitive Data Protection Policy
 Now, you are ready to test the transparent sensitive data protection policy.
- Step 6: Remove the Components of This Tutorial If you no longer need the components of this tutorial, then you can remove them.



15.12.3.1 Step 1: Create the hr_appuser User Account

First, you must create a sample user account and then grant this user the appropriate privileges.

1. Log into the database instance as user SYS with the SYSDBA administrative privilege.

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

2. If you are using a multitenant environment, then connect to the appropriate pluggable database (PDB).

For example:

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

To find the available PDBs, run the show pdbs command. To check the current PDB, run the show con name command.

Create the following user accounts:

```
GRANT CREATE SESSION TO hr_appuser IDENTIFIED BY password; GRANT CREATE SESSION TO tsdp_admin IDENTIFIED BY password;
```

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

4. Grant user tsdp admin the following privileges:

```
GRANT CREATE PROCEDURE TO tsdp_admin;
GRANT EXECUTE ON DBMS_TSDP_MANAGE TO tsdp_admin;
GRANT EXECUTE ON DBMS_TSDP_PROTECT TO tsdp_admin;
GRANT EXECUTE ON DBMS_RLS to tsdp_admin;
```

Connect as user SCOTT.

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

6. Grant the hr_appuser the READ object privilege for the EMP table.

```
GRANT READ ON EMP TO hr_appuser;
```

15.12.3.2 Step 2: Identify the Sensitive Columns

As the sample user tsdp_admin, you are ready to identify sensitive columns to protect.

1. Connect as user tsdp admin.

```
CONNECT tsdp_admin -- Or, CONNECT tsdb_admin@hrpdb
Enter password: password
```

2. Create the salary type sensitive type:

```
BEGIN
DBMS_TSDP_MANAGE.ADD_SENSITIVE_TYPE (
   sensitive_type => 'salary_type',
   user_comment => 'Type for SCOTT.EMP column');
END;
/
```

3. Associate the salary type sensitive type with the SCOTT.EMP table.

15.12.3.3 Step 3: Create an Oracle Virtual Private Database Function

TSDP will associate the Oracle VPD policy function with the VPD policy that is automatically created when the TSDP policy is enabled.

 To create the VPD policy function, use the CREATE OR REPLACE FUNCTION procedure, as follows:

```
CREATE OR REPLACE FUNCTION vpd_function (
   v_schema IN VARCHAR2,
   v_objname IN VARCHAR2)
RETURN VARCHAR2 AS
BEGIN
RETURN 'SYS_CONTEXT(''USERENV'',''SESSION_USER'') = ''HR_APPUSER''';
END vpd_function;
//
```

15.12.3.4 Step 4: Create and Enable a Transparent Sensitive Data Protection Policy

After you have created the VPD policy function, you can associate it with a transparent sensitive data protection policy.

Create the Transparent Sensitive Data Protection policy.

```
vpd_feature_options DBMS_TSDP_PROTECT.FEATURE_OPTIONS;
policy_conditions DBMS_TSDP_PROTECT.POLICY_CONDITIONS;
BEGIN
vpd_feature_options ('policy_function') := 'vpd_function';
vpd_feature_options ('sec_relevant_cols_opt') := 'DBMS_RLS.ALL_ROWS';
dbms_tsdp_protect.add_policy('tsdp_vpd', DBMS_TSDP_PROTECT.VPD,
vpd_feature_options, policy_conditions);
END;
//
```

In this example, the <code>vpd_feature_options</code> parameter refers to the <code>sec_relevant_cols_opt</code> parameter from the <code>DBMS_RLS.ADD_POLICY</code> procedure. When the TSDP policy is enabled, the VPD policy that is automatically created will have its <code>sec_relevant_cols</code> parameter (of <code>DBMS_RLS.ADD_POLICY</code>) set to the name of the sensitive column on which TSDP enables the VPD policy. If you had not used the <code>sec_relevant_cols_opt</code> parameter, then TSDP would not have used the <code>DBMS_RLS.ADD_POLICY</code> <code>sec_relevant_cols_opt</code> parameter.

2. Associate the tsdp vpd1 TSDP policy with the salary type sensitive type.



```
END;
```

3. Enable protection to enforce the Virtual Private Database policy on all columns identified as SALARY TYPE:

```
BEGIN
DBMS_TSDP_PROTECT.ENABLE_PROTECTION_TYPE(
   sensitive_type => 'salary_type');
END;
//
```

15.12.3.5 Step 5: Test the Transparent Sensitive Data Protection Policy

Now, you are ready to test the transparent sensitive data protection policy.

Connect as user hr_appuser.

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

2. Query the SCOTT.EMP table as follows:

```
SELECT SAL, COMM, EMPNO FROM SCOTT.EMP;
```

The following output appears:

SA	L CO	MM	EMPNO
80	0		7369
160	0 3	00	7499
125	0 5	00	7521
297	5		7566
125	0 14	00	7654
285	0		7698
245	0		7782
300	0		7788
500	0		7839
150	0	0	7844
110	0		7876
95	0		7900
300	0		7902
130	0		7934
14 row	s sele	cted.	

The <code>vpd_function</code> function enables user <code>hr_appuser</code> to see the salaries in the <code>SAL</code> column of the <code>EMP</code> table.

3. Connect as user SCOTT and then perform the same query.

```
CONNECT SCOTT -- Or, CONNECT SCOTT@hrpdb
Enter password: password
SELECT SAL, COMM, EMPNO FROM SCOTT.EMP;
```

The following output appears:

SAL	COMM	EMPNO
		7369
	300	7499
	500	7521
		7566
	1400	7654



```
7698
7782
7788
7839
0 7844
7876
7900
7902
7934
14 rows selected.
```

Even though SCOTT owns the EMP table, the $vpd_function$ function prevents him from seeing the salaries in the SAL column of this table

15.12.3.6 Step 6: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

1. Connect as user tsdp admin.

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

2. Execute the following statements in the order shown.

3. Connect as user SYSTEM.

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

4. Drop the tsdp admin and hr appuser accounts.

```
DROP USER tsdp_admin CASCADE;
DROP USER hr_appuser
```

15.13 Using Transparent Sensitive Data Protection Policies with Unified Auditing

The transparent sensitive data protection and unified auditing procedures can combine the protections of these two features.

- About Using TSDP Policies with Unified Audit Policies
 You can configure transparent sensitive data protection policies to audit object actions using unified auditing.
- Unified Audit Policy Settings That Are Used with TSDP Policies
 Audit policy settings can be used in the POLICY_ENABLE_OPTIONS parameter for the DBMS_TSDP_PROTECT.ADD_POLICY or DBMS_TSDP_PROTECT.ALTER_POLICY procedure.

15.13.1 About Using TSDP Policies with Unified Audit Policies

You can configure transparent sensitive data protection policies to audit object actions using unified auditing.

The DBMS_TSDP_PROTECT.ADD_POLICY and DBMS_TSDP_PROTECT.ALTER_POLICY procedures enable you to specify settings from the CREATE AUDIT POLICY, ALTER AUDIT POLICY, AUDIT POLICY, and COMMENT SQL statements. The TSDP policy enables the creation of action auditoptions for object-specific options in the policy, such as INSERT or DELETE operations. Systemwide audit options are not supported. Therefore, the audited object type is always TABLE. Only standard actions (such as INSERT) are permitted. Component actions, such as creating policies for Oracle Label Security or other Oracle Database features, are not supported.

This feature works as follows:

- 1. You create a TSDP policy with the necessary unified audit settings.
 - The TSDP policy uses parameter settings from the CREATE AUDIT POLICY, AUDIT POLICY, and COMMENT statements. Unified Audit Policy Settings That Are Used with TSDP Policies lists these settings.
- 2. You associate the TSDP policy with the necessary sensitive types by using the DBMS TSDP PROTECT.ASSOCIATE POLICY procedure.
- You then enable TSDP protection by using any of the DBMS TSDP PROTECT.ENABLE PROTECTION * procedures.
- 4. You enable the TSDP policy. As part of the TSDP policy enablement process, Oracle Database internally creates a unified audit policy and then enables it on the list of target users and roles that you specified in the DBMS_TSDP_PROTECT.ADD_POLICY procedure from Step 1.
 - The name of the internal policy begins with <code>ORA\$UNIFIED_AUDIT_</code> followed by a random alpha-numeric string (for example, <code>ORA\$UNIFIED_AUDIT_6J6L3RSJSN2VANOXF</code>). You can find this policy by querying the <code>POLICY_NAME</code> column of the <code>AUDIT_UNIFIED_POLICIES</code> data dictionary view. To find the names of the users and roles on which this internally created TSDP unified audit policy is enforced, query the <code>AUDIT_UNIFIED_ENABLED_POLICIES</code> view.
- 5. When users try to perform an action on the table that is being protected by the TSDP policy, then based on the TSDP unified audit policy configuration, a unified audit record is written to the unified audit trail for this object access. You can then query the UNIFIED_AUDIT_TRAIL view to see the unified audit record that was created because of the TSDP unified audit policy enforcement.
- 6. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database automatically disables and then drops the internal policy, because it is no longer necessary. (A unified audit policy must be disabled before it can be dropped.) If you re-enable the TSDP policy, then the internal policy is recreated.



15.13.2 Unified Audit Policy Settings That Are Used with TSDP Policies

Audit policy settings can be used in the <code>POLICY_ENABLE_OPTIONS</code> parameter for the <code>DBMS_TSDP_PROTECT.ADD_POLICY</code> or <code>DBMS_TSDP_PROTECT.ALTER_POLICY</code> procedure.

These audit policy settings are from the AUDIT, CREATE AUDIT POLICY, and ALTER AUDIT POLICY statements.

The following table describes these settings.

Table 15-2 Unified Audit Policy Settings Used for TSDP Policies

Parameter	Description	Default
ACTION_AUDIT_OPTIONS	A string containing a comma-separated list of SQL actions.	ALL
	Valid actions are: ALTER, AUDIT, COMMENT, DELETE, FLASHBACK, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE	
	To configure the policy to audit all of these actions, specify the keyword ALL.	
AUDIT_CONDITION	SYS_CONTEXT (namespace, attribute) operation value-list	NULL
	In this syntax, operation can be any of the following operators: IN, $ $ NOT IN, =, <, >, or <>	
	If the audit condition contains a single quotation mark, then specify two single quotation marks instead of one, and enclose the SYS_CONTEXT in single quotations. For example:	
	<pre>'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'') = ''myclient'''</pre>	
EVALUATE_PER	Can be one of the following:	STATEMENT
	• STATEMENT	
	• SESSION	
	• INSTANCE	
ENTITY_NAME	A string that contains a comma-separated list of users or roles. If you omit this parameter, then the audit policy is enabled for all users.	NULL (that is, all database users)
ENABLE_OPTION	Applies only if the ENTITY_NAME parameter is used. It specifies if the ENTITY_NAME is a BY user list, an EXCEPT user list, or a BY USERS WITH GRANTED ROLES role list. Valid settings are:	ВУ
	• BY	
	• EXCEPT	
	BY USERS WITH GRANTED ROLES	
UNIFIED_AUDIT_POLICY_COMME	A string that describes the unified audit policy that will be created	NULL



15.14 Using Transparent Sensitive Data Protection Policies with Fine-Grained Auditing

The transparent sensitive data protection and fine-grained auditing procedures can combine the protections of these two features.

- About Using TSDP Policies with Fine-Grained Auditing
 You can configure a Transparent Sensitive Data Protection policy for fine-grained auditing.
- Fine-Grained Auditing Parameters That Are Used with TSDP Policies

 DBMS_FGA.ADD_POLICY settings can be used in the POLICY_ENABLE_OPTIONS parameter for the DBMS_TSDP_PROTECT.ADD_POLICY or DBMS_TSDP_PROTECT.ALTER_POLICY procedure.

15.14.1 About Using TSDP Policies with Fine-Grained Auditing

You can configure a Transparent Sensitive Data Protection policy for fine-grained auditing.

The DBMS_TSDP_PROTECT.ADD_POLICY and DBMS_TSDP_PROTECT.ALTER_POLICY procedures enable you to specify settings from the DBMS FGA.ADD POLICY procedure.

This feature works as follows:

- 1. You create a TSDP policy with the necessary fine-grained audit settings.
 - The TSDP policy uses parameter settings from the <code>DBMS_FGA.ADD_POLICY</code> procedure. Fine-Grained Auditing Parameters That Are Used with TSDP Policies lists these settings.
- 2. You associate the TSDP policy with the necessary sensitive types by using the DBMS TSDP PROTECT.ASSOCIATE POLICY procedure.
- 3. You then enable TSDP protection by using any of the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_* procedures.
- 4. You enable the TSDP policy. As part of the TSDP policy enablement process, Oracle Database internally creates a fine-grained audit policy that you specified in the DBMS_TSDP_PROTECT.ADD_POLICY procedure from Step 1.
 - The name of the internal policy begins with <code>ORA\$FGA_</code> followed by a random alpha-numeric string (for example, <code>ORA\$FGA_6J6L3RSJSN2VAN0XF</code>). You can find this policy by querying the <code>POLICY_NAME</code> column of the <code>DBA_POLICIES</code> data dictionary view.
- 5. When users try to perform an action on the table that is being protected by the TSDP policies, then based on the policy configuration, a fine-grained audit record is generated in the DBA_FGA_AUDIT_TRAIL data dictionary view for this object access.
- 6. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database automatically drops the internal policy, because it is no longer needed. If you reenable the TSDP policy, then the internal policy is recreated.

15.14.2 Fine-Grained Auditing Parameters That Are Used with TSDP Policies

DBMS_FGA.ADD_POLICY settings can be used in the POLICY_ENABLE_OPTIONS parameter for the DBMS_TSDP_PROTECT.ADD_POLICY or DBMS_TSDP_PROTECT.ALTER_POLICY procedure.

The following table describes these settings.



Table 15-3 Fine-Grained Audit Policy Settings Used for TSDP Policies

Parameter	Description	Default
audit_condition	Specifies a Boolean value to indicate a monitoring condition, using the following syntax:	NULL
	operator value	
	For example: < 1000	
handler_schema	Schema that contains the event handler. The default, $\mathtt{NULL},$ enables the current schema to be used.	NULL
handler_module	Function name of the event handler. Include the package name if necessary. This function is invoked only after the first row that matches the audit condition in the query is processed. If the procedure fails with an exception, then the user's SQL statement fails as well.	NULL
statement_types	You can specify one of the following statement types: INSERT, UPDATE, SELECT, or DELETE.	SELECT
audit_trail	If you have not yet migrated the database to full unified auditing, then use this setting to set the destination of the audit records: DB for the database or XML for XML records. This setting also specifies whether to populate the LSQLTEXT and LSQLBIND columns in the FGA_LOG\$ system table.	NULL
	If full unified auditing is enabled, then Oracle Database ignores this parameter and writes the audit records to the unified audit trail.	
object_schema	The schema that corresponds to the sensitive column	Schema that contains the sensitive column
object_name	The table that contains the sensitive column	The object (table or view) that contains the sensitive column
policy_name	A system-generated name for the internal fine- grained audit policy	Internal fine-grained audit policy system-generated name
audit_column	The sensitive column	The sensitive column
audit_column_opts	Determines whether to audit all or specific columns	DBMS_FGA.ANY_COL UMN
enable	Enable status for the TSDP policy; can be either ${\tt TRUE}$ or ${\tt FALSE}$	TRUE
policy_owner	User who invokes the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_* procedure	Current user

15.15 Using Transparent Sensitive Data Protection Policies with TDE Column Encryption

The TSDP procedures and Transparent Data Encryption column encryption statements can combine the protections of these two features.

- About Using TSDP Policies with TDE Column Encryption
 A TSDP policy can enable the encryption of columns that use Transparent Data Encryption.
- TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies
 The CREATE TABLE and ALTER TABLE statement ENCRYPT clause settings can be used in the
 POLICY_ENABLE_OPTIONS parameter for the DBMS_TSDP_PROTECT.ADD_POLICY or
 DBMS_TSDP_PROTECT.ALTER_POLICY procedure.

15.15.1 About Using TSDP Policies with TDE Column Encryption

A TSDP policy can enable the encryption of columns that use Transparent Data Encryption.

The DBMS_TSDP_PROTECT.ADD_POLICY and DBMS_TSDP_PROTECT.ALTER_POLICY procedures enable you to specify the ENCRYPT clause settings from the CREATE TABLE or ALTER TABLE statement.

This feature works as follows:

- You can create a TSDP policy by using the DBMS_TSDP_PROTECT.ADD_POLICY procedure. In the ADD_POLICY procedure, you can configure the policy for column encryption by setting the SECURITY_FEATURE parameter to DBMS_TSDP_PROTECT.COLUMN_ENCRYPTION. This setting enables encryption on the sensitive column when the TSDP policy is enabled on the object.
- 2. You create a TSDP policy with the necessary table encryption settings.
 - The TSDP policy uses parameter settings from the CREATE TABLE Or ALTER TABLE SQL statement. TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies lists these settings.
- 3. You associate the TSDP policy with the necessary sensitive types by using the DBMS_TSDP_PROTECT.ASSOCIATE_POLICY procedure.
- You then enable TSDP protection by using any of the DBMS_TSDP_PROTECT.ENABLE_PROTECTION_* procedures.
- You enable the TSDP policy. At this point, Oracle Database creates an internal TSDP policy that uses the encrypted table settings that you created earlier in this procedure.
 - The name of the internal policy begins with <code>ORA\$TDECE_</code> followed by a random alphanumeric string (for example, <code>ORA#TDECE_6J6L3RSJSN2VANOXF</code>). You can find this policy by querying the <code>TSDP_POLICY</code> column of <code>DBA_TSDP_POLICY_PROTECTION</code> view.
- 6. When users try to perform an action on the table that is being protected by the policies, the output for the column is based on both the TDE column protections and the TSDP policy that are now in place. You can check if the column has been encrypted after you enabled the TSDP policy by querying the ENCRYPTION_ALG column of the DBA_ENCRYPTED_COLUMNS view.
- 7. These protections remain in place until you disable the TSDP policy for this column. At that point, Oracle Database internally issues an ALTER TABLE statement on the table that contains the sensitive column, so that the sensitive column is decrypted. If you reenable the TSDP policy, then TSDP internally executes the ALTER TABLE statement with the ENCRYPT clause for the column.





It is possible to create two policies on the same column with each policy specifying a different encryption algorithm. In this case, the stronger of the two algorithms is enforced on the sensitive column.

15.15.2 TDE Column Encryption ENCRYPT Clause Settings Used with TSDP Policies

The CREATE TABLE and ALTER TABLE statement ENCRYPT clause settings can be used in the POLICY_ENABLE_OPTIONS parameter for the DBMS_TSDP_PROTECT.ADD_POLICY or DBMS_TSDP_PROTECT.ALTER_POLICY procedure.

The following table describes these settings.

Table 15-4 TDE Column Encryption ENCRYPT Settings Used for TSDP Policies

Parameter	Description	Default
encrypt_algorithm	Available values	AES192
_	• 3DES168	
	• AES128	
	• AES192	
	• AES256	
	• ARIA128	
	• ARIA192	
	• ARIA256	
	• SEED128	
	• GOST256	
salt	Available values:	SALT
	• SALT	
	• NO SALT	
integrity_algorithm	Available values:	SHA-1
	• SHA-1	
	• NOMAC	

15.16 Transparent Sensitive Data Protection Data Dictionary Views

Oracle Database provides data dictionary views that list information about transparent sensitive data protection policies.

Table 15-5 describes these views. Before you can use these views, you must be granted the SELECT CATALOG ROLE role.



Table 15-5 Transparent Sensitive Data Protection Views

View	Description
DBA_DISCOVERY_SOURCE	Describes discovery import information with regard to transparent sensitive data protection policies
DBA_SENSITIVE_COLUMN_TYPES	Describes the sensitive column types that have been defined for the current database
DBA_SENSITIVE_DATA	Describes the sensitive columns in the database
DBA_TSDP_IMPORT_ERRORS	Shows information regarding the errors encountered during import of discovery result. It shows information with regard to the error code, schema name, table name, column name, and sensitive type.
DBA_TSDP_POLICY_CONDITION	Describes the transparent sensitive data protection policy and condition mapping. This view also lists the property-value pairs for the condition.
DBA_TSDP_POLICY_FEATURE	Shows the transparent sensitive data protection policy security feature mapping. (At this time, only Oracle Data Redaction and Oracle Virtual Private Database are supported.)
DBA_TSDP_POLICY_PARAMETER	Describes the parameters of transparent sensitive data protection policies
DBA_TSDP_POLICY_PROTECTION	Shows the list of columns that have been protected through transparent sensitive data protection
DBA_TSDP_POLICY_TYPE	Shows the policy to sensitive column type mapping

Related Topics

Oracle Database Reference



Encryption of Sensitive Credential Data in the Data Dictionary

You can encrypt sensitive credential information, such as passwords that are stored in the data dictionary.

- About Encrypting Sensitive Credential Data in the Data Dictionary
 The data dictionary SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL system tables store sensitive credential data, such as user passwords.
- How the Multitenant Option Affects the Encryption of Sensitive Data
 In a multitenant environment, you can encrypt sensitive data dictionary information from the application root, as well as within individual pluggable databases (PDBs).
- Encrypting Sensitive Credential Data in System Tables
 The ALTER DATABASE DICTIONARY statement can encrypt sensitive credential data in the
 SYS.LINK\$ and SYS.SCHEDULER\$ CREDENTIAL system tables.
- Rekeying Sensitive Credential Data in the SYS.LINK\$ System Table
 You can use the ALTER DATABASE DICTIONARY statement to rekey sensitive credential data
 in the data dictionary SYS.LINK\$ and SYS.SCHEDULER\$ CREDENTIAL system tables.
- Deleting Sensitive Credential Data in System Tables
 The ALTER DATABASE DICTIONARY statement can invalidate existing credentials in SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL and obfuscate future credential entries to those tables.
- Restoring the Functioning of Database Links After a Lost Keystore
 Database links can be adversely affected if the TDE keystore and its master encryption
 key is inadvertently lost.
- Data Dictionary Views for Encrypted Data Dictionary Credentials
 Oracle Database provides a set of data dictionary views that provide information about the encryption of sensitive credential data in the data dictionary.

16.1 About Encrypting Sensitive Credential Data in the Data Dictionary

The data dictionary SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL system tables store sensitive credential data, such as user passwords.

The SYS.LINK\$ table stores information about database links. SYS.SCHEDULER\$_CREDENTIAL stores information about Oracle Scheduler events. By default, the sensitive credential data stored in these tables is obfuscated.

You can manually encrypt the data that is stored in the SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL tables by using the ALTER DATABASE DICTIONARY statement. Though this feature makes use of Transparent Data Encryption (TDE), you do not need to have an Advanced Security Option license to perform the encryption, but you must have the SYSKM administrative privilege. TDE performs the encryption by using the AES256 (Advanced

Encryption Standard) algorithm. The encryption follows the same behavior as other data that is encrypted using TDE.

As a best security practice, Oracle recommends that you encrypt this sensitive credential data. To check the status the data dictionary credentials, you can query the DICTIONARY CREDENTIALS ENCRYPT data dictionary view.

16.2 How the Multitenant Option Affects the Encryption of Sensitive Data

In a multitenant environment, you can encrypt sensitive data dictionary information from the application root, as well as within individual pluggable databases (PDBs).

When you encrypt, rekey, or decrypt sensitive credential data in the SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL system tables, you must synchronize the affected PDBs after you complete the process. The instructions for doing so are in the procedures that cover these topics.

16.3 Encrypting Sensitive Credential Data in System Tables

The ALTER DATABASE DICTIONARY statement can encrypt sensitive credential data in the SYS.LINK\$ and SYS.SCHEDULER\$ CREDENTIAL system tables.

The database must have an open keystore and an encryption key before you run the ALTER DATABASE DICTIONARY statement with the ENCRYPT CREDENTIALS clause to encrypt SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL. The credential data encryption process de-obfuscates the obfuscated passwords and then encrypts them. The encryption applies to any future password changes that users may make after you complete this process.

1. Connect to the database instance as a user who as been granted the SYSKM administrative privilege.

For example:

```
CONNECT hr_admin AS SYSKM Enter password: password
```

In a multitenant environment, connect to either the application root or to a pluggable database (PDB).

2. If necessary, create and open a keystore and then set an encryption key.

You can query the V\$ENCRYPTION WALLET dynamic view to find the status of a keystore.

Use the administer key management statement to perform these three tasks. For example:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl' IDENTIFIED BY password;

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";

ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password" WITH BACKUP;
```

In a multitenant environment, include the CONTAINER = ALL clause if you are in the application root. This applies the keystore operation for PDBs that are in united mode. For PDBs that are in isolated mode, run the statement from within the PDB.

Run the ALTER DATABASE DICTIONARY statement to encrypt the data.

For example:



ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;

In an application root, to apply the encryption to the associated PDBs, include the CONTAINER = ALL clause.

ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS CONTAINER = ALL;

If you performed the encryption from the application root, then synchronize the associated PDBs.

ALTER PLUGGABLE DATABASE APPLICATION APP\$CDB\$SYSTEM SYNC;

16.4 Rekeying Sensitive Credential Data in the SYS.LINK\$ System Table

You can use the ALTER DATABASE DICTIONARY statement to rekey sensitive credential data in the data dictionary SYS.LINK\$ and SYS.SCHEDULER\$ CREDENTIAL system tables.

To rekey this sensitive credential data, you must run the ALTER DATABASE DICTIONARY statement with the REKEY CREDENTIALS clause. The rekey operation, which uses column encryption, does not affect other TDE master encryption keys.

 Connect to the database instance as a user who as been granted the SYSKM administrative privilege.

For example:

```
CONNECT hr_admin AS SYSKM Enter password: password
```

In a multitenant environment, connect to either the application root or to a pluggable database (PDB).

2. If necessary, create and open a keystore and then set an encryption key.

You can query the V\$ENCRYPTION WALLET dynamic view to find the status of a keystore.

Use the ADMINISTER KEY MANAGEMENT statement to perform these three tasks. For example:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl' IDENTIFIED BY password;

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";

ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password" WITH BACKUP;
```

In a multitenant environment, include the CONTAINER = ALL clause if you are in the application root.

3. Run the ALTER DATABASE DICTIONARY statement to rekey the data.

For example:

```
ALTER DATABASE DICTIONARY REKEY CREDENTIALS;
```

In an application root, to apply the encryption to the associated PDBs, include the CONTAINER = ALL clause.

```
ALTER DATABASE DICTIONARY REKEY CREDENTIALS CONTAINER = ALL;
```

If you performed the rekey operation from the application root, then synchronize the associated PDBs.

ALTER PLUGGABLE DATABASE APPLICATION APP\$CDB\$SYSTEM SYNC;



16.5 Deleting Sensitive Credential Data in System Tables

The ALTER DATABASE DICTIONARY statement can invalidate existing credentials in SYS.LINK\$ and SYS.SCHEDULER\$ CREDENTIAL and obfuscate future credential entries to those tables.

To delete this credential data, you must run the ALTER DATABASE DICTIONARY statement with the DELETE CREDENTIALS clause. This statement is mainly used in cases where you must recover the database link from the loss of a Transparent Data Encryption (TDE) keystore.

1. Connect to the database instance as a user who as been granted the SYSKM administrative privilege.

For example:

```
CONNECT hr_admin AS SYSKM Enter password: password
```

In a multitenant environment, connect to either the application root or to a pluggable database (PDB).

2. If necessary, create and open a keystore and then set an encryption key.

You can query the V\$ENCRYPTION WALLET dynamic view to find the status of a keystore.

Use the ADMINISTER KEY MANAGEMENT statement to perform these three tasks. For example:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl' IDENTIFIED BY password;

ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";

ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password" WITH BACKUP;
```

In a multitenant environment, include the ${\tt CONTAINER} = {\tt ALL}$ clause if you are in the application root.

3. Run the ALTER DATABASE DICTIONARY statement to delete the password credential.

For example:

```
ALTER DATABASE DICTIONARY DELETE CREDENTIALS KEY;
```

In an application root, to delete the SYS.LINK\$ and SYS.SCHEDULER\$_CREDENTIAL password credentials in the associated PDBs, include the CONTAINER = ALL clause.

```
ALTER DATABASE DICTIONARY DELETE CREDENTIALS CONTAINER = ALL;
```

4. If you performed the credential deletion from the application root, then synchronize the associated PDBs.

ALTER PLUGGABLE DATABASE APPLICATION APP\$CDB\$SYSTEM SYNC;

Related Topics

Restoring the Functioning of Database Links After a Lost Keystore
 Database links can be adversely affected if the TDE keystore and its master encryption key is inadvertently lost.

16.6 Restoring the Functioning of Database Links After a Lost Keystore

Database links can be adversely affected if the TDE keystore and its master encryption key is inadvertently lost.

When a TDE keystore and master encryption key are lost, existing database links that are authenticated with encrypted passwords become unusable.

1. Connect to the database instance as a user who as been granted the SYSKM administrative privilege and who has the ALTER DATABASE LINK system privilege.

For example:

```
CONNECT hr_admin AS SYSKM Enter password: password
```

In a multitenant environment, connect to either the application root or to a pluggable database (PDB).

2. Delete the encrypted credentials from the SYS.LINK\$ system table.

```
ALTER DATABASE DICTIONARY DELETE CREDENTIALS KEY;
```

If you are performing the deletion from the application root, then include the CONTAINER = ALL clause.

```
ALTER DATABASE DICTIONARY DELETE CREDENTIALS CONTAINER = ALL;
```

Create and open a keystore and then set an encryption key.

For example:

```
ADMINISTER KEY MANAGEMENT CREATE KEYSTORE '/etc/ORACLE/WALLETS/orcl' IDENTIFIED BY password;
ADMINISTER KEY MANAGEMENT SET KEYSTORE OPEN IDENTIFIED BY "password";
ADMINISTER KEY MANAGEMENT SET ENCRYPTION KEY IDENTIFIED BY "password" WITH BACKUP;
```

In a multitenant environment, include the CONTAINER = ALL clause if you are in the application root.

4. Encrypt the password credentials in SYS.LINK\$ and SYS.SCHEDULER\$ CREDENTIAL.

```
ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS;
```

If you are performing the encryption from the application root, then include the CONTAINER = ALL clause.

```
ALTER DATABASE DICTIONARY ENCRYPT CREDENTIALS CONTAINER = ALL;
```

5. Using the password of the user who is associated with the database link, reset the database link passwords that were affected by the ALTER DATABASE DICTIONARY DELETE CREDENTIALS KEY Statement.

For example:

```
ALTER DATABASE LINK database_link_name CONNECT TO user_id IDENTIFIED BY password;
```

To find existing database links and their owners, query the DBA_DB_LINKS data dictionary view.

If you performed the credential deletion from the application root, then synchronize the associated PDBs.

ALTER PLUGGABLE DATABASE APPLICATION APP\$CDB\$SYSTEM SYNC;

16.7 Data Dictionary Views for Encrypted Data Dictionary Credentials

Oracle Database provides a set of data dictionary views that provide information about the encryption of sensitive credential data in the data dictionary.

Table 16-1 lists the data dictionary views.

Table 16-1 Data Dictionary Views for Encrypted Data Dictionary Credentials

View	Description
ALL_DB_LINKS	Describes database links that are accessible to the current user. A value of YES in the VALID column indicates that the database link is usable.
DBA_DB_LINKS	Describes describes all database links in the database. A value of YES in the VALID column indicates that the database link is usable. (This view is available to administrative users only, such as SYS or users who have been granted the DBA role.)
DICTIONARY_CREDENTIALS_ENCRYPT	Describes the status of dictionary credentials. The ENFORCEMENT column lists ENABLED if the credentials are encrypted and DISABLED if the credentials are not encrypted.
USER_DB_LINKS	Describes the database links hat are owned by the current user. A value of ${\tt YES}$ in the ${\tt VALID}$ column indicates that the database link is usable.

Related Topics

Oracle Database Reference



Manually Encrypting Data

You can use the DBMS CRYPTO PL/SQL package to manually encrypt data.

- Security Problems That Encryption Does Not Solve
 While there are many good reasons to encrypt data, there are many reasons not to encrypt
 data.
- Data Encryption Challenges
 In cases where encryption can provide additional security, there are some associated technical challenges.
- Data Encryption Storage with the DBMS_CRYPTO Package
 The DBMS_CRYPTO package provides several ways to address security issues.
- Asymmetric Key Operations with the DBMS_CRYPTO Package
 The DBMS_CRYPTO package provides four functions that enable you to perform asymmetric key operations for encryption, decryption, signing, and verification.
- Using Ciphertexts Encrypted in OFB Mode in Oracle Database Release 11g
 In Oracle Database Release 11g, ciphertexts configured to use output feedback (OFB) used electronic codebook (ECB) mode instead.
- Examples of Using the Data Encryption API Examples of using the data encryption API include using the DBMS_CRYPTO.SQL procedure, encrypting AES 256-bit data, and encrypting BLOB data.
- Data Dictionary Views for Encrypted Data
 Oracle Database provides data dictionary views to find information about encrypted data.

17.1 Security Problems That Encryption Does Not Solve

While there are many good reasons to encrypt data, there are many reasons not to encrypt data.

- Principle 1: Encryption Does Not Solve Access Control Problems
 When you encrypt data, you should be aware that encryption must not interfere with how you configure access control.
- Principle 2: Encryption Does Not Protect Against a Malicious Administrator
 You can protect your databases against malicious database administrators by using other
 Oracle features, such as Oracle Database Vault.
- Principle 3: Encrypting Everything Does Not Make Data Secure
 A common error is to think that if encrypting some data strengthens security, then encrypting everything makes all data secure.

17.1.1 Principle 1: Encryption Does Not Solve Access Control Problems

When you encrypt data, you should be aware that encryption must not interfere with how you configure access control.

Most organizations must limit data access to users who need to see this data. For example, a human resources system may limit employees to viewing only their own employment records,

while allowing managers of employees to see the employment records of subordinates. Human resource specialists may also need to see employee records for multiple employees.

Typically, you can use access control mechanisms to address security policies that limit data access to those with a need to see it. Oracle Database has provided strong, independently evaluated access control mechanisms for many years. It enables access control enforcement to a fine level of granularity through Virtual Private Database.

Because human resource records are considered sensitive information, it is tempting to think that all information should be encrypted for better security. However, encryption cannot enforce granular access control, and it may hinder data access. For example, an employee, his manager, and a human resources clerk may all need to access an employee record. If all employee data is encrypted, then all three must be able to access the data in unencrypted form. Therefore, the employee, the manager and the human resources clerk would have to share the same encryption key to decrypt the data. Encryption would, therefore, not provide any additional security in the sense of better access control, and the encryption might hinder the proper or efficient functioning of the application. An additional issue is that it is difficult to securely transmit and share encryption keys among multiple users of a system.

A basic principle behind encrypting stored data is that it must not interfere with access control. For example, a user who has the SELECT privilege on emp should not be limited by the encryption mechanism from seeing all the data he is otherwise allowed to see. Similarly, there is little benefit to encrypting part of a table with one key and part of a table with another key if users need to see all encrypted data in the table. In this case, encryption adds to the overhead of decrypting the data before users can read it. If access controls are implemented well, then encryption adds little additional security within the database itself. A user who has privileges to access data within the database has no more nor any less privileges as a result of encryption. Therefore, you should never use encryption to solve access control problems.

17.1.2 Principle 2: Encryption Does Not Protect Against a Malicious Administrator

You can protect your databases against malicious database administrators by using other Oracle features, such as Oracle Database Vault.

Some organizations, concerned that a malicious user might gain elevated (database administrator) privileges by guessing a password, like the idea of encrypting stored data to protect against this threat.

However, the correct solution to this problem is to protect the database administrator account, and to change default passwords for other privileged accounts. The easiest way to break into a database is by using a default password for a privileged account that an administrator allowed to remain unchanged. One example is SYS/CHANGE ON INSTALL.

While there are many destructive things a malicious user can do to a database after gaining the DBA privilege, encryption will not protect against many of them. Examples include corrupting or deleting data, exporting user data to the file system to email the data back to himself to run a password cracker on it, and so on.

Some organizations are concerned that database administrators, typically having all privileges, are able to see all data in the database. These organizations feel that the database administrators should administer the database, but should not be able to see the data that the database contains. Some organizations are also concerned about concentrating so much privilege in one person, and would prefer to partition the DBA function, or enforce two-person access rules.



It is tempting to think that encrypting all data (or significant amounts of data) will solve these problems, but there are better ways to protect against these threats. For example, Oracle Database supports limited partitioning of DBA privileges. Oracle Database provides native support for SYSDBA and SYSOPER users. SYSDBA has all privileges, but SYSOPER has a limited privilege set (such as startup and shutdown of the database).

Furthermore, you can create smaller roles encompassing several system privileges. A jr_dba role might not include all system privileges, but only those appropriate to a junior database administrator (such as CREATE TABLE, CREATE USER, and so on).

Oracle Database also enables auditing the actions taken by SYS (or SYS-privileged users) and storing that audit trail in a secure operating system location. Using this model, a separate auditor who has root privileges on the operating system can audit all actions by SYS, enabling the auditor to hold all database administrators accountable for their actions.

You can also fine-tune the access and control that database administrators have by using Oracle Database Vault.

The database administrator function is a trusted position. Even organizations with the most sensitive data, such as intelligence agencies, do not typically partition the database administrator function. Instead, they manage their database administrators strongly, because it is a position of trust. Periodic auditing can help to uncover inappropriate activities.

Encryption of stored data must not interfere with the administration of the database, because otherwise, larger security issues can result. For example, if by encrypting data you corrupt the data, then you create a security problem, the data itself cannot be interpreted, and it may not be recoverable.

You can use encryption to limit the ability of a database administrator or other privileged user to see data in the database. However, it is not a substitute for managing the database administrator privileges properly, or for controlling the use of powerful system privileges. If untrustworthy users have significant privileges, then they can pose multiple threats to an organization, some of them far more significant than viewing unencrypted credit card numbers.



Oracle Database Vault Administrator's Guide for more information about using Oracle Database Vault to fine-tune the access and control that database administrators have

17.1.3 Principle 3: Encrypting Everything Does Not Make Data Secure

A common error is to think that if encrypting some data strengthens security, then encrypting everything makes all data secure.

As the discussion of the previous two principles illustrates, encryption does not address access control issues well, and it is important that encryption not interfere with normal access controls. Furthermore, encrypting an entire production database means that all data must be decrypted to be read, updated, or deleted. Encryption is inherently a performance-intensive operation; encrypting all data will significantly affect performance.

Availability is a key aspect of security. If encrypting data makes data unavailable, or adversely affects availability by reducing performance, then encrypting everything will create a new security problem. Availability is also adversely affected by the database being inaccessible when encryption keys are changed, as good security practices require on a regular basis.



When the keys are to be changed, the database is inaccessible while data is decrypted and reencrypted with a new key or keys.

There may be advantages to encrypting data stored off-line. For example, an organization may store backups for a period of 6 months to a year off-line, in a remote location. Of course, the first line of protection is to secure the facility storing the data, by establishing physical access controls. Encrypting this data before it is stored may provide additional benefits. Because it is not being accessed on-line, performance need not be a consideration. While an Oracle database does not provide this capability, there are vendors who provide encryption services. Before embarking on large-scale encryption of backup data, organizations considering this approach should thoroughly test the process. It is essential to verify that data encrypted before off-line storage can be decrypted and re-imported successfully.

17.2 Data Encryption Challenges

In cases where encryption can provide additional security, there are some associated technical challenges.

- Encrypted Indexed Data
 Special difficulties arise when encrypted data is indexed.
- Generated Encryption Keys
 Encrypted data is only as secure as the key used for encrypting it.
- Transmitted Encryption Keys
 If the encryption key is to be passed by the application to the database, then you must encrypt it.
- Storing Encryption Keys
 You can store encryption keys in the database or on an operating system.
- Importance of Changing Encryption Keys
 Prudent security practice dictates that you periodically change encryption keys.
- Encryption of Binary Large Objects
 Certain data types require more work to encrypt.

17.2.1 Encrypted Indexed Data

Special difficulties arise when encrypted data is indexed.

For example, suppose a company uses a national identity number, such as the U.S. Social Security number (SSN), as the employee number for its employees. The company considers employee numbers to be sensitive data, and, therefore, wants to encrypt data in the <code>employee_number column</code> of the <code>employees</code> table. Because <code>employee_number contains</code> unique values, the database designers want to have an index on it for better performance.

However, if <code>DBMS_CRYPTO</code> (or another mechanism) is used to encrypt data in a column, then an index on that column will also contain encrypted values. Although an index can be used for equality checking (for example, <code>SELECT * FROM emp WHERE employee_number = '987654321')</code>, if the index on that column contains encrypted values, then the index is essentially unusable for any other purpose. You should not encrypt indexed data.

Oracle recommends that you do not use national identity numbers as unique IDs. Instead, use the CREATE SEQUENCE statement to generate unique identity numbers. Reasons to avoid using national identity numbers are as follows:



- There are privacy issues associated with overuse of national identity numbers (for example, identity theft).
- Sometimes national identity numbers can have duplicates, as with U.S. Social Security numbers.

17.2.2 Generated Encryption Keys

Encrypted data is only as secure as the key used for encrypting it.

An encryption key must be securely generated using secure cryptographic key generation. Oracle Database provides support for secure random number generation, with the RANDOMBYTES function of DBMS_CRYPTO. (This function replaces the capabilities provided by the Getkey procedure of the earlier DBMS_OBFUSCATION_TOOLKIT, which has been deprecated.) DBMS_CRYPTO calls the secure random number generator (RNG) previously certified by RSA Security.

Note:

Do not use the DBMS_RANDOM package. The DBMS_RANDOM package generates pseudorandom numbers, which, as Randomness Recommendations for Security (RFC-1750) states that using pseudo-random processes to generate secret quantities can result in pseudo-security.

Be sure to provide the correct number of bytes when you encrypt a key value. For example, you must provide a 16-byte key for the ENCRYPT AES128 encryption algorithm.

17.2.3 Transmitted Encryption Keys

If the encryption key is to be passed by the application to the database, then you must encrypt it.

Otherwise, an intruder could get access to the key as it is being transmitted. Network data encryption protects all data in transit from modification or interception, including cryptographic keys.

Related Topics

Configuring Oracle Database Native Network Encryption and Data Integrity
 You can configure native Oracle Net Services data encryption and data integrity for both servers and clients.

17.2.4 Storing Encryption Keys

You can store encryption keys in the database or on an operating system.

- About Storing Encryption Keys
 Storing encryption keys is one of the most important, yet difficult, aspects of encryption.
- Storage of Encryption Keys in the Database
 Storing encryption keys in the database does not always prevent a database administrator from accessing encrypted data.



- Storage of Encryption Keys in the Operating System
 When you store encryption keys in an operating system flat file, you can make callouts from PL/SQL to retrieve these encryption keys.
- Users Managing Their Own Encryption Keys
 Having the user supply the key assumes the user will be responsible with the key.
- Manual Encryption with Transparent Database Encryption and Tablespace Encryption
 Transparent database encryption and tablespace encryption provide secure encryption
 with automatic key management for the encrypted tables and tablespaces.

17.2.4.1 About Storing Encryption Keys

Storing encryption keys is one of the most important, yet difficult, aspects of encryption.

To recover data encrypted with a symmetric key, the key must be accessible to an authorized application or user seeking to decrypt the data. At the same time, the key must be inaccessible to someone who is maliciously trying to access encrypted data that he is not supposed to see.

17.2.4.2 Storage of Encryption Keys in the Database

Storing encryption keys in the database does not always prevent a database administrator from accessing encrypted data.

An all-privileged database administrator could still access tables containing encryption keys. However, it can often provide good security against the casual curious user or against someone compromising the database file on the operating system.

As a trivial example, suppose you create a table (EMP) that contains employee data. You want to encrypt the employee Social Security number (SSN) stored in one of the columns. You could encrypt employee SSN using a key that is stored in a separate column. However, anyone with SELECT access on the entire table could retrieve the encryption key and decrypt the matching SSN.

While this encryption scheme seems easily defeated, with a little more effort you can create a solution that is much harder to break. For example, you could encrypt the SSN using a technique that performs some additional data transformation on the <code>employee_number</code> before using it to encrypt the SSN. This technique might be as simple as using an XOR operation on the <code>employee_number</code> and the birth date of the employee to determine the validity of the values.

As additional protection, PL/SQL source code performing encryption can be wrapped, (using the WRAP utility) which obfuscates (scrambles) the code. The WRAP utility processes an input SQL file and obfuscates the PL/SQL units in it. For example, the following command uses the keymanage.sql file as the input:

```
wrap iname=/mydir/keymanage.sql
```

A developer can subsequently have a function in the package call the DBMS_CRYPTO package calls with the key contained in the wrapped package.

Oracle Database enables you to obfuscate dynamically generated PL/SQL code. The DBMS_DDL package contains two subprograms that allow you to obfuscate dynamically generated PL/SQL program units. For example, the following block uses the DBMS_DDL.CREATE_WRAPPED procedure to wrap dynamically generated PL/SQL code.

```
BEGIN
.....
SYS.DBMS DDL.CREATE WRAPPED(function returning PLSQL code());
```



END;

While wrapping is not unbreakable, it makes it harder for an intruder to get access to the encryption key. Even in cases where a different key is supplied for each encrypted data value, you should not embed the key value within a package. Instead, wrap the package that performs the key management (that is, data transformation or padding).

See Also:

Oracle Database PL/SQL Packages and Types Reference for additional information about the \mbox{WRAP} command line utility and the $\mbox{DBMS}_{\mbox{DDL}}$ subprograms for dynamic wrapping

An alternative to wrapping the data is to have a separate table in which to store the encryption key and to envelope the call to the keys table with a procedure. The key table can be joined to the data table using a primary key to foreign key relationship. For example, <code>employee_number</code> is the primary key in the <code>employees</code> table that stores employee information and the encrypted SSN. The <code>employee_number</code> column is a foreign key to the <code>ssn_keys</code> table that stores the encryption keys for the <code>employee SSN</code>. The key stored in the <code>ssn_keys</code> table can also be transformed before use (by using an <code>xor</code> operation), so the key itself is not stored unencrypted. If you wrap the procedure, then that can hide the way in which the keys are transformed before use.

The strengths of this approach are:

- Users who have direct table access cannot see the sensitive data unencrypted, nor can they retrieve the keys to decrypt the data.
- Access to decrypted data can be controlled through a procedure that selects the encrypted data, retrieves the decryption key from the key table, and transforms it before it can be used to decrypt the data.
- The data transformation algorithm is hidden from casual snooping by wrapping the procedure, which obfuscates the procedure code.
- SELECT access to both the data table and the keys table does not guarantee that the user with this access can decrypt the data, because the key is transformed before use.

The weakness to this approach is that a user who has \mathtt{SELECT} access to both the key table and the data table, and who can derive the key transformation algorithm, can break the encryption scheme.

The preceding approach is not infallible, but it is adequate to protect against easy retrieval of sensitive information stored in clear text.

17.2.4.3 Storage of Encryption Keys in the Operating System

When you store encryption keys in an operating system flat file, you can make callouts from PL/SQL to retrieve these encryption keys.

However, if you store keys in the operating system and make callouts to it, then your data is only as secure as the protection on the operating system.

If your primary security concern is that the database can be broken into from the operating system, then storing the keys in the operating system makes it easier for an intruder to retrieve encrypted data than storing the keys in the database itself.

17.2.4.4 Users Managing Their Own Encryption Keys

Having the user supply the key assumes the user will be responsible with the key.

Considering that 40 percent of help desk calls are from users who have forgotten their passwords, you can see the risks of having users manage encryption keys. In all likelihood, users will either forget an encryption key, or write the key down, which then creates a security weakness. If a user forgets an encryption key or leaves the company, then your data is not recoverable.

If you do decide to have user-supplied or user-managed keys, then you need to ensure you are using native network encryption so that the key is not passed from the client to the server in the clear. You also must develop key archive mechanisms, which is also a difficult security problem. Key archives and backdoors create the security weaknesses that encryption is attempting to solve.

17.2.4.5 Manual Encryption with Transparent Database Encryption and Tablespace Encryption

Transparent database encryption and tablespace encryption provide secure encryption with automatic key management for the encrypted tables and tablespaces.

If the application requires protection of sensitive column data stored on the media, then these two types of encryption are a simple and fast way of achieving this.



Oracle Database Advanced Security Guide for more information about Transparent Data Encryption

17.2.5 Importance of Changing Encryption Keys

Prudent security practice dictates that you periodically change encryption keys.

For stored data, this requires periodically unencrypting the data, and then reencrypting it with another well-chosen key.

You would most likely change the encryption key while the data is not being accessed, which creates another challenge. This is especially true for a Web-based application encrypting credit card numbers, because you do not want to shut down the entire application while you switch encryption keys.

17.2.6 Encryption of Binary Large Objects

Certain data types require more work to encrypt.

For example, Oracle Database supports storage of binary large objects (BLOBs), which stores very large objects (for example, multiple gigabytes) in the database. A BLOB can be either stored internally as a column, or stored in an external file.



Related Topics

 Example: Encryption and Decryption Procedures for BLOB Data You can encrypt BLOB data.

17.3 Data Encryption Storage with the DBMS_CRYPTO Package

The DBMS CRYPTO package provides several ways to address security issues.

While encryption is not the ideal solution for addressing several security threats, it is clear that selectively encrypting sensitive data before storage in the database does improve security. Examples of such data could include credit card numbers and national identity numbers.

Oracle Database provides the PL/SQL package DBMS_CRYPTO to encrypt and decrypt stored data. This package supports several industry-standard encryption and hashing algorithms, including the Advanced Encryption Standard (AES) encryption algorithm. AES was approved by the National Institute of Standards and Technology (NIST) to replace the Data Encryption Standard (DES).

The DBMS_CRYPTO package enables encryption and decryption for common Oracle Database data types, including RAW and large objects (LOBs), such as images and sound. Specifically, it supports BLOBs and CLOBs. In addition, it provides Globalization Support for encrypting data across different database character sets.

The following cryptographic algorithms are supported:

- Advanced Encryption Standard (AES)
- SHA-2 Cryptographic Hash settings:
 - HASH SH256
 - HASH SH384
 - HASH SH512
- SHA-2 Message Authentication Code (MAC)

Block cipher modifiers are also provided with DBMS_CRYPTO. You can choose from several padding options, including Public Key Cryptographic Standard (PKCS) #5, and from four block cipher chaining modes, including Cipher Block Chaining (CBC). Padding must be done in multiples of eight bytes.



Note:

- DES is no longer recommended by the National Institute of Standards and Technology (NIST).
- Usage of SHA-1 is more secure than MD5. (MD5 has been deprecated starting in Oracle Database 21c.)

Starting with Oracle Database 21c, older encryption and hashing algorithms are deprecated. Deprecated algorithms include MD4, MD5, DES, 3DES, and RC4-related algorithms. Removing older, less secure cryptography algorithms prevents accidental use of these APIs. To meet your security requirements, Oracle recommends that you use more modern cryptography algorithms such as AES.

Starting with Oracle Database 21c, older encryption and hashing algorithms are deprecated.

As a consequence of this deprecation, Oracle recommends that you review your network encryption configuration to see if you have specified use of any of the deprecated algorithms. If any are found, then switch to using a more modern cipher, such as AES. See Improving Native Network Encryption Security for more information.

- Usage of SHA-2 is more secure than SHA-1.
- Keyed MD5 is not vulnerable.

Table 17-1 summarizes the DBMS CRYPTO package features.

Table 17-1 DBMS_CRYPTO Package Feature Summary

Feature	DBMS_CRYPTO Supported Functionality
Block cipher chaining modes	CBC, CFB, ECB, OFB
Cryptographic algorithms	AES
Cryptographic hash algorithms	SHA-1, SHA-2, HASH_SH256, HASH_SH384, HASH_SH512
Cryptographic pseudo-random number generator	RAW, NUMBER, BINARY_INTEGER
Database types	RAW, CLOB, BLOB
Keyed hash (MAC) algorithms	HMAC_MD5, HMAC_SH1, HMAC_SH256, HMAC_SH384, HMAC_SH512
Padding forms	PKCS5, zeroes

Table 17-2 shows supported SHA hash functions, many of which can be used with RSA environments.

Table 17-2 SHA Hash Algorithms

Hash Algorithm	Description
SIGN_RSA_PKCS1_OAEP_SHA256	RSA with Public Key Cryptographic Standards, SHA 256 bit hash function and OAEP padding



Table 17-2 (Cont.) SHA Hash Algorithms

Hash Algorithm	Description
SIGN_SHA1_ECDSA	SHA hash function with Elliptic Curve Digital Signature Algorithm
SIGN_SHA1_RSA	SHA hash function with RSA
SIGN_SHA1_RSA_X931	SHA hash function with RSA and X931 padding
SIGN_SHA224_ECDSA	SHA 224 bit hash function with Elliptic Curve Digital Signature Algorithm
SIGN_SHA224_RSA	SHA 224 bit hash function with RSA
SIGN_SHA256_ECDSA	SHA 256 bit hash function with Elliptic Curve Digital Signature Algorithm
SIGN_SHA256_RSA	SHA 256 bit hash function with RSA
SIGN_SHA256_RSA_X931	SHA 256 bit hash function with RSA and X931 padding
SIGN_SHA384_ECDSA	SHA 384 bit hash function with Elliptic Curve Digital Signature Algorithm
SIGN_SHA384_RSA	SHA 384 bit hash function with RSA
SIGN_SHA384_RSA_X931	SHA 384 bit hash function with RSA and X931 padding
SIGN_SHA512_ECDSA	SHA 512bit hash function with Elliptic Curve Digital Signature Algorithm
SIGN_SHA512_RSA	SHA 384 bit hash function with RSA
SIGN_SHA512_RSA_X931	SHA 384 bit hash function with RSA and X931 padding

Table 17-3 shows supported encryption and decryption algorithms.

Table 17-3 Encryption and Decryption Algorithms

Algorithm	Description
PKENCRYPT_ECDH	Elliptic Curve Diffie Hellman
PKENCRYPT_RSA_PKCS1_OAEP	RSA Public Key Cryptosystem with PKCS1 and OAEP padding

Table 17-4 shows other supported algorithms.

Table 17-4 Other Algorithms

Algorithm	Description
KEY_TYPE_RSA	RSA key type
SIGN_ECDSA	Elliptic Curve Digital Signature Algorithm

DBMS_CRYPTO supports a range of algorithms that accommodate both new and existing systems. Although 3DES_2KEY and MD4 are provided for backward compatibility, you achieve better security using 3DES, AES, or SHA-1. Therefore, 3DES_2KEY is not recommended.



The DBMS_CRYPTO package includes cryptographic checksum capabilities (MD5), which are useful for comparisons, and the ability to generate a secure random number (the RANDOMBYTES function). Secure random number generation is an important part of cryptography; predictable keys are easily guessed keys; and easily guessed keys may lead to easy decryption of data. Most cryptanalysis is done by finding weak keys or poorly stored keys, rather than through brute force analysis (cycling through all possible keys).



Do not use DBMS RANDOM, because it is unsuitable for cryptographic key generation.

Key management is programmatic. That is, the application (or caller of the function) must supply the encryption key. This means that the application developer must find a way of storing and retrieving keys securely. The relative strengths and weaknesses of various key management techniques are discussed in the sections that follow. The DES algorithm itself has an effective key length of 56-bits.

17.4 Asymmetric Key Operations with the DBMS_CRYPTO Package

The DBMS_CRYPTO package provides four functions that enable you to perform asymmetric key operations for encryption, decryption, signing, and verification.

Asymmetric key operations (also called public key cryptography) use a public key and private key to encrypt and decrypt a message in order to protect it from unauthorized access.

The asymmetric key operation functions are as follows:

- PKDECRYPT decrypts RAW data using a private key assisted with key algorithm and encryption algorithm.
- PKENCRYPT encrypts RAW data using a public key assisted with key algorithm and encryption algorithm.
- SIGN signs RAW data using a private key assisted with key algorithm and sign algorithm
- VERIFY verifies RAW data using signature, public key assisted with key algorithm and sign algorithm.

Related Topics

Oracle Database PL/SQL Packages and Types Reference

17.5 Using Ciphertexts Encrypted in OFB Mode in Oracle Database Release 11g

In Oracle Database Release 11g, ciphertexts configured to use output feedback (OFB) used electronic codebook (ECB) mode instead.

In Oracle Database Release 11*g*, if you set the <code>DBMS_CRYPTO.CHAIN_OFB</code> block cipher chaining modifier to configure ciphertext encryption to use output feedback (OFB) mode, then due to Oracle Bug 13001552, the result is that the configuration used electronic codebook (ECB) mode erroneously. This bug has been fixed in Oracle Database Release 12c. Therefore, after

an upgrade from Oracle Database release 11g to Release 12c, the ciphertexts that were encrypted using OFB mode in release 11g will no longer decrypt properly in the corrected OFB mode in Oracle Database Release 12c or later.

To remedy this problem:

- Log in to the database as a user who has the EXECUTE privilege for the DBMS_CRYPTO PL/SQL package.
- 2. Decrypt the cyphertexts using the DBMS CRYPTO.CHAIN ECB block cipher chaining modifier.

The following example, dbmscryptoll.sql, shows the wrong behavior in Oracle Database Release 11*q*:

```
dbmscryptol1.sql:
set serveroutput on
declare
 1 mod ofb pls integer;
 1 mod ecb pls integer;
 v key raw(32);
 v iv raw(16);
 v test in raw(16);
 v ciphertext raw(16);
  v test out ECB raw(16);
 v test out OFB raw(16);
begin
  1 mod ofb := dbms crypto.ENCRYPT AES256
       + dbms crypto.CHAIN OFB
       + DBMS CRYPTO.PAD NONE ;
  1 mod ecb := dbms crypto.ENCRYPT AES256
       + dbms crypto.CHAIN ECB
       + DBMS CRYPTO.PAD NONE ;
  v key := hextoraw
  ('603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4');
 v iv := hextoraw('000102030405060708090A0B0C0D0E0F');
  v test in := hextoraw('6bc1bee22e409f96e93d7e117393172a');
  v ciphertext := dbms crypto.encrypt(src => v test in,
                                  TYP => 1 \mod ofb,
                                  key => v key,
                                  iv => v iv);
  v test out ECB := dbms crypto.decrypt(src => v ciphertext,
                                  TYP => 1 mod ecb,
                                  key => v key,
                                  iv \Rightarrow v iv);
  v test out OFB := dbms crypto.decrypt(src => v ciphertext,
                                  TYP => 1 \mod ofb,
                                  key => v key,
                                  iv => v iv);
  dbms output.put line
  ('Input plaintext
                                         : '||rawtohex(v test in));
  dbms output.put line
  ('11g: Ciphertext (encrypt in OFB mode): '||rawtohex(v ciphertext));
  dbms output.put line
  ('11g: Output of decrypt in ECB mode : '||rawtohex(v_test_out_ECB));
  dbms output.put line
  ('11g: Output of decrypt in OFB mode : '||rawtohex(v test out OFB));
```

```
end;
/
```

The resulting output is as follows:

```
SQL> @dbmscryptol1.sql

Input plaintext : 6BC1BEE22E409F96E93D7E117393172A

11g: Ciphertext (encrypt in OFB mode): F3EED1BDB5D2A03C064B5A7E3DB181F8

11g: Output of decrypt in ECB mode : 6BC1BEE22E409F96E93D7E117393172A

11g: Output of decrypt in OFB mode : 6BC1BEE22E409F96E93D7E117393172A
```

This output illustrates that in Oracle Database release 11*g*, OFB mode is wrongly ECB mode, and therefore decrypting in either OFB or ECB mode results in the correct plaintext.

The next example, dbmscrypto12from11.sql, shows that, after an upgrade from Oracle Database release 11g to release 12c, ECB mode and not OFB mode has to be used in order to properly decrypt a ciphertext encrypted in OFB mode in Release 11g.

```
dbmscrypto12from11.sql:
set serveroutput on
declare
  1 mod ofb pls integer;
  1 mod ecb pls integer;
  v key raw(32);
  v iv raw(16);
  v test in raw(16);
  v ciphertext11 raw(16);
  v_test_out_ECB raw(16);
  v test out OFB raw(16);
begin
  1 mod ofb := dbms crypto.ENCRYPT AES256
       + dbms crypto.CHAIN OFB
       + DBMS CRYPTO.PAD NONE ;
  1_mod_ecb := dbms_crypto.ENCRYPT_AES256
       + dbms_crypto.CHAIN ECB
       + DBMS CRYPTO.PAD NONE ;
  v key := hextoraw
  ('603deb1015ca71be2b73aef0857d77811f352c073b6108d72d9810a30914dff4');
  v iv := hextoraw('000102030405060708090A0B0C0D0E0F');
  v test in := hextoraw('6bc1bee22e409f96e93d7e117393172a');
  v ciphertext11 := hextoraw('F3EED1BDB5D2A03C064B5A7E3DB181F8');
  v test out ECB := dbms crypto.decrypt(src => v ciphertext11,
                                  TYP => 1 mod ecb,
                                  key => v key,
                                  iv => v iv);
  v test out OFB := dbms crypto.decrypt(src => v ciphertext11,
                                  TYP => 1 mod ofb,
                                  key => v_key,
                                  iv => v iv);
  dbms output.put line
  ('Input plaintext (to 11g)
                                          : '||rawtohex(v test in));
```



```
dbms_output.put_line
  ('11g: Ciphertext (encrypt in OFB mode): '||rawtohex(v_ciphertext11));
  dbms_output.put_line
   ('12c: Output of decrypt in ECB mode : '||rawtohex(v_test_out_ECB));
  dbms_output.put_line
   ('12c: Output of decrypt in OFB mode : '||rawtohex(v_test_out_OFB));
end;
/
```

The resulting output is as follows:

```
SQL> @dbmscrypto12from11.sql
Input plaintext (to 11g) : 6BC1BEE22E409F96E93D7E117393172A
11g: Ciphertext (encrypt in OFB mode): F3EED1BDB5D2A03C064B5A7E3DB181F8
12c: Output of decrypt in ECB mode : 6BC1BEE22E409F96E93D7E117393172A
12c: Output of decrypt in OFB mode : 4451EBE041EB29E191BBA0E9D67FAEB2
```

If you are preparing to upgrade from Oracle Database Release 11g to Release 12c, then edit any scripts that you may have in which OFB mode is specified so that the decrypt operations use ECB mode. This way, the scripts will work in both release 11g and release 12c and later, ensuring business continuity.

17.6 Examples of Using the Data Encryption API

Examples of using the data encryption API include using the DBMS_CRYPTO.SQL procedure, encrypting AES 256-bit data, and encrypting BLOB data.

- Example: Data Encryption Procedure
 The DBMS CRYPTO.SQL PL/SQL program can be used to encrypt data.
- Example: AES 256-Bit Data Encryption and Decryption Procedures
 You can use a PL/SQL block to encrypt and decrypt a predefined variable.
- Example: Encryption and Decryption Procedures for BLOB Data You can encrypt BLOB data.

17.6.1 Example: Data Encryption Procedure

The DBMS CRYPTO.SQL PL/SQL program can be used to encrypt data.

This example code performs the following actions:

Encrypts a string (VARCHAR2 type) using DES after first converting it into the RAW data type.

This step is necessary because encrypt and decrypt functions and procedures in DBMS CRYPTO package work on the RAW data type only.

- Shows how to create a 160-bit hash using SHA-1 algorithm.
- Demonstrates how MAC, a key-dependent one-way hash, can be computed using the MD5 algorithm.

The DBMS CRYPTO.SQL procedure follows:

```
UTL RAW.CAST_TO_RAW(CONVERT(key_string,'AL32UTF8','US7ASCII'));
   encrypted raw RAW(2048);
   encrypted_string VARCHAR2(2048);
   decrypted raw RAW(2048);
   decrypted string VARCHAR2 (2048);
-- Begin testing Encryption:
   dbms output.put line('> Input String
   CONVERT(UTL_RAW.CAST_TO_VARCHAR2(raw_input),'US7ASCII','AL32UTF8'));
   dbms output.put line('> ====== BEGIN TEST Encrypt =======');
   encrypted raw := dbms_crypto.Encrypt(
       src => raw input,
       typ => DBMS CRYPTO.DES CBC PKCS5,
       key => raw key);
       dbms output.put line('> Encrypted hex value
                                                             : ' ||
       rawtohex(UTL RAW.CAST TO RAW(encrypted raw)));
decrypted raw := dbms crypto.Decrypt(
       src => encrypted raw,
       typ => DBMS CRYPTO.DES CBC PKCS5,
       key => raw key);
   decrypted string :=
   CONVERT (UTL RAW.CAST TO VARCHAR2 (decrypted raw), 'US7ASCII', 'AL32UTF8');
dbms output.put line('> Decrypted string output : ' ||
       decrypted string);
if input_string = decrypted_string THEN
   dbms_output.put_line('> String DES Encyption and Decryption successful');
END if;
dbms output.put line('');
dbms output.put line('> ====== BEGIN TEST Hash =======');
   encrypted_raw := dbms_crypto.Hash(
       src => raw_input,
       typ => DBMS CRYPTO.HASH SH1);
dbms_output.put_line('> Hash value of input string
                                                     : ' ||
       rawtohex(UTL_RAW.CAST_TO_RAW(encrypted_raw)));
dbms output.put line('> ====== BEGIN TEST Mac =======');
   encrypted_raw := dbms_crypto.Mac(
       src => raw input,
       typ => DBMS CRYPTO.HMAC MD5,
       key => raw key);
dbms output.put line('> Message Authentication Code
                                                       : ' 11
       rawtohex(UTL RAW.CAST TO RAW(encrypted_raw)));
dbms output.put line('');
dbms_output.put_line('> End of DBMS_CRYPTO tests ');
END:
```

17.6.2 Example: AES 256-Bit Data Encryption and Decryption Procedures

You can use a PL/SQL block to encrypt and decrypt a predefined variable.

For the following example, the predefined variable is named input_string and it uses the AES 256-bit algorithm with Cipher Block Chaining and PKCS #5 padding:

```
PLS INTEGER :=
                                              -- total encryption type
   encryption type
                           DBMS CRYPTO.ENCRYPT AES256
                          + DBMS_CRYPTO.CHAIN_CBC
                          + DBMS CRYPTO.PAD PKCS5;
begin
   DBMS OUTPUT.PUT LINE ('Original string: ' || input string);
   key bytes raw := DBMS CRYPTO.RANDOMBYTES (num key bytes);
  encrypted raw := DBMS CRYPTO.ENCRYPT
         src => UTL_I18N.STRING_TO_RAW (input_string, 'AL32UTF8'),
         typ => encryption type,
         key => key bytes raw
     );
    -- The encrypted value in the encrypted_raw variable can be used here:
   decrypted raw := DBMS CRYPTO.DECRYPT
        src => encrypted raw,
        typ => encryption type,
        key => key bytes raw
   output string := UTL I18N.RAW TO CHAR (decrypted raw, 'AL32UTF8');
   DBMS OUTPUT.PUT LINE ('Decrypted string: ' || output string);
end;
```

17.6.3 Example: Encryption and Decryption Procedures for BLOB Data

You can encrypt BLOB data.

The following sample PL/SQL program (blob_test.sql) shows how to encrypt and decrypt BLOB data. This example code does the following, and prints out its progress (or problems) at each step:

- Creates a table for the BLOB column
- Inserts the raw values into that table
- Encrypts the raw data
- Decrypts the encrypted data

The blob test.sql procedure follows:

```
-- 1. Create a table for BLOB column:
create table table lob (id number, loc blob);
-- 2. Insert 3 empty lobs for src/enc/dec:
insert into table lob values (1, EMPTY BLOB());
insert into table lob values (2, EMPTY BLOB());
insert into table lob values (3, EMPTY BLOB());
set echo on
set serveroutput on
declare
   srcdata RAW(1000);
srcblob BLOB;
   encrypblob BLOB;
   encrypraw RAW(1000);
   encrawlen BINARY INTEGER;
   decrypblob BLOB;
    decrypraw RAW(1000);
    decrawlen BINARY_INTEGER;
```

```
leng
               INTEGER;
begin
    -- RAW input data 16 bytes
    srcdata := hextoraw('6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D6D);
    dbms_output.put_line('---');
    dbms_output.put_line('input is ' || srcdata);
    dbms_output.put_line('---');
    -- select empty lob locators for src/enc/dec
    select loc into srcblob from table lob where id = 1;
    select loc into encrypblob from table lob where id = 2;
    select loc into decrypblob from table_lob where id = 3;
    dbms output.put line('Created Empty LOBS');
    dbms output.put line('---');
    leng := DBMS LOB.GETLENGTH(srcblob);
    IF leng IS NULL THEN
       dbms output.put line('Source BLOB Len NULL ');
       dbms output.put line('Source BLOB Len ' || leng);
    END IF;
    leng := DBMS LOB.GETLENGTH(encrypblob);
    IF leng IS NULL THEN
        dbms output.put line('Encrypt BLOB Len NULL ');
    ELSE
       dbms output.put line('Encrypt BLOB Len ' || leng);
    END IF;
    leng := DBMS LOB.GETLENGTH(decrypblob);
    IF leng IS NULL THEN
       dbms output.put line('Decrypt BLOB Len NULL ');
    ELSE
       dbms output.put line('Decrypt BLOB Len ' || leng);
    END IF;
    -- 3. Write source raw data into blob:
    DBMS LOB.OPEN (srcblob, DBMS LOB.lob readwrite);
    DBMS LOB.WRITEAPPEND (srcblob, 16, srcdata);
    DBMS LOB.CLOSE (srcblob);
    dbms output.put line('Source raw data written to source blob');
    dbms output.put line('---');
    leng := DBMS LOB.GETLENGTH(srcblob);
    IF leng IS NULL THEN
       dbms output.put line('source BLOB Len NULL ');
       dbms output.put line('Source BLOB Len ' || leng);
    END IF;
    * Procedure Encrypt
    * Arguments: srcblob -> Source BLOB
                encrypblob -> Output BLOB for encrypted data
                 DBMS CRYPTO.AES CBC PKCS5 -> Algo : AES
                                              Chaining : CBC
                                              Padding: PKCS5
```

```
256 bit key for AES passed as RAW
                  ->
   hextoraw('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F')
               IV (Initialization Vector) for AES algo passed as RAW
                  DBMS CRYPTO. Encrypt (encrypblob,
              srcblob,
              DBMS CRYPTO.AES CBC PKCS5,
              hextoraw
('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F'),
              dbms output.put line('Encryption Done');
   dbms output.put line('---');
   leng := DBMS LOB.GETLENGTH(encrypblob);
   IF leng IS NULL THEN
       dbms output.put line('Encrypt BLOB Len NULL');
      dbms output.put line('Encrypt BLOB Len ' | leng);
   END IF;
   -- 4. Read encrypblob to a raw:
   encrawlen := 999;
   DBMS LOB.OPEN (encrypblob, DBMS LOB.lob readwrite);
   DBMS LOB.READ (encrypblob, encrawlen, 1, encrypraw);
   DBMS LOB.CLOSE (encrypblob);
   dbms_output.put_line('Read encrypt blob to a raw');
   dbms output.put line('---');
   dbms output.put line('Encrypted data is (256 bit key) ' || encrypraw);
   dbms output.put line('---');
   /*
   * Procedure Decrypt
   * Arguments: encrypblob -> Encrypted BLOB to decrypt
               decrypblob -> Output BLOB for decrypted data in RAW
               DBMS CRYPTO.AES CBC PKCS5 -> Algo : AES
                                         Chaining: CBC
                                         Padding: PKCS5
               256 bit key for AES passed as RAW (same as used during Encrypt)
   hextoraw('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F')
               IV (Initialization Vector) for AES algo passed as RAW (same as
               used during Encrypt)
                  DBMS CRYPTO.Decrypt(decrypblob,
              encrypblob,
              DBMS CRYPTO.AES_CBC_PKCS5,
              hext.oraw
          ('000102030405060708090A0B0C0D0E0F101112131415161718191A1B1C1D1E1F'),
              leng := DBMS LOB.GETLENGTH(decrypblob);
   IF leng IS NULL THEN
```

```
dbms output.put line('Decrypt BLOB Len NULL');
   ELSE
       dbms_output.put_line('Decrypt BLOB Len ' || leng);
    END IF;
    -- Read decrypblob to a raw
    decrawlen := 999;
    DBMS LOB.OPEN (decrypblob, DBMS LOB.lob readwrite);
    DBMS LOB.READ (decrypblob, decrawlen, 1, decrypraw);
    DBMS LOB.CLOSE (decrypblob);
    dbms_output.put_line('Decrypted data is (256 bit key) ' || decrypraw);
    dbms output.put line('---');
    DBMS_LOB.OPEN (srcblob, DBMS_LOB.lob_readwrite);
    DBMS LOB.TRIM (srcblob, 0);
   DBMS LOB.CLOSE (srcblob);
    DBMS LOB.OPEN (encrypblob, DBMS LOB.lob readwrite);
    DBMS LOB.TRIM (encrypblob, 0);
    DBMS LOB.CLOSE (encrypblob);
    DBMS LOB.OPEN (decrypblob, DBMS LOB.lob readwrite);
    DBMS LOB.TRIM (decrypblob, 0);
    DBMS LOB.CLOSE (decrypblob);
end;
truncate table table lob;
drop table table lob;
```

17.7 Data Dictionary Views for Encrypted Data

Oracle Database provides data dictionary views to find information about encrypted data.

Table 17-5 Data Dictionary Views That Display Information about Encrypted Data

View	Description
ALL_ENCRYPTED_COLUMNS	Describes encryption algorithm information for all encrypted columns in all tables accessible to the user
DBA_ENCRYPTED_COLUMNS	Describes encryption algorithm information for all encrypted columns in the database
USER_ENCRYPTED_COLUMNS	Describes encryption algorithm information for all encrypted columns in all tables in the schema of the user
V\$ENCRYPTED_TABLESPACES	Displays information about the current pluggable database (PDB) tablespaces that are encrypted
V\$ENCRYPTION_WALLET	Displays information on the status of the wallet and the wallet location for Transparent Data Encryption; applies to the current PDB only
V\$RMAN_ENCRYPTION_ALGORITHMS	Displays supported encryption algorithms for the current PDB

Related Topics

Oracle Database Reference

Part IV

Securing Data on the Network

Part IV describes how to secure data on the network.

- Configuring Oracle Database Native Network Encryption and Data Integrity
 You can configure native Oracle Net Services data encryption and data integrity for both servers and clients.
- Configuring the Thin JDBC Client Network
 Oracle Database native network encryption and strong authentication enables thin Java Database Connectivity (JDBC) clients to securely connect to Oracle databases.



Configuring Oracle Database Native Network Encryption and Data Integrity

You can configure native Oracle Net Services data encryption and data integrity for both servers and clients.

- About Oracle Database Native Network Encryption and Data Integrity
 Oracle Database enables you to encrypt data that is sent over a network.
- Oracle Database Native Network Encryption Data Integrity
 Encrypting network data provides data privacy so that unauthorized parties cannot view plaintext data as it passes over the network.
- Improving Native Network Encryption Security
 Oracle provides a patch that will strengthen native network encryption security for both
 Oracle Database servers and clients.
- Data Integrity Algorithms Support
 Data integrity algorithms protect against third-party attacks and message replay attacks.
 Oracle recommends SHA-2, but maintains SHA-1 (deprecated) and MD5 for backward compatibility.
- Diffie-Hellman Based Key Negotiation
 You can use the Diffie-Hellman key negotiation algorithm to secure data in a multiuser environment.
- Configuration of Data Encryption and Integrity
 Oracle Database native Oracle Net Services encryption and integrity presumes the prior
 installation of Oracle Net Services.

18.1 About Oracle Database Native Network Encryption and Data Integrity

Oracle Database enables you to encrypt data that is sent over a network.

- How Oracle Database Native Network Encryption and Integrity Works
 Oracle Database provides native data network encryption and integrity to ensure that data
 is secure as it travels across the network.
- Advanced Encryption Standard
 Oracle Database supports the Federal Information Processing Standard (FIPS) encryption
 algorithm, Advanced Encryption Standard (AES).
- Triple-DES Encryption
 Triple-DES encryption (3DES) encrypts message data with three passes of the DES algorithm.
- Choosing Between Native Network Encryption and Transport Layer Security
 Oracle offers two ways to encrypt data over the network, native network encryption and
 Transport Layer Security (TLS).

18.1.1 How Oracle Database Native Network Encryption and Integrity Works

Oracle Database provides native data network encryption and integrity to ensure that data is secure as it travels across the network.

The purpose of a secure cryptosystem is to convert plaintext data into unintelligible ciphertext based on a key, in such a way that it is very hard (computationally infeasible) to convert ciphertext back into its corresponding plaintext without knowledge of the correct key.

In a symmetric cryptosystem, the same key is used both for encryption and decryption of the same data. Oracle Database provides the Advanced Encryption Standard (AES) symmetric cryptosystem for protecting the confidentiality of Oracle Net Services traffic.

18.1.2 Advanced Encryption Standard

Oracle Database supports the Federal Information Processing Standard (FIPS) encryption algorithm, Advanced Encryption Standard (AES).

AES can be used by all U.S. government organizations and businesses to protect sensitive data over a network. This encryption algorithm defines three standard key lengths, which are 128-bit, 192-bit, and 256-bit. All versions operate in outer Cipher Block Chaining (CBC) mode. CBC mode is an encryption method that protects against block replay attacks by making the encryption of a cipher block dependent on all blocks that precede it; it is designed to make unauthorized decryption incrementally more difficult. Oracle Database employs outer cipher block chaining because it is more secure than inner cipher block chaining, with no material performance penalty.



The AES algorithms have been improved. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

18.1.3 Triple-DES Encryption

Triple-DES encryption (3DES) encrypts message data with three passes of the DES algorithm.



The DES, DES40, 3DES112, and 3DES168 algorithms are deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

3DES provides a high degree of message security, but with a performance penalty. The magnitude of the performance penalty depends on the speed of the processor performing the encryption. 3DES typically takes three times as long to encrypt a data block when compared to the standard DES algorithm.

3DES is available in two-key and three-key versions, with effective key lengths of 112-bits and 168-bits, respectively. Both versions operate in outer Cipher Block Chaining (CBC) mode.

The DES40 algorithm, available with Oracle Database and Secure Network Services, is a variant of DES in which the secret key is preprocessed to provide 40 effective key bits. It was designed to provide DES-based encryption to customers outside the U.S. and Canada at a time when the U.S. export laws were more restrictive. Currently DES40, DES, and 3DES are all available for export. DES40 is still supported to provide backward-compatibility for international customers.

18.1.4 Choosing Between Native Network Encryption and Transport Layer Security

Oracle offers two ways to encrypt data over the network, native network encryption and Transport Layer Security (TLS).

There are advantages and disadvantages to both methods.

Table 18-1 Comparison of Native Network Encryption and Transport Layer Security

-	Native Network Encryption	Transport Layer Security
Advantages	 It is configured with parameters in the sqlnet.ora configuration file. In most cases, no client configuration changes are required. No certificates are required. Clients that do not support native network encryption can fall back to unencrypted connections while incompatibility is mitigated. 	 It is an industry standard for encrypting data in motion. It provides non-repudiation for server connections to prevent third-party attacks. It can be used for database user authentication.
Disadvantages	 It uses a non-standard, Oracle proprietary implementation. It provides no non-repudiation of the server connection (that is, no protection against a third-party attack). 	 It requires client and server changes. Certificates are required for server and are optional for the client. However, the client must have the trusted root certificate for the certificate authority that issued the server's certificate. Certificates eventually expire.

18.2 Oracle Database Native Network Encryption Data Integrity

Encrypting network data provides data privacy so that unauthorized parties cannot view plaintext data as it passes over the network.

Oracle Database also provides protection against two forms of active attacks.

Table 18-2 provides information about these attacks.

Table 18-2 Two Forms of Network Attacks

Type of Attack	Explanation
Data modification attack	An unauthorized party intercepting data in transit, altering it, and retransmitting it is a data modification attack. For example, intercepting a \$100 bank deposit, changing the amount to \$10,000, and retransmitting the higher amount is a data modification attack.



Table 18-2 (Cont.) Two Forms of Network Attacks

Type of Attack	Explanation
Replay attack	Repetitively retransmitting an entire set of valid data is a replay attack, such as intercepting a \$100 bank withdrawal and retransmitting it ten times, thereby receiving \$1,000.

18.3 Improving Native Network Encryption Security

Oracle provides a patch that will strengthen native network encryption security for both Oracle Database servers and clients.

- About Improving Native Network Encryption Security
 The Oracle patch will update encryption and checksumming algorithms and deprecate weak encryption and checksumming algorithms.
- Applying Security Improvement Updates to Native Network Encryption
 In addition to applying a patch to the Oracle Database server and client, you must set the server and client sqlnet.ora parameters.

18.3.1 About Improving Native Network Encryption Security

The Oracle patch will update encryption and checksumming algorithms and deprecate weak encryption and checksumming algorithms.

This patch, which you can download from My Oracle Support note 2118136.2, strengthens the connection between servers and clients, fixing a vulnerability in native network encryption and checksumming algorithms. It adds two parameters that make it easy to disable older, less secure encryption and checksumming algorithms. Oracle strongly recommends that you apply this patch to your Oracle Database server and clients.

This patch applies to Oracle Database releases 11.2 and later. You can apply this patch in the following environments: standalone, multitenant, primary-standby, Oracle Real Application Clusters (Oracle RAC), and environments that use database links.

The supported algorithms that have been improved are as follows:

- Encryption algorithms: AES128, AES192 and AES256
- Checksumming algorithms: SHA1, SHA256, SHA384, and SHA512

Weak algorithms that are deprecated and should not be used after you apply the patch are as follows:

- Encryption algorithms: DES, DES40, 3DES112, 3DES168, RC4_40, RC4_56, RC4_128, and RC4_256
- Checksumming algorithm: MD5

The general procedure that you will follow is to first replace references to desupported algorithms in your Oracle Database environment with supported algorithms, patch the server, patch the client, and finally, set sqlnet.ora parameters to re-enable a proper connection between the server and clients.

The patch affects the following areas including, but not limited to, the following:

JDBC network encryption-related configuration settings

- Encryption and integrity parameters that you have configured using Oracle Net Manager
- Transport Layer Security (TLS) SSL CIPHER SUITE parameter settings
- SecureFiles LOB encrypted columns
- Database Resident Connection Pooling (DRCP) configurations
- Encryption settings used for the configuration of Oracle Call Interface (Oracle OCI), ODP.NET

Related Topics

- Oracle Database JDBC Developer's Guide
- Configuring Encryption and Integrity Parameters Using Oracle Net Manager
 You can set up or change encryption and integrity parameter settings using Oracle Net Manager.
- Choosing Between Native Network Encryption and Transport Layer Security
 Oracle offers two ways to encrypt data over the network, native network encryption and
 Transport Layer Security (TLS).

18.3.2 Applying Security Improvement Updates to Native Network Encryption

In addition to applying a patch to the Oracle Database server and client, you must set the server and client sqlnet.ora parameters.

Ensure that you perform the following steps in the order shown:

- 1. Back up the servers and clients to which you will install the patch.
- 2. Log in to My Oracle Support and then download patch described in My Oracle Support note 2118136.2.

My Oracle Support is located at the following URL:

https://support.oracle.com

Patch the server.

Follow the instructions in My Oracle Support note 2118136.2 to apply the patch to the server. You will apply the same patch to the client in a later step.

4. Patch the clients.

Determine which clients you need to patch.

Follow the instructions in My Oracle Support note 2118136.2 to apply the patch to each client.

5. In each client sqlnet.ora file, remove all deprecated algorithms if they are defined.

You can bypass this step if the following parameters are not defined or have no algorithms listed.

- SQLNET.ENCRYPTION TYPES CLIENT
- SQLNET.CRYPTO CHECKSUM TYPES CLIENT
- 6. In the server sqlnet.ora file, remove all deprecated algorithms if they are defined.

You can bypass this step if the following parameters are not defined or have no algorithms listed.

SQLNET.ENCRYPTION TYPES SERVER



- SQLNET.CRYPTO CHECKSUM TYPES SERVER
- 7. For maximum security on the server, set the following sqlnet.ora parameters:
 - SQLNET.ENCRYPTION SERVER = REQUIRED
 - SQLNET.ENCRYPTION TYPES SERVER = (AES256)
 - SQLNET.CRYPTO CHECKSUM SERVER = REQUIRED
 - SQLNET.CRYPTO CHECKSUM TYPES SERVER = (SHA512)
 - SQLNET.ALLOW WEAK CRYPTO CLIENTS = FALSE
- 8. For maximum security on the client, set the following sqlnet.ora parameters:
 - SQLNET.ENCRYPTION_CLIENT = REQUIRED
 - SQLNET.ENCRYPTION TYPES CLIENT = (AES256)
 - SQLNET.CRYPTO_CHECKSUM_CLIENT = REQUIRED
 - SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (SHA512)
 - SQLNET.ALLOW WEAK CRYPTO = FALSE
- 9. In each client sqlnet.ora file, after you have removed all the deprecated algorithms from the server and the clients per steps 5 and 6, set the parameter SQLNET.ALLOW_WEAK_CRYPTO = FALSE so that the clients can be prevented from communicating with unpatched servers.

If the SQLNET.ALLOW_WEAK_CRYPTO parameter is set to FALSE, then a client attempting to use a weak algorithm will produce an ORA-12269: client uses weak encryption/crypto-checksumming version error at the server. A client connecting to a server (or proxy) that is using weak algorithms will receive an ORA-12268: server uses weak encryption/crypto-checksumming version error.

10. In the server sqlnet.ora file, after you have updated all the clients with SQLNET.ALLOW_WEAK_CRYPTO = FALSE per step 9, set the parameter SQLNET.ALLOW_WEAK_CRYPTO_CLIENTS = FALSE. This parameter prevents a patched server from communicating with unpatched clients.

If the SQLNET.ALLOW_WEAK_CRYPTO parameter is set to FALSE, then a client attempting to use a weak algorithm will produce an ORA-12269: client uses weak encryption/crypto-checksumming version error at the server. A client connecting to a server (or proxy) that is using weak algorithms will receive an ORA-12268: server uses weak encryption/crypto-checksumming version error.

If you use the database links, then the first database server acts as a client and connects to the second server. Therefore, ensure that all servers are fully patched and unsupported algorithms are removed before you set SQLNET.ALLOW WEAK CRYPTO to FALSE

18.4 Data Integrity Algorithms Support

Data integrity algorithms protect against third-party attacks and message replay attacks. Oracle recommends SHA-2, but maintains SHA-1 (deprecated) and MD5 for backward compatibility.



Note:

MD5 is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

These hashing algorithms create a checksum that changes if the data is altered in any way. This protection operates independently from the encryption process so you can enable data integrity with or without enabling encryption.

Related Topics

Configuring Integrity on the Client and the Server
You can use Oracle Net Manager to configure network integrity on both the client and the
server.

18.5 Diffie-Hellman Based Key Negotiation

You can use the Diffie-Hellman key negotiation algorithm to secure data in a multiuser environment.

Secure key distribution is difficult in a multiuser environment. Oracle Database uses the well known Diffie-Hellman key negotiation algorithm to perform secure key distribution for both encryption and data integrity.

When encryption is used to protect the security of encrypted data, keys must be changed frequently to minimize the effects of a compromised key. Accordingly, the Oracle Database key management function changes the session key with every session.

The Diffie-Hellman key negotiation algorithm is a method that lets two parties communicating over an insecure channel to agree upon a random number known only to them. Oracle Database uses the Diffie-Hellman key negotiation algorithm to generate session keys.

The client and the server begin communicating using the session key generated by Diffie-Hellman. When the client authenticates to the server, they establish a shared secret that is only known to both parties. Oracle Database combines the shared secret and the Diffie-Hellman session key to generate a stronger session key designed to defeat a third-party attack.

Note:

Oracle recommends that you use the more secure authenticated connections available with Oracle Database. If you use anonymous Diffie-Hellman with RC4 for connecting to Oracle Internet Directory for Enterprise User Security, then you must migrate to use a different algorithm connection. Oracle recommends that you use either TLS one-way, or mutual authentication using certificates.

18.6 Configuration of Data Encryption and Integrity

Oracle Database native Oracle Net Services encryption and integrity presumes the prior installation of Oracle Net Services.

- About Activating Encryption and Integrity
 - In any network connection, both the client and server can support multiple encryption algorithms and integrity algorithms.
- About Negotiating Encryption and Integrity
 - The sqlnet.ora file on systems using data encryption and integrity must contain some or all the REJECTED, ACCEPTED, REQUESTED, and REQUIRED parameters.
- Configuring Encryption and Integrity Parameters Using Oracle Net Manager
 You can set up or change encryption and integrity parameter settings using Oracle Net Manager.

18.6.1 About Activating Encryption and Integrity

In any network connection, both the client and server can support multiple encryption algorithms and integrity algorithms.

When a connection is made, the server selects which algorithm to use, if any, from those algorithms specified in the sqlnet.ora files. The server searches for a match between the algorithms available on both the client and the server, and picks the first algorithm in its own list that also appears in the client list. If one side of the connection does not specify an algorithm list, all the algorithms installed on that side are acceptable. The connection fails with error message ORA-12650 if either side specifies an algorithm that is not installed.

Encryption and integrity parameters are defined by modifying a sqlnet.ora file on the clients and the servers on the network.

You can choose to configure any or all of the available encryption algorithms, and either or both of the available integrity algorithms. Only one encryption algorithm and one integrity algorithm are used for each connect session.

Note:

Oracle Database selects the first encryption algorithm and the first integrity algorithm enabled on the client and the server. Oracle recommends that you select algorithms and key lengths in the order in which you prefer negotiation, choosing the strongest key length first.

See Also:

- Table 18-4 for a listing of valid encryption algorithms
- Oracle Database Advanced Security Guide for a listing of available integrity algorithms
- Data Encryption and Integrity Parameters



18.6.2 About Negotiating Encryption and Integrity

The sqlnet.ora file on systems using data encryption and integrity must contain some or all the REJECTED, ACCEPTED, REQUESTED, and REQUIRED parameters.

- About the Values for Negotiating Encryption and Integrity
 - Oracle Net Manager can be used to specify four possible values for the encryption and integrity configuration parameters.
- REJECTED Configuration Parameter

The REJECTED value disables the security service, even if the other side requires this service.

ACCEPTED Configuration Parameter

The ACCEPTED value enables the security service if the other side requires or requests the service.

REQUESTED Configuration Parameter

The REQUESTED value enables the security service if the other side permits this service.

REQUIRED Configuration Parameter

The REQUIRED value enables the security service or preclude the connection.

18.6.2.1 About the Values for Negotiating Encryption and Integrity

Oracle Net Manager can be used to specify four possible values for the encryption and integrity configuration parameters.

The following four values are listed in the order of increasing security, and they must be used in the profile file (sqlnet.ora) for the client and server of the systems that are using encryption and integrity.

The value REJECTED provides the *minimum* amount of security between client and server communications, and the value REQUIRED provides the *maximum* amount of network security:

- REJECTED
- ACCEPTED
- REQUESTED
- REQUIRED

The default value for each of the parameters is ACCEPTED.

Oracle Database servers and clients are set to ACCEPT encrypted connections out of the box. This means that you can enable the desired encryption and integrity settings for a connection pair by configuring just one side of the connection, server-side or client-side.

So, for example, if there are many Oracle clients connecting to an Oracle database, you can configure the required encryption and integrity settings for all these connections by making the appropriate sqlnet.ora changes at the server end. You do not need to implement configuration changes for each client separately.

Table 18-3 shows whether the security service is enabled, based on a combination of client and server configuration parameters. If either the server or client has specified REQUIRED, the lack of a common algorithm causes the connection to fail. Otherwise, if the service is enabled, lack of a common service algorithm results in the service being disabled.



Table 18-3 Encryption and Data Integrity Negotiations

Client Setting	Server Setting	Encryption and Data Negotiation
REJECTED	REJECTED	OFF
ACCEPTED	REJECTED	OFF
REQUESTED	REJECTED	OFF
REQUIRED	REJECTED	Connection fails
REJECTED	ACCEPTED	OFF
ACCEPTED	ACCEPTED	OFF ¹
REQUESTED	ACCEPTED	ON
REQUIRED	ACCEPTED	ON
REJECTED	REQUESTED	OFF
ACCEPTED	REQUESTED	ON
REQUESTED	REQUESTED	ON
REQUIRED	REQUESTED	ON
REJECTED	REQUIRED	Connection fails
ACCEPTED	REQUIRED	ON
REQUESTED	REQUIRED	ON
REQUIRED	REQUIRED	ON

¹ This value defaults to OFF. Cryptography and data integrity are not enabled until the user changes this parameter by using Oracle Net Manager or by modifying the sqlnet.ora file.

18.6.2.2 REJECTED Configuration Parameter

The REJECTED value disables the security service, even if the other side requires this service.

In this scenario, this side of the connection specifies that the security service is not permitted. If the other side is set to REQUIRED, the connection *terminates* with error message ORA-12650. If the other side is set to REQUESTED, ACCEPTED, or REJECTED, the connection continues without error and without the security service enabled.

18.6.2.3 ACCEPTED Configuration Parameter

The ACCEPTED value enables the security service if the other side requires or requests the service.

In this scenario, this side of the connection does not require the security service, but it is enabled if the other side is set to REQUIRED or REQUESTED. If the other side is set to REQUIRED or REQUESTED, and an encryption or integrity algorithm match is found, the connection continues without error and with the security service enabled. If the other side is set to REQUIRED and no algorithm match is found, the connection terminates with error message ORA-12650.

If the other side is set to REQUESTED and no algorithm match is found, or if the other side is set to ACCEPTED or REJECTED, the connection continues without error and without the security service enabled.

18.6.2.4 REQUESTED Configuration Parameter

The REQUESTED value enables the security service if the other side permits this service.

In this scenario, this side of the connection specifies that the security service is desired but not required. The security service is enabled if the other side specifies ACCEPTED, REQUESTED, or REQUIRED. There must be a matching algorithm available on the other side, otherwise the service is not enabled. If the other side specifies REQUIRED and there is no matching algorithm, the connection fails.

18.6.2.5 REQUIRED Configuration Parameter

The REQUIRED value enables the security service or preclude the connection.

In this scenario, this side of the connection specifies that the security service must be enabled. The connection fails if the other side specifies REJECTED or if there is no compatible algorithm on the other side.

18.6.3 Configuring Encryption and Integrity Parameters Using Oracle Net Manager

You can set up or change encryption and integrity parameter settings using Oracle Net Manager.

- Configuring Encryption on the Client and the Server
 Use Oracle Net Manager to configure encryption on the client and on the server.
- Configuring Integrity on the Client and the Server
 You can use Oracle Net Manager to configure network integrity on both the client and the
 server.
- Enabling Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently

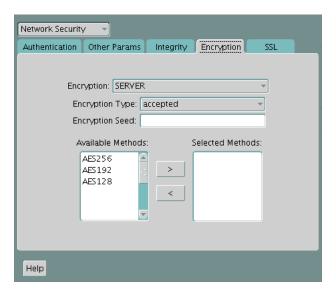
Depending on the SQLNET.ENCRYPTION_CLIENT and SQLNET.ENCRYPTION_SERVER settings, you can configure Oracle Database to allow both Oracle native encryption and SSL authentication for different users concurrently.

18.6.3.1 Configuring Encryption on the Client and the Server

Use Oracle Net Manager to configure encryption on the client and on the server.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- **3.** From the **Naming** list, select **Network Security**.
 - The Network Security tabbed window appears.
- 4. Select the **Encryption** tab.





- Select CLIENT or SERVER option from the Encryption box.
- 6. From the Encryption Type list, select one of the following:
 - REQUESTED
 - REQUIRED
 - ACCEPTED
 - REJECTED
- 7. (Optional) In the **Encryption Seed** field, enter between 10 and 70 random characters. The encryption seed for the client should not be the same as that for the server.
- 8. Select an encryption algorithm in the **Available Methods** list. Move it to the **Selected Methods** list by choosing the right arrow (>). Repeat for each additional method you want to use.
- 9. Select File, Save Network Configuration. The sqlnet.ora file is updated.
- 10. Repeat this procedure to configure encryption on the other system. The sqlnet.ora file on the two systems should contain the following entries:
 - On the server:

```
SQLNET.ENCRYPTION_SERVER = [accepted | rejected | required]
SQLNET.ENCRYPTION_TYPES_SERVER = (valid_encryption_algorithm
[,valid_encryption_algorithm])
```

On the client:

```
SQLNET.ENCRYPTION_CLIENT = [accepted | rejected | required]
SQLNET.ENCRYPTION_TYPES_CLIENT = (valid_encryption_algorithm
[,valid encryption algorithm])
```

Table 18-4 lists valid encryption algorithms and their associated legal values.

Table 18-4 Valid Encryption Algorithms

Algorithm Name	Legal Value
AES 256-bit key	AES256
AES 192-bit key	AES192



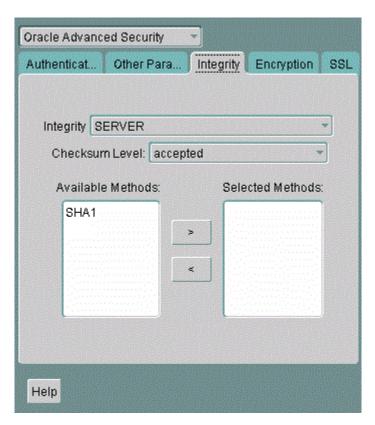
Table 18-4 (Cont.) Valid Encryption Algorithms

Algorithm Name	Legal Value
AES 128-bit key	AES128

18.6.3.2 Configuring Integrity on the Client and the Server

You can use Oracle Net Manager to configure network integrity on both the client and the server.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.The Network Security tabbed window appears.
- 4. Select the Integrity tab.



- Depending upon which system you are configuring, select the Server or Client from the Integrity box.
- From the Checksum Level list, select one of the following checksum level values:

- REQUESTED
- REQUIRED
- ACCEPTED
- REJECTED
- Select an integrity algorithm in the Available Methods list. Move it to the Selected
 Methods list by choosing the right arrow (>). Repeat for each additional method you want
 to use.
- Select File, Save Network Configuration.

The sqlnet.ora file is updated.

9. Repeat this procedure to configure integrity on the other system.

The sqlnet.ora file on the two systems should contain the following entries:

On the server:

```
SQLNET.CRYPTO_CHECKSUM_SERVER = [accepted | rejected | required]
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (valid_crypto_checksum_algorithm
[,valid_crypto_checksum_algorithm])
```

On the client:

```
SQLNET.CRYPTO_CHECKSUM_CLIENT = [accepted | rejected | required]
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (valid_crypto_checksum_algorithm
[,valid_crypto_checksum_algorithm])
```

Valid integrity/checksum algorithms that you can use are as follows:

- SHA1
- SHA256
- SHA384
- SHA512

Related Topics

Oracle Database Advanced Security Guide

18.6.3.3 Enabling Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently

Depending on the SQLNET.ENCRYPTION_CLIENT and SQLNET.ENCRYPTION_SERVER settings, you can configure Oracle Database to allow both Oracle native encryption and SSL authentication for different users concurrently.

- About Enabling Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently
 - By default, Oracle Database does not allow both Oracle native encryption and Transport Layer Security (SSL) authentication for different users concurrently.
- Configuring Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently

Use the <code>IGNORE_ANO_ENCRYPTION_FOR_TCPS</code> parameter to enable the concurrent use of both Oracle native encryption and Transport Layer Security (SSL) authentication.



18.6.3.3.1 About Enabling Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently

By default, Oracle Database does not allow both Oracle native encryption and Transport Layer Security (SSL) authentication for different users concurrently.

The use of both Oracle native encryption (also called Advanced Networking Option (ANO) encryption) and TLS authentication together is called double encryption.

There are cases in which both a TCP and TCPS listener must be configured, so that some users can connect to the server using a user name and password, and others can validate to the server by using a TLS certificate. In these situations, you must configure both password-based authentication and TLS authentication. A workaround in previous releases was to set the SQLNET.ENCRYPTION_SERVER parameter to requested. If your requirements are that SQLNET.ENCRYPTION_SERVER be set to required, then you can set the IGNORE_ANO_ENCRYPTION_FOR_TCPS parameter in both SQLNET.ENCRYPTION_CLIENT and SQLNET.ENCRYPTION SERVER to TRUE. By default, it is set to FALSE.

Setting IGNORE_ANO_ENCRYPTION_FOR_TCPS to TRUE forces the client to ignore the value that is set for the SQLNET.ENCRYPTION_CLIENT parameter for all outgoing TCPS connections. This parameter allows the database to ignore the SQLNET.ENCRYPTION_CLIENT or SQLNET.ENCRYPTION_SERVER setting when there is a conflict between the use of a TCPS client and when these two parameters are set to required.

18.6.3.3.2 Configuring Both Oracle Native Encryption and SSL Authentication for Different Users Concurrently

Use the IGNORE_ANO_ENCRYPTION_FOR_TCPS parameter to enable the concurrent use of both Oracle native encryption and Transport Layer Security (SSL) authentication.

On the server, you must set IGNORE_ANO_ENCRYPTION_FOR_TCPS in the sqlnet.ora file, and on the client, you can set it in either the sqlnet.ora file or the tnsnames.ora file.

- 1. Log in to the database server
- 2. Go to the location of the sqlnet.ora file.

By default, sqlnet.ora is in the <code>ORACLE_BASE/network/admin</code> directory. The sqlnet.ora file can also be stored in the directory specified by the <code>TNS ADMIN</code> environment variable.

- 3. In sqlnet.ora, check if the current SQLNET.ENCRYPTION_SERVER setting is required or requested.
- 4. If SQLNET.ENCRYPTION_SERVER is set to required, then add the SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS to sqlnet.ora and then set it to TRUE. IGNORE_ANO_ENCRYPTION_FOR_TCPS=TRUE
- 5. Save and exit sqlnet.ora.
- 6. Log in to the client.

For the client, you can set the value in either the sqlnet.ora file or the tnsnames.ora file.

• Setting the value in sqlnet.ora: Check if the SQLNET.ENCRYPTION_CLIENT parameter is set to required. If SQLNET.ENCRYPTION_CLIENT, then edit the sqlnet.ora file to have the following setting:

IGNORE_ANO_ENCRYPTION_FOR_TCPS=TRUE



• Setting the value in thsnames.ora: By default, thsnames.ora is in the same location as sqlnet.ora. If SQLNET.ENCRYPTION_CLIENT is set to required in sqlnet.ora, then in the SECURITY portion of the TNS_ALIAS setting, set IGNORE_ANO_ENCRYPTION_FOR_TCPS=TRUE. For example:



Configuring the Thin JDBC Client Network

Oracle Database native network encryption and strong authentication enables thin Java Database Connectivity (JDBC) clients to securely connect to Oracle databases.

About the Java Implementation

Oracle Database provides a Java implementation of native network encryption and strong authentication.

Java Database Connectivity Support

JDBC, an industry-standard Java interface, is a Java standard for connecting to a relational database from a Java program.

Thin JDBC Features

The Thin JDBC driver provides security features such as strong authentication, data encryption, and data integrity checking.

Implementation Overview

On the server side, the negotiation of algorithms and the generation of keys function exactly the same as Oracle Database native encryption.

Obfuscation of the Java Cryptography Code

The obfuscation of the Java cryptography code protects Java classes and methods that contain encryption and decryption capabilities with obfuscation software.

Configuration Parameters for the Thin JDBC Network Implementation

The Thin JDBC network implementation for the client provides parameters to control encryption, integrity, and the authentication service.

19.1 About the Java Implementation

Oracle Database provides a Java implementation of native network encryption and strong authentication.

The Java implementation of Oracle Database native network encryption and strong authentication provides network authentication, encryption and integrity protection for Thin JDBC clients that must communicate with Oracle Databases that have Oracle Database native network encryption and strong authentication configured.



Oracle Database JDBC Developer's Guide for information about JDBC, including examples

19.2 Java Database Connectivity Support

JDBC, an industry-standard Java interface, is a Java standard for connecting to a relational database from a Java program.

Sun Microsystems defined the JDBC standard and Oracle implements and extends the standard with its own JDBC drivers.

Oracle JDBC drivers are used to create Java Database Connectivity (JDBC) applications to communicate with Oracle databases. Oracle implements two types of JDBC drivers: Thick JDBC drivers built on top of the C-based Oracle Net client, as well as a Thin (Pure Java) JDBC driver to support downloadable applets. Oracle extensions to JDBC include the following features:

- Data access and manipulation
- LOB access and manipulation
- Oracle object type mapping
- Object reference access and manipulation
- Array access and manipulation
- Application performance enhancement

19.3 Thin JDBC Features

The Thin JDBC driver provides security features such as strong authentication, data encryption, and data integrity checking.

Because the Thin JDBC driver is designed to be used with downloadable applets used over the Internet, Oracle designed a 100 percent Java implementation of Oracle Database native network encryption and strong authentication, encryption, and integrity algorithms, for use with thin clients.

Oracle Database provides the following features for Thin JDBC:

- Strong Authentication
- Data encryption
- Data integrity checking
- Secure connections from Thin JDBC clients to the Oracle RDBMS
- Ability for developers to build applets that transmit data over a secure communication channel
- Secure connections from middle tier servers with Java Server Pages (JSP) to the Oracle RDBMS
- Secure connections from the current release of Oracle Database to older versions of Oracle databases

The Oracle JDBC Thin driver supports the Oracle Database SSL implementation and third-party authentication methods such as RADIUS and Kerberos. Thin JDBC support for authentication methods like RADIUS, Kerberos, and SSL were introduced in Oracle Database 11g release 1 (11.1).

The Oracle Database native network encryption and strong authentication Java implementation provides Java versions of the following encryption algorithms:

- AES256: AES 256-bit key
- AES192: AES 192-bit key
- AES128: AES 128-bit key



Note:

In the preceding list of algorithms, CBC refers to the Cipher Block Chaining mode.

Thin JDBC support for the Advanced Encryption Standard (AES) was introduced in Oracle Database 12c Release 1 (12.1).

In addition, this implementation provides data integrity checking for Thin JDBC using Secure Hash Algorithm (SHA1) and Message Digest 5 (MD5). Thin JDBC support for SHA1 was introduced in Oracle Database 11*g* release 1 (11.1).

See Also:

Oracle Database JDBC Developer's Guide for details on configuring authentication, encryption, and integrity for thin JDBC clients.

Note:

MD5 is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

19.4 Implementation Overview

On the server side, the negotiation of algorithms and the generation of keys function exactly the same as Oracle Database native encryption.

This feature enables backward and forward compatibility of clients and servers.

On the client side, the algorithm negotiation and key generation occur in exactly the same manner as OCI clients. The client and server negotiate encryption algorithms, generate random numbers, use Diffie-Hellman to exchange session keys, and use the Oracle Password Protocol, in the same manner as the traditional Oracle Net clients. Thin JDBC contains a complete implementation of an Oracle Net client in pure Java.

19.5 Obfuscation of the Java Cryptography Code

The obfuscation of the Java cryptography code protects Java classes and methods that contain encryption and decryption capabilities with obfuscation software.

Java byte code obfuscator is a process frequently used to protect intellectual property written in the form of Java programs. It mixes up Java symbols found in the code. The process leaves the original program structure intact, letting the program run correctly while changing the names of the classes, methods, and variables in order to hide the intended behavior. Although it is possible to decompile and read non-obfuscated Java code, obfuscated Java code is sufficiently difficult to decompile to satisfy U.S. government export controls.



19.6 Configuration Parameters for the Thin JDBC Network Implementation

The Thin JDBC network implementation for the client provides parameters to control encryption, integrity, and the authentication service.

- About the Thin JDBC Network Implementation Configuration Parameters
 The JDBC network implementation configuration parameters control network settings such as the level of security used between client and server connections.
- Client Encryption Level Parameter

The CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_LEVEL parameter defines the level of security that the client uses to negotiate with the server.

- Client Encryption Selected List Parameter

 The CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_TYPES parameter defines the encryption algorithm to be used.
- Client Integrity Level Parameter
 The CONNECTION_PROPERTY_THIN_NET_CHECKSUM_LEVEL parameter defines the level of security to negotiate with the server for data integrity.
- Client Integrity Selected List Parameter
 The CONNECTION_PROPERTY_THIN_NET_CHECKSUM_TYPES parameter defines the data integrity algorithm to be used.
- Client Authentication Service Parameter
 The CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_SERVICES parameter determines
 the authentication service to be used.
- AnoServices Constants
 The oracle.net.ano.AnoServices interface includes the names of the encryption, authentication, and checksum algorithms that the JDBC Thin driver supports.

19.6.1 About the Thin JDBC Network Implementation Configuration Parameters

The JDBC network implementation configuration parameters control network settings such as the level of security used between client and server connections.

A properties class object containing several configuration parameters is passed to the Oracle Database native network encryption and strong authentication interface.

All JDBC connection properties including the ones pertaining to Oracle Database are defined as constants in the <code>oracle.jdbc.OracleConnection</code> interface. The following list enumerates some of these connection properties:



Oracle Database JDBC Developer's Guide for detailed information on configuration parameters and configuration examples



19.6.2 Client Encryption Level Parameter

The CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_LEVEL parameter defines the level of security that the client uses to negotiate with the server.

Table 19-1 describes the attributes of this parameter.

Table 19-1 CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_LEVEL Attributes

Attribute	Description
Parameter Type	String
Parameter Class	Static
Permitted Values	REJECTED; ACCEPTED; REQUESTED; REQUIRED
Default Value	ACCEPTED
Syntax	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_ENCRYPTION_LEVEL, level);</pre>
	where prop is an object of the Properties class
Example	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_ENCRYPTION_LEVEL, "REQUIRED"); where prop is an object of the Properties class</pre>

19.6.3 Client Encryption Selected List Parameter

The CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_TYPES parameter defines the encryption algorithm to be used.

Table 19-2 describes attributes of this parameter.

Table 19-2 CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_TYPES Attributes

Attribute	Description
Parameter Type	String
Parameter Class	Static
Permitted Values	AES256 (AES 256-bit key), AES192 (AES 192-bit key), AES128 (AES 128-bit key),
Syntax	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_ENCRYPTION_TYPES, algorithm);</pre>
	where prop is an object of the Properties class
Example	prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_ENCRYPTION_TYPES, "(AES256, AES192)");
	where prop is an object of the Properties class

19.6.4 Client Integrity Level Parameter

The <code>CONNECTION_PROPERTY_THIN_NET_CHECKSUM_LEVEL</code> parameter defines the level of security to negotiate with the server for data integrity.

Table 19-3 describes the attributes of this parameter.



Table 19-3 CONNECTION_PROPERTY_THIN_NET_CHECKSUM_LEVEL Attributes

Attribute	Description
Parameter Type	String
Parameter Class	Static
Permitted Values	REJECTED; ACCEPTED; REQUESTED; REQUIRED
Default Value	ACCEPTED
Syntax	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_CHECKSUM_LEVEL, level);</pre>
	where prop is an object of the Properties class
Example	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_CHECKSUM_LEVEL,"REQUIRED");</pre>
	where prop is an object of the Properties class

19.6.5 Client Integrity Selected List Parameter

The CONNECTION_PROPERTY_THIN_NET_CHECKSUM_TYPES parameter defines the data integrity algorithm to be used.

Table 19-4 describes this parameter's attributes.

Table 19-4 CONNECTION_PROPERTY_THIN_NET_CHECKSUM_TYPES Attributes

Attribute	Description
Parameter Type	String
Parameter Class	Static
Permitted Values	SHA1
Syntax	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_CHECKSUM_TYPES, algorithm);</pre>
	where prop is an object of the Properties class
Example	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_CHECKSUM_TYPES,"(SHA1)");</pre>
	where prop is an object of the Properties class

19.6.6 Client Authentication Service Parameter

The CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_SERVICES parameter determines the authentication service to be used.

Table 19-5 describes this parameter's attributes.

Table 19-5 CONNECTION_PROPERTY_THIN_NET_AUTHENTICATION_SERVICES Attributes

Attribute	Description
Parameter Type	String



Table 19-5 (Cont.)
CONNECTION PROPERTY THIN NET AUTHENTICATION SERVICES Attributes

Attribute	Description
Parameter Class	Static
Permitted Values	RADIUS, KERBEROS, SSL
Syntax	<pre>prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_AUTHENTICATION_SERVICES, authentication); where prop is an object of the Properties class</pre>
Example	prop.setProperty(OracleConnection.CONNECTION_PROPERTY_T HIN_NET_AUTHENTICATION_SERVICES,"(RADIUS, KERBEROS, SSL)"); where prop is an object of the Properties class

19.6.7 AnoServices Constants

The oracle.net.ano.AnoServices interface includes the names of the encryption, authentication, and checksum algorithms that the JDBC Thin driver supports.

The following constants are in the oracle.net.ano.AnoServices interface:

```
// ---- SUPPORTED ENCRYPTION ALG -----
public static final String ENCRYPTION RC4 40 = "RC4 40";
public static final String ENCRYPTION RC4 56 = "RC4 56";
public static final String ENCRYPTION RC4 128 = "RC4 128";
public static final String ENCRYPTION RC4 256 = "RC4 256";
public static final String ENCRYPTION DES40C = "DES40C";
public static final String ENCRYPTION DES56C = "DES56C";
public static final String ENCRYPTION 3DES112 = "3DES112";
public static final String ENCRYPTION 3DES168 = "3DES168";
public static final String ENCRYPTION AES128 = "AES128";
public static final String ENCRYPTION AES192 = "AES192";
public static final String ENCRYPTION AES256 = "AES256";
// ---- SUPPORTED INTEGRITY ALG ----
public static final String CHECKSUM MD5 = "MD5";
public static final String CHECKSUM SHA1 = "SHA1";
// ---- SUPPORTED AUTHENTICATION ADAPTORS ----
public static final String AUTHENTICATION RADIUS = "RADIUS";
public static final String AUTHENTICATION KERBEROS = "KERBEROS";
```

Note:

The DES40, 3DES112, 3DES168, MD5, RC4_40, RC4_56, RC4_128, and RC4-256 algorithms are deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

You can use these constants to set the encryption, integrity, and authentication parameters. Example 19-1 illustrates one such scenario.

Example 19-1 Using AnoServices Constants in JDBC Client Code

```
import java.sql.*;
import java.util.Properties;import oracle.jdbc.*;
import oracle.net.ano.AnoServices;
* JDBC thin driver demo: new security features in 11gR1.
^{\star} This program attempts to connect to the database using the JDBC thin
* driver and requires the connection to be encrypted with either AES256 or AES192
* and the data integrity to be verified with SHA1.
^{\star} In order to activate encryption and checksumming in the database you need to
* modify the sqlnet.ora file. For example:
    SQLNET.ENCRYPTION TYPES SERVER = (AES256, AES192, AES128)
    SQLNET.ENCRYPTION SERVER = accepted
    SQLNET.CRYPTO CHECKSUM TYPES SERVER= (SHA1)
    SQLNET.CRYPTO CHECKSUM SERVER = accepted
* This output of this program is:
   Connection created! Encryption algorithm is: AES256, data integrity algorithm
    is: SHA1
* /
public class DemoAESAndSHA1
 static final String USERNAME= "hr";
 static final String PASSWORD= "hr";
 static final String URL =
"jdbc:oracle:thin:@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=somehost.us.example.com)
(PORT=5561))"
+"(CONNECT DATA=(SERVICE NAME=itydemo.regress.rdbms.dev.us.example.com)))";
 public static final void main(String[] argv)
    DemoAESAndSHA1 demo = new DemoAESAndSHA1();
    try
    {
     demo.run();
    }catch(SQLException ex)
      ex.printStackTrace();
  }
 void run() throws SQLException
   OracleDriver dr = new OracleDriver();
   Properties prop = new Properties();
    // We require the connection to be encrypted with either AES256 or AES192.
    // If the database doesn't accept such a security level, then the connection
    // attempt will fail.
   prop.setProperty(
OracleConnection.CONNECTION_PROPERTY_THIN_NET_ENCRYPTION_LEVEL,AnoServices.ANO_REQUIRED);
   prop.setProperty(
     OracleConnection.CONNECTION PROPERTY THIN NET ENCRYPTION TYPES,
AnoServices.ENCRYPTION AES256 + "," +AnoServices.ENCRYPTION AES192 + ")");
    // We also require the use of the SHA1 algorithm for data integrity checking.
   prop.setProperty(
OracleConnection.CONNECTION PROPERTY THIN NET CHECKSUM LEVEL, AnoServices.ANO REQUIRED);
```



Part V

Managing Strong Authentication

Part V describes how to manage strong authentication.

- Introduction to Strong Authentication
 Strong authentication supports tools such as Transport Layer Security (TLS) to verify the identities of users who log in to the database.
- Strong Authentication Administration Tools
 You can use a set of strong authentication administration tools for native network
 encryption and public key infrastructure credentials.
- Configuring Kerberos Authentication
 Kerberos is a trusted third-party authentication system that relies on shared secrets and presumes that the third party is secure.
- Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.
- Configuring RADIUS Authentication
 RADIUS is a client/server security protocol widely used to enable remote authentication
 and access.
- Customizing the Use of Strong Authentication
 You can configure multiple authentication methods under Oracle Database native network
 encryption and strong authentication.



Introduction to Strong Authentication

Strong authentication supports tools such as Transport Layer Security (TLS) to verify the identities of users who log in to the database.

- What Is Strong Authentication?
 You use authentication to prove the identities of users who are attempting to log into the database.
- Centralized Authentication and Single Sign-On
 Single sign-on enables users to access multiple accounts and applications with a single password.
- How Centralized Network Authentication Works
 A centralized network authentication system works with an Oracle server, an authentication server, and users who connect to the Oracle server.
- Supported Strong Authentication Methods
 Oracle Database supports industry-standard authentication methods.
- Oracle Database Native Network Encryption/Strong Authentication Architecture
 The Oracle Database native network encryption and strong authentication architecture
 complements an Oracle database server or client installations.
- System Requirements for Strong Authentication
 Kerberos, RADIUS, and Transport Layer Security (TLS) have a set of system requirements
 for strong authentication.
- Oracle Database Native Network Encryption and Strong Authentication Restrictions
 Oracle applications support Oracle Database native network encryption and strong
 authentication.

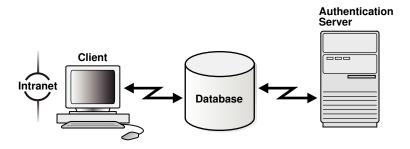
20.1 What Is Strong Authentication?

You use authentication to prove the identities of users who are attempting to log into the database.

Authenticating user identity is imperative in distributed environments, without which there can be little confidence in network security. Passwords are the most common means of authentication. Oracle Database enables strong authentication with Oracle authentication adapters that support various third-party authentication services, including SSL with digital certificates.

Figure 20-1 shows user authentication with an Oracle database instance configured to use a third-party authentication server. Having a central facility to authenticate all members of the network (clients to servers, servers to servers, users to both clients and servers) is one effective way to address the threat of network nodes falsifying their identities.

Figure 20-1 Strong Authentication with Oracle Authentication Adapters



20.2 Centralized Authentication and Single Sign-On

Single sign-on enables users to access multiple accounts and applications with a single password.

Centralized authentication also provides the benefit of single sign-on (SSO) for users.

In single sign-on, a user only needs to login once and can then automatically connect to any other service without having to give the user name and password again. Single sign-on eliminates the need for the user to remember and administer multiple passwords, reducing the time spent logging into multiple services.

20.3 How Centralized Network Authentication Works

A centralized network authentication system works with an Oracle server, an authentication server, and users who connect to the Oracle server.

The following diagram shows how a centralized network authentication service typically operates.

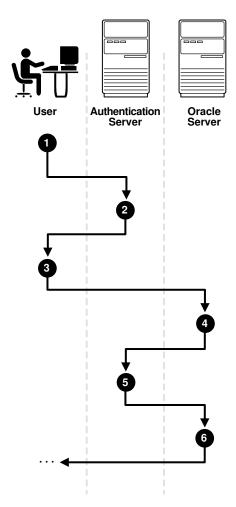


Figure 20-2 How a Network Authentication Service Authenticates a User

The following steps describe how centralized Network Authentication Process works.

- A user (client) requests authentication services and provides identifying information, such as a token or password.
- 2. The authentication server validates the user's identity and passes a ticket or credentials back to the client, which may include an expiration time.
- The client passes these credentials to the Oracle server concurrent with a service request, such as connection to a database.
- 4. The server sends the credentials back to the authentication server for authentication.
- 5. The authentication server checks the credentials and notifies the Oracle server.
- 6. If the credentials were accepted by the authentication server, then the Oracle server authenticates the user. If the authentication server rejected the credentials, then authentication fails, and the service request is denied.

20.4 Supported Strong Authentication Methods

Oracle Database supports industry-standard authentication methods.

About Kerberos

Oracle Database support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users.

- About Remote Authentication Dial-In User Service (RADIUS)
 RADIUS is a client/server security protocol that is most widely known for enabling remote authentication and access.
- About Transport Layer Security
 Transport Layer Security (TLS) is an industry standard protocol for securing network connections.

20.4.1 About Kerberos

Oracle Database support for Kerberos provides the benefits of single sign-on and centralized authentication of Oracle users.

Kerberos is a trusted third-party authentication system that relies on shared secrets. It presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does this through a Kerberos authentication server. Refer to Configuring Kerberos Authentication for information about configuring and using this adapter.



Oracle authentication for Kerberos provides database link authentication (also called proxy authentication). Kerberos is also an authentication method that is supported with Enterprise User Security.

20.4.2 About Remote Authentication Dial-In User Service (RADIUS)

RADIUS is a client/server security protocol that is most widely known for enabling remote authentication and access.

Oracle Database uses this standard in a client/server network environment to enable use of any authentication method that supports the RADIUS protocol. RADIUS can be used with a variety of authentication mechanisms, including token cards and smart cards.

- Smart Cards. A RADIUS-compliant smart card is a credit card-like hardware device which
 has memory and a processor. It is read by a smart card reader located at the client
 workstation.
- Token Cards. Token cards (Secure ID or RADIUS-compliant) can improve ease of use through several different mechanisms. Some token cards dynamically display one-time passwords that are synchronized with an authentication service. The server can verify the password provided by the token card at any given time by contacting the authentication service. Other token cards have a keypad and operate on a challenge-response basis. In this case, the server offers a challenge (a number) that the user enters into a token card. The token card provides a response (another number cryptographically derived from the challenge) that the user enters and sends to the server.

You can use SecurID tokens through the RADIUS adapter.

Related Topics

Configuring RADIUS Authentication
 RADIUS is a client/server security protocol widely used to enable remote authentication
 and access.

20.4.3 About Transport Layer Security

Transport Layer Security (TLS) is an industry standard protocol for securing network connections.

TLS provides authentication, data encryption, and data integrity.

The TLS protocol is the foundation of a public key infrastructure (PKI). For authentication, TLS uses digital certificates that comply with the X.509v3 standard and a public and private key pair.

With a public and a private key page, a set of two numbers are used for encryption and decryption, where one is called the private key and the other is called the public key. Public keys are typically made widely available, while private keys are held by their respective owners. Though mathematically related, it is generally viewed as computationally infeasible to derive the private key from the public key. Public and private keys are used only with asymmetric encryption algorithms, also called public-key encryption algorithms, or public-key cryptosystems. Data encrypted with either a public key or a private key from a key pair can be decrypted with its associated key from the key-pair. However, data encrypted with a public key cannot be decrypted with the same public key, and data enwrapped with a private key cannot be decrypted with the same private key.

Oracle Database TLS can be used to secure communications between any client and any server. You can configure TLS to provide authentication for the server only, the client only, or both client and server. You can also configure TLS features in combination with other authentication methods supported by Oracle Database (database user names and passwords, RADIUS, and Kerberos).

To support your PKI implementation, Oracle Database includes the following features in addition to TLS:

- Oracle wallets, where you can store PKI credentials
- Oracle Wallet Manager, which you can use to manage your Oracle wallets (Oracle Wallet Manager is deprecated with Oracle Database 21c. Instead of using Oracle Wallet Manager, Oracle recommends that you use the command line tools orapki and mkstore.)
- Certificate validation with certificate revocation lists (CRLs)
- Hardware security module support

Related Topics

- Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.
- Customizing the Use of Strong Authentication
 You can configure multiple authentication methods under Oracle Database native network
 encryption and strong authentication.



20.5 Oracle Database Native Network Encryption/Strong Authentication Architecture

The Oracle Database native network encryption and strong authentication architecture complements an Oracle database server or client installations.

The following diagram shows the this architecture within an Oracle networking environment.

Client Application OCI Two-Task Common Oracle Database Native Network Encryption and Strong Authentication Data Encryption Authentication Integrity Oracle Net AES Kerberos MD5 DES RADIUS SHA **RSA** DCE 3DES SSL Libraries SSL Adapter Oracle Protocols SPX/IPX Adapter TCP/IP Adapter Network Specific Protocols TCP/IP SPX/IPX To Network

Figure 20-3 Oracle Native Network Encryption and Strong Authentication Architecture

Oracle Database supports authentication through adapters that are similar to the existing Oracle protocol adapters. As shown in Figure 20-4, authentication adapters integrate the Oracle Net interface, and allow existing applications to take advantage of new authentication systems transparently, without any changes to the application.

Oracle Forms and Oracle Reports

Oracle Net

Oracle Database Native Network Encryption and Strong Authentication

Kerberos Adapter

Kerberos TLS

RADIUS

RADIUS

Figure 20-4 Oracle Net Services with Authentication Adapters

See Also:

Oracle Database Net Services Administrator's Guide for more information about stack communications in an Oracle networking environment

20.6 System Requirements for Strong Authentication

Kerberos, RADIUS, and Transport Layer Security (TLS) have a set of system requirements for strong authentication.

Table 20-1 lists the TLS system requirements for strong authentication.

Table 20-1 Authentication Methods and System Requirements

Authentication Method	System Requirements
Kerberos	 MIT Kerberos Version 5, release 1.8 or above. The Kerberos authentication server must be installed on a physically secure system.
RADIUS	 A RADIUS server that is compliant with the standards in the Internet Engineering Task Force (IETF) RFC #2138, Remote Authentication Dial In User Service (RADIUS) and RFC #2139 RADIUS Accounting.
	 To enable challenge-response authentication, you must run RADIUS on an operating system that supports the Java Native Interface as specified in release 1.1 of the Java Development Kit from JavaSoft.
TLS	 A wallet that is compatible with Oracle Database 10g.

20.7 Oracle Database Native Network Encryption and Strong Authentication Restrictions

Oracle applications support Oracle Database native network encryption and strong authentication.

However, because Oracle Database native network encryption and strong authentication requires Oracle Net Services to transmit data securely, these external authentication features are not supported by some parts of Oracle Financial, Human Resource, and Manufacturing Applications when they are running on Microsoft Windows.

The portions of these products that use Oracle Display Manager (ODM) do not take advantage of Oracle Database native network encryption and strong authentication, because ODM does not use Oracle Net Services.



Strong Authentication Administration Tools

You can use a set of strong authentication administration tools for native network encryption and public key infrastructure credentials.

- About the Configuration and Administration Tools
 The configuration and administration tools manage the encryption, integrity
 (checksumming), and strong authentication methods for Oracle Net Services.
- Native Network Encryption and Strong Authentication Configuration Tools
 Oracle Net Services can encrypt data using standard encryption algorithms, and for strong authentication methods, such as Kerberos, RADIUS, and SSL.
- Public Key Infrastructure Credentials Management Tools
 The security provided by a public key infrastructure (PKI) depends on how effectively you store, manage, and validate your PKI credentials.
- Duties of Strong Authentication Administrators
 Most of the tasks of a security administrator involve ensuring that the connections to and from Oracle databases are secure.

21.1 About the Configuration and Administration Tools

The configuration and administration tools manage the encryption, integrity (checksumming), and strong authentication methods for Oracle Net Services.

Strong authentication method configuration can include third-party software, as is the case for Kerberos or RADIUS, or it may entail configuring and managing a public key infrastructure for using digital certificates with Transport Layer Security (TLS).

21.2 Native Network Encryption and Strong Authentication Configuration Tools

Oracle Net Services can encrypt data using standard encryption algorithms, and for strong authentication methods, such as Kerberos, RADIUS, and SSL.

- About Oracle Net Manager
 Oracle Net Manager configures Oracle Net Services for an Oracle home on a local client or
 server host.
- Kerberos Adapter Command-Line Utilities
 The Kerberos adapter provides command-line utilities that obtain, cache, display, and remove Kerberos credentials.

21.2.1 About Oracle Net Manager

Oracle Net Manager configures Oracle Net Services for an Oracle home on a local client or server host.

Although you can use Oracle Net Manager, a graphical user interface tool, to configure Oracle Net Services, such as naming, listeners, and general network settings, it also enables you to configure the following features, which use the Oracle Net protocol:

- Strong authentication (Kerberos, RADIUS, and Transport Layer Security)
- Native network encryption (RC4, DES, 3DES, and AES)
- Checksumming for data integrity (MD5, SHA-1, SHA-2)



The DES, 3DES112, 3DES168, MD5, and RC4 algorithms are deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

21.2.2 Kerberos Adapter Command-Line Utilities

The Kerberos adapter provides command-line utilities that obtain, cache, display, and remove Kerberos credentials.

The following table briefly describes these utilities.

Table 21-1 Kerberos Adapter Command-Line Utilities

Utility Name	Description	
okinit	Obtains Kerberos tickets from the Key Distribution Center (KDC) and caches them in the user's credential cache	
oklist	Displays a list of Kerberos tickets in the specified credential cache	
okdstry	Removes Kerberos credentials from the specified credential cache	
okcreate	Automates the creation of keytabs from either the KDC or a service endpoint	



The Cybersafe adapter is not supported beginning with this release. You should use Oracle's Kerberos adapter in its place. Kerberos authentication with the Cybersafe KDC (Trust Broker) continues to be supported when using the Kerberos adapter.

Related Topics

Utilities for the Kerberos Authentication Adapter
 The Oracle Kerberos authentication adapter utilities are designed for an Oracle client with
 Oracle Kerberos authentication support installed.

21.3 Public Key Infrastructure Credentials Management Tools

The security provided by a public key infrastructure (PKI) depends on how effectively you store, manage, and validate your PKI credentials.

About Oracle Wallet Manager

Wallet owners and security administrators use Oracle Wallet Manager to manage and edit the security credentials in their Oracle wallets.

About the orapki Utility

The orapki utility manages certificate revocation lists (CRLs), creates and manages Oracle wallets, and creates signed certificates.

21.3.1 About Oracle Wallet Manager

Wallet owners and security administrators use Oracle Wallet Manager to manage and edit the security credentials in their Oracle wallets.

A wallet is a password-protected container that is used to store authentication and signing credentials, including private keys, certificates, and trusted certificates needed by Transport Layer Security. You can use Oracle Wallet Manager to perform the following tasks:

- Create public and private key pairs
- Store and manage user credentials
- Generate certificate requests
- Store and manage certificate authority certificates (root key certificate and certificate chain)
- Upload and download wallets to and from an LDAP directory
- Create wallets to store hardware security module credentials



In previous releases of Oracle Database, you could use Oracle Wallet Manager to configure wallets for Transparent Data Encryption. In this release, you can use the ADMINISTER KEY MANAGEMENT SQL statement instead.

Related Topics

Oracle Database Advanced Security Guide

21.3.2 About the orapki Utility

The orapki utility manages certificate revocation lists (CRLs), creates and manages Oracle wallets, and creates signed certificates.

The basic syntax for this command-line utility is as follows:

```
orapki module command -option 1 argument ... -option n argument
```

For example, the following command lists all certificate revocation lists (CRLs) in the CRL subtree in an instance of Oracle Internet Directory that is installed on machinel.us.example.com and that uses port 389:

```
orapki crl list -ldap machine1.us.example.com:389
```



Note:

The use of orapki to configure Transparent Data Encryption has been deprecated. Instead, use the ADMINISTER KEY MANAGEMENT SQL statement.

Related Topics

- Certificate Revocation List Management
 Certificate revocation list management entails ensuring that the CRLs are the correct format before you enable certificate revocation checking.
- Managing Public Key Infrastructure (PKI) Elements
 You can use the orapki command line utility and sqlnet.ora parameters to manage public key infrastructure (PKI) elements.

21.4 Duties of Strong Authentication Administrators

Most of the tasks of a security administrator involve ensuring that the connections to and from Oracle databases are secure.

The following table describes the primary tasks of security administrators who are responsible for strong authentication, the tools used to perform the tasks, and links to where the tasks are documented.

Table 21-2 Common Security Administrator/DBA Configuration and Administrative Tasks

Task	Tools Used	See Also
Configure encrypted Oracle Net connections between database servers and clients	Oracle Net Manager	Configuring Encryption on the Client and the Server
Configure checksumming on Oracle Net connections between database servers and clients	Oracle Net Manager	Configuring Integrity on the Client and the Server
Configure database clients to accept RADIUS authentication	Oracle Net Manager	Step 1A: Configure RADIUS on the Oracle Client
Configure a database to accept RADIUS authentication	Oracle Net Manager	Step 1B: Configure RADIUS on the Oracle Database Server
Create a RADIUS user and grant them access to a database session	SQL*Plus	Step 2: Create a User and Grant Access
Configure Kerberos authentication on a database client and server	Oracle Net Manager	Step 6: Configure Kerberos Authentication
Create a Kerberos database user	kadmin.localOracle Net Manager	 Step 7: Create a Kerberos User Step 8: Create an Externally Authenticated Oracle User
Manage Kerberos credentials in the credential cache	okinitoklistokdstryokcreate	 okinit Utility Options for Obtaining the Initial Ticket oklist Utility Options for Displaying Credentials okdstry Utility Options for Removing Credentials from the Cache File
Create a wallet for a database client or server	Oracle Wallet Manager	Oracle Database Enterprise User Security Administrator's Guide



Table 21-2 (Cont.) Common Security Administrator/DBA Configuration and Administrative Tasks

Task	Tools Used	See Also
Request a user certificate from a certificate authority (CA) for TLS authentication	Oracle Wallet Manager	Oracle Database Enterprise User Security Administrator's Guide to add a certificate request
		 Oracle Database Enterprise User Security Administrator's Guide to import a user certificate into an Oracle wallet
Import a user certificate and its associated trusted certificate (CA certificate) into a wallet	Oracle Wallet Manager	 Oracle Database Enterprise User Security Administrator's Guide to import a trusted certificate Oracle Database Enterprise User Security Administrator's Guide to import a user certificate into an Oracle wallet
Configuring TLS connections for a database client	Oracle Net Manager	Step 2: Configure Transport Layer Security on the Client
Configuring TLS connections for a database server	Oracle Net Manager	Step 1: Configure Transport Layer Security on the Server
Enabling certificate validation with a certificate revocation list (CRL)	Oracle Net Manager	Configuring Certificate Validation with Certificate Revocation Lists



Configuring Kerberos Authentication

Kerberos is a trusted third-party authentication system that relies on shared secrets and presumes that the third party is secure.

- Enabling Kerberos Authentication
 - To enable Kerberos authentication for Oracle Database, you must first install it, and then follow a set of configuration steps.
- Utilities for the Kerberos Authentication Adapter
 - The Oracle Kerberos authentication adapter utilities are designed for an Oracle client with Oracle Kerberos authentication support installed.
- Connecting to an Oracle Database Server Authenticated by Kerberos
 After Kerberos is configured, you can connect to an Oracle database server without using
 a user name or password.
- Configuring Interoperability with a Windows 2008 Domain Controller KDC
 You can configure Oracle Database to interoperate with a Microsoft Windows 2008 domain controller key distribution center (KDC).
- Configuring Kerberos Authentication Fallback Behavior
 You can configure fallback behavior (password-based authentication) in case the Kerberos authentication fails.
- Troubleshooting the Oracle Kerberos Authentication Configuration
 Oracle provides guidance for common Kerberos configuration problems.

22.1 Enabling Kerberos Authentication

To enable Kerberos authentication for Oracle Database, you must first install it, and then follow a set of configuration steps.

- Step 1: Install Kerberos
 - You should install Kerberos Version 5.
- Step 2: Configure a Service Principal for an Oracle Database Server
 - You must create a service principal for Oracle Database before the server can validate the identity of clients that authenticate themselves using Kerberos.
- Step 3: Extract a Service Key Table from Kerberos
 - Next, you are ready to extract the service key table from Kerberos and copy it to the Oracle database server/Kerberos client system.
- Step 4: Install an Oracle Database Server and an Oracle Client
 - After you extract a service key table from Kerberos, you are ready to install the Oracle Database server and an Oracle client.
- Step 5: Configure Oracle Net Services and Oracle Database
 - After you install the Oracle Database server and client, you can configure Oracle Net Services on the server and client.
- Step 6: Configure Kerberos Authentication
 - You must set the required parameters in the Oracle database server and client sqlnet.ora files.

Step 7: Create a Kerberos User

You must create the Kerberos user on the Kerberos authentication server where the administration tools are installed.

- Step 8: Create an Externally Authenticated Oracle User
 Next, you are ready to create an externally authenticated Oracle user.
- Step 9: Get an Initial Ticket for the Kerberos/Oracle User
 Before you can connect to the database, you must ask the Key Distribution Center (KDC)
 for an initial ticket.



Oracle Database Enterprise User Security Administrator's Guide for information on migrating Kerberos users to Kerberos-authenticated enterprise users

22.1.1 Step 1: Install Kerberos

You should install Kerberos Version 5.

The source distribution for notes about building and installing Kerberos provide details. After you install Kerberos, if you are using IBM AIX on POWER systems (64-bit), you should ensure that Kerboros 5 is the preferred authentication method.

1. Install Kerberos on the system that functions as the authentication server.

Note:

After upgrading from a 32-bit version of Oracle Database, the first use of the Kerberos authentication adapter causes an error message: ORA-01637: Packet receive failed.

Workaround: After upgrading to the 64-bit version of the database and before using Kerberos external authentication method, check for a file named /usr/tmp/oracle service name.RC on your computer, and remove it.

2. For IBM AIX on POWER systems (64-bit), check the authentication method.

For example:

/usr/bin/lsauthent

Output similar to the following may appear:

Standard Aix

3. Configure Kerberos 5 as the preferred method.

For example:

/usr/bin/chauthent -k5 -std

This command sets Kerberos 5 as the preferred authentication method (k5) and Standard AIX as the second (std).

4. To ensure that Kerberos 5 is now the preferred method, check the new configuration.

/usr/bin/lsauthent

Kerberos 5 Standard Aix

22.1.2 Step 2: Configure a Service Principal for an Oracle Database Server

You must create a service principal for Oracle Database before the server can validate the identity of clients that authenticate themselves using Kerberos.

Decide on a name for the service principal, using the following format:

kservice/kinstance@REALM

Each of the fields in the service principal specify the following values:

Service Principal Field	Description
kservice	A case-sensitive string that represents the Oracle service. This can be the same as the database service name.
kinstance	Typically the fully qualified DNS name of the system on which Oracle Database is running.
REALM	The name of the Kerberos realm with which the service principal is registered. REALM must always be uppercase and is typically the DNS domain name.

The utility names in this section are executable programs. However, the Kerberos user name krbuser and the realm EXAMPLE.COM are examples only.

For example, suppose kservice is oracle, the fully qualified name of the system on which Oracle Database is running is dbserver.example.com and the realm is EXAMPLE.COM. The principal name then is:

oracle/dbserver.example.com@EXAMPLE.COM

2. Run kadmin.local to create the service principal. On UNIX, run this command as the root user, by using the following syntax:

```
# cd /kerberos-install-directory/sbin
# ./kadmin.local
```

For example, to add a principal named oracle/dbserver.example.com@EXAMPLE.COM to the list of server principals known by Kerberos, you can enter the following:

kadmin.local:addprinc -randkey oracle/dbserver.example.com@EXAMPLE.COM

22.1.3 Step 3: Extract a Service Key Table from Kerberos

Next, you are ready to extract the service key table from Kerberos and copy it to the Oracle database server/Kerberos client system.

For example, to extract a service key table for dbserver.example.com:

- Ensure that you have domain administrative privileges.
- Enter the following to extract the service key table:

```
kadmin.local: ktadd -k /tmp/keytab oracle/dbserver.example.com
Entry for principal oracle/dbserver.example.com with kvno 2,
encryption type AES-256 CTS mode with 96-bit SHA-1 HMAC added to keytab WRFILE:
WRFILE:/tmp/keytab
```

```
kadmin.local: exit
```

3. To check the service key table, enter the following command:

```
oklist -k -t /tmp/keytab
```

4. After the service key table has been extracted, verify that the new entries are in the table in addition to the old ones.

If they are not, or you need to add more, use kadmin.local to append to them.

If you do not enter a realm when using ktadd, it uses the default realm of the Kerberos server. kadmin.local is connected to the Kerberos server running on the localhost.

5. If the Kerberos service key table is on the same system as the Kerberos client, you can move it. If the service key table is on a different system from the Kerberos client, you must transfer the file with a program such as FTP. If using FTP, transfer the file in binary mode.

The following example shows how to move the service key table on a UNIX platform:

```
# mv /tmp/keytab /etc/v5srvtab
```

The default name of the service file is /etc/v5srvtab.

6. Verify that the owner of the Oracle database server executable can read the service key table (/etc/v5srvtab in the previous example).

To do so, set the file owner to the Oracle user, or make the file readable by the group to which Oracle belongs.

Do not make the file readable to all users. This can cause a security breach.

22.1.4 Step 4: Install an Oracle Database Server and an Oracle Client

After you extract a service key table from Kerberos, you are ready to install the Oracle Database server and an Oracle client.

 See the Oracle Database operating system-specific installation documentation for instructions on installing the Oracle database server and client software.

22.1.5 Step 5: Configure Oracle Net Services and Oracle Database

After you install the Oracle Database server and client, you can configure Oracle Net Services on the server and client.

- See the following documentation for information on configuring Oracle Net Services on the Oracle database server and client.
 - Oracle Database operating system-specific installation documentation
 - Oracle Database Net Services Administrator's Guide

22.1.6 Step 6: Configure Kerberos Authentication

You must set the required parameters in the Oracle database server and client sqlnet.ora files.

Note:

Be aware that in a multitenant environment, the settings in the sqlnet.ora file apply to all pluggable databases (PDBs). However, this does not mean that all PDBs must authenticate with one KDC if using Kerberos; the settings in the sqlnet.ora file and Kerberos configuration files can support multiple KDCs.

- Step 6A: Configure Kerberos on the Client and on the Database Server First, you must configure Kerberos authentication service parameters on the client and on the database server.
- Step 6B: Set the Initialization Parameters

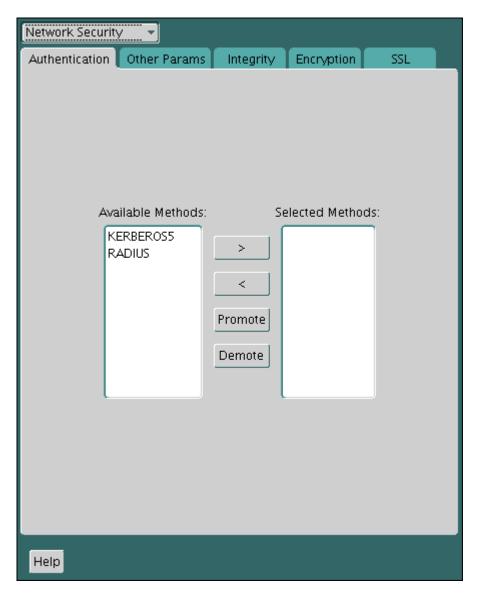
 Next, you are ready to set the OS AUTHENT PREFIX initialization parameter.
- Step 6C: Set sqlnet.ora Parameters (Optional)
 You can set optional sqlnet.ora parameters, in addition to the required parameters, for better security.

22.1.6.1 Step 6A: Configure Kerberos on the Client and on the Database Server

First, you must configure Kerberos authentication service parameters on the client and on the database server.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line: netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.The Network Security tabbed window appears.
- Select the Authentication tab.





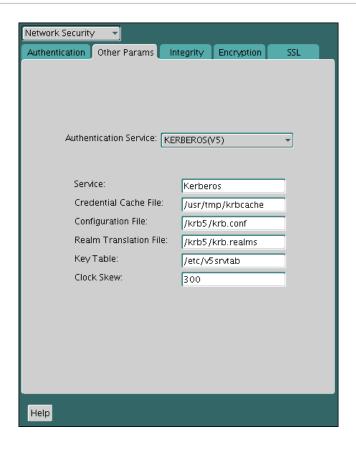
5. From the Available Methods list, select **KERBEROS5**.

Be aware that cross-realm Kerberos authentication is not supported using constraint delegation with the KERBEROS5 or KERKBEROS5PRE adapter.

- **6.** Move **KERBEROS5** to the Selected Methods list by clicking the right arrow (>).
- 7. Arrange the selected methods in order of use.

To do so, select a method in the Selected Methods list, then click **Promote** or **Demote** to position it in the list. For example, if you want KERBEROS5 to be the first service used, move it to the top of the list.

- 8. Select the Other Params tab.
- 9. From the Authentication Service list, select **KERBEROS(V5)**.
- 10. Type **Kerberos** into the **Service** field.



This field defines the name of the service Oracle Database uses to obtain a Kerberos service ticket. When you provide the value for this field, the other fields are enabled.

- 11. Optionally enter values for the following fields:
 - Credential Cache File
 - Configuration File
 - Realm Translation File
 - Key Table
 - Clock Skew

See the Oracle Net Manager online Help, and Step 6C: Set sqlnet.ora Parameters (Optional), for more information about the fields and the parameters they configure.

12. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entries in addition to any optional choices that you may have made in the previous step:

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=kservice
```

22.1.6.2 Step 6B: Set the Initialization Parameters

Next, you are ready to set the OS AUTHENT PREFIX initialization parameter.

1. Locate the init.ora file.

By default, the <code>init.ora</code> file is located in the <code>ORACLE_HOME/dbs</code> directory (or the same location of the data files) on Linux and UNIX systems, and in the <code>ORACLE_HOME\database</code> directory on Windows.

In the init.ora file, set the value of OS_AUTHENT_PREFIX to null in the init.ora initialization parameter file.

For example:

OS_AUTHENT_PREFIX=""

Set this value to null because Kerberos user names can be long, and Oracle user names are limited to 30 bytes. Setting this parameter to null overrides the default value of OPS\$.

Note:

You can create external database users that have Kerberos user names of more than 30 bytes. See Step 8: Create an Externally Authenticated Oracle User for more information.

22.1.6.3 Step 6C: Set sqlnet.ora Parameters (Optional)

You can set optional sqlnet.ora parameters, in addition to the required parameters, for better security.

 Optionally, set the parameters listed in the following table on both the client and the Oracle database server.



Table 22-1 Kerberos-Specific sqlnet.ora Parameters

Parameter

Description

SQLNET.KERBEROS5_CC_NAME=pathname_t o_credentials_cache_file|OS_MEMORY

Specifies the complete path name to the Kerberos credentials cache (CC) file. The default value is operating system-dependent. For UNIX, it is /tmp/krb5cc userid.

Using the OS_MEMORY option indicates that an OS-managed memory credential cache is used for the credential cache file. This option is supported in all platforms.

You can use the following formats to specify a value for SQLNET. KERBEROS5 CC NAME:

SQLNET.KERBEROS5_CC_NAME=complete_path_to_cc_file

For example:

SQLNET.KERBEROS5_CC_NAME=/tmp/kcache SQLNET.KERBEROS5 CC NAME=D:\tmp\kcache

• SQLNET.KERBEROS5_CC_NAME=FILE:complete_path_to_cc_file

For example:

SQLNET.KERBEROS5 CC NAME=FILE:/tmp/kcache

• SQLNET.KERBEROS5 CC NAME=OSMSFT://

Use this value if you are running Windows and using a Microsoft KDC. You can also set this parameter by using the KRB5CCNAME environment variable, but the value set in the sqlnet.ora file takes precedence over the value set in KRB5CCNAME.

For example:

SQLNET.KERBEROS5 CC NAME=/usr/tmp/krbcache

SQLNET.KERBEROS5_CLOCKSKEW=number_o f seconds accepted as network delay

This parameter specifies how many seconds can pass before a Kerberos credential is considered out-of-date. It is used when a credential is actually received by either a client or a database server. An Oracle database server also uses it to decide if a credential needs to be stored to protect against a replay attack. The default is 300 seconds.

For example:

SQLNET.KERBEROS5 CLOCKSKEW=1200

SQLNET.KERBEROS5_CONF=pathname_to_K
erberos_configuration_file|
AUTO_DISCOVER

This parameter specifies the complete path name to the Kerberos configuration file. The configuration file contains the realm for the default KDC (key distribution center) and maps realms to KDC hosts. The default is operating system-dependent. For UNIX, it is /krb5/krb.conf.

Using the AUTO_DISCOVER option in place of the configuration file enables Kerberos clients to auto-discover the KDC.

For example:

SQLNET.KERBEROS5_CONF=/krb/krb.conf SQLNET.KERBEROS5_CONF=AUTO_DISCOVER

SQLNET.KERBEROS5_CONF_LOCATION=path
 to_Kerberos_configuration_director
v

This parameter indicates that the Kerberos configuration file is created by the system, and does not need to be specified by the client. The configuration file uses DNS lookup to obtain the realm for the default KDC, and maps realms to KDC hosts.

For example:

SQLNET.KERBEROS5 CONF LOCATION=/krb

Table 22-1 (Cont.) Kerberos-Specific sqlnet.ora Parameters

Parameter	Description
SQLNET.KERBEROS5_KEYTAB=path_to_Ker beros_principal/key_table	This parameter specifies the complete path name to the Kerberos principal/secret key mapping file. It is used by the Oracle database server to extract its key and decrypt the incoming authentication information from the client. The default is operating system-dependent. For UNIX, it is /etc/v5srvtab.
	For example:
	SQLNET.KERBEROS5_KEYTAB=/etc/v5srvtab
SQLNET.KERBEROS5_REALMS=path_to_Kerberos_realm_translation_file	This parameter specifies the complete path name to the Kerberos realm translation file. The translation file provides a mapping from a host name or domain name to a realm. The default is operating system-dependent. For UNIX, it is /etc/krb.realms.
	For example:
	SQLNET.KERBEROS5_REALMS=/krb5/krb.realms

22.1.7 Step 7: Create a Kerberos User

You must create the Kerberos user on the Kerberos authentication server where the administration tools are installed.

The realm must already exist.



The utility names in this section are executable programs. However, the Kerberos user name krbuser and realm EXAMPLE.COM are examples only. They can vary among systems.

Run /krb5/admin/kadmin.local as root to create a new Kerberos user, such as krbuser.

For example, to create a Kerberos user is UNIX-specific:

```
# /krb5/admin/kadmin.local
kadmin.local: addprinc krbuser
Enter password for principal: "krbuser@example.com": (password does not display)
Re-enter password for principal: "krbuser@example.com": (password does not display)
kadmin.local: exit
```

22.1.8 Step 8: Create an Externally Authenticated Oracle User

Next, you are ready to create an externally authenticated Oracle user.

1. Log in to SQL*Plus as a user who has the CREATE USER privilege.

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

2. Ensure that the OS AUTHENT PREFIX is set to null ("").

Create an Oracle Database user account that corresponds to the Kerberos user. Enter the Oracle user name in uppercase and enclose it in double quotation marks.

For example:

```
CREATE USER krbuser IDENTIFIED EXTERNALLY AS 'krbuser@example.com'; GRANT CREATE SESSION TO krbuser;
```



The database administrator should ensure that two database users are not identified externally by the same Kerberos principal name.

22.1.9 Step 9: Get an Initial Ticket for the Kerberos/Oracle User

Before you can connect to the database, you must ask the Key Distribution Center (KDC) for an initial ticket.

To request an initial ticket, run the following command on the client:

```
% okinit username
```

If you want to enable credentials that can be used across database links, then include the -f option and provide the Kerberos password when prompted.

```
% services/okinit -f
Password for krbuser@EXAMPLE.COM:(password does not display)
```

Note:

The following check is only required when using the KERBEROS5PRE adapter. It is not required for the KERBEROS5 adapter.

The use of the KERBEROS5PRE adapter is deprecated with Oracle Database 21c. Oracle recommends that you use the KERBEROS5 adapter instead.

If you encounter an error such as <code>okinit:</code> Cannot contact any KDC for requested realm, then check the <code>/etc/services</code> file if there are the kerberos5 entries. For example:

```
kerberos88/tcpkerberos5 krb5# Kerberos v5kerberos88/udpkerberos5 krb5# Kerberos v5
```

22.2 Utilities for the Kerberos Authentication Adapter

The Oracle Kerberos authentication adapter utilities are designed for an Oracle client with Oracle Kerberos authentication support installed.

- okinit Utility Options for Obtaining the Initial Ticket
 The okinit utility obtains and caches Kerberos tickets.
- oklist Utility Options for Displaying Credentials
 The oklist utility displays the list of tickets held.

- okdstry Utility Options for Removing Credentials from the Cache File
 The okdstry (okdestroy) utility removes credentials from the cache file.
- okcreate Utility Options for Automatic Keytab Creation
 The okcreate utility automates the creation of keytabs from either the KDC or a service endpoint.

22.2.1 okinit Utility Options for Obtaining the Initial Ticket

The okinit utility obtains and caches Kerberos tickets.

This utility is typically used to obtain the ticket-granting ticket, using a password entered by the user to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in the user's credential cache.

The following table lists the options available with <code>okinit</code>. To use the functionality that is described in this table, you must set the <code>sqlnet.oraSQLNET.KERBEROS5_CONF_MIT</code> parameter to <code>TRUE</code>. (Note that <code>SQLNET.KERBEROS5_CONF_MIT</code> is deprecated, but is retained for backward compatibility for <code>okinit</code>.)

Table 22-2 Options for the okinit Utility

Option	Description
-f -F	Requests forwardable or non-forwardable tickets. This option is necessary to follow database links.
-l lifetime	Specifies the lifetime of the ticket-granting ticket and all subsequent tickets. By default, the ticket-granting ticket is good for eight (8) hours, but shorter or longer-lived credentials may be desired. The KDC can ignore this option or put site-configured limits on what can be specified. The lifetime value is a string that consists of a number qualified by ${\tt w}$ (weeks), ${\tt d}$ (days), ${\tt h}$ (hours), ${\tt m}$ (minutes), or s (seconds), as in the following example:
	okinit -1 2wld6h20m30s
	The example requests a ticket-granting ticket that has a lifetime of 2 weeks, 1 day, 6 hours, 20 minutes, and 30 seconds.
-s start_time	Specifies the duration of the delay before the ticket can become valid. Tickets are issued with the invalid flag set.
-r renewable_life	Requests renewable tickets with a total lifetime of renewable_life
-p -P	Requests proxiable or non-proxiable tickets
-a	Requests tickets that are restricted to the local address of the host
-A	Requests tickets not restricted by address
-E	Treats the principal name as an enterprise name
-v	Requests that the ticket-granting ticket in the cache be passed to the KDC for validation. If the ticket is within the requested time range, then the cache is replaced with the validated ticket.
-R	Requests renewal of the ticket-granting ticket
-k[-t keytab_file]	Requests a ticket, which is obtained from a key in the local host's keytab
-n	Requests anonymous processing
-C	Requests canonicalization of the principal name, and enables the KDC to reply with a different client principal from the one that was requested



Table 22-2 (Cont.) Options for the okinit Utility

Option	Description					
-c cache_name	Specifies the name of a cache as a cache location. For UNIX, the default is /tmp/krb5cc_uid. You can also specify the alternate credential cache by using the SQLNET.KERBEROS5_CC_NAME parameter in the sqlnet.ora file.					
-I input_cache	Specifies the name of a credential cache that already contains a ticket. When obtains that ticket, if the information about how the ticket was obtained is store in cache, then the same information will be used to affect how new credentials are obtained.					
-T armor_cache	If supported by the KDC, this cache is used to armor the request, preventing offline dictionary attacks and enabling the use of additional pre-authentication mechanisms.					
-X attribute[=value	 Specifies a pre-authentication attribute and value. Specifies one of the following values: X509_user_identity=value specifies where to find the user's X509 identity information X509_anchors=value specifies where to find trusted X509 anchor information flag_RSA_PROTOCOL[=yes] specifies the use of RSA rather than the default Diffie-Hellman protocol 					
-?	List command line options.					

22.2.2 oklist Utility Options for Displaying Credentials

The oklist utility displays the list of tickets held.

The following table lists the available <code>oklist</code> options. To use the functionality that is described in this table, you must set the <code>sqlnet.oraSQLNET.KERBEROS5_CONF_MIT</code> parameter to <code>TRUE</code>. (Note that <code>SQLNET.KERBEROS5_CONF_MIT</code> is deprecated, but is retained for backward compatibility for <code>oklist</code>.)

Table 22-3 Options for the oklist Utility

Option	Description
-f	Show flags with credentials. Relevant flags are:
	 I, credential is a ticket-granting ticket
	F, credential is forwardable
	f, credential is forwarded.
-c	Specify an alternative credential cache. In UNIX, the default is /tmp/krb5cc_uid. The alternate credential cache can also be specified by using the SQLNET.KERBEROS5_CC_NAME parameter in the sqlnet.ora file.
-k	List the entries in the service table (default /etc/v5srvtab) on UNIX. The alternate service table can also be specified by using the SQLNET.KERBEROS5_KEYTAB parameter in the sqlnet.ora file.
- e	Displays the encryption types of the session key and the ticket for each credential in the credential cache, or each key in the keytab file.
-1	If a cache collection is available, displays a table summarizing the caches present in the collection.



Table 22-3 (Cont.) Options for the oklist Utility

Option	Description
-A	If a cache collection is available, displays the contents of all of the caches in the collection
-s	Runs utility without producing output. Utility will exit with status 1 if the cache cannot be read or is expired, else with status 0
-a	Displays a list of addresses in the credential
-n	Shows numeric addresses instead of reverse-resolving addresses
-C	Lists configuration data that has been stored in the credentials cache when klist encounters it. By default, configuration data is not listed.
-t	Displays the time entry timestamps for each keytab entry in the keytab file
-K	Displays the value of the encryption key in each keytab entry in the keytab file
-V	Displays the Kerberos version number and exit.

The show flag option (-f) displays additional information, as shown in the following example:

% oklist -f 04-Aug-2015 21:57:51 28-Aug-2015 05:58:14 krbtgt/EXAMPLE.COM@EXAMPLE.COM Flags: FI

22.2.3 okdstry Utility Options for Removing Credentials from the Cache File

The okdstry (okdestroy) utility removes credentials from the cache file.

The following table lists the available <code>okdstry</code> options. To use the functionality that is described in this table, you must set the <code>sqlnet.oraSQLNET.KERBEROS5_CONF_MIT</code> parameter to <code>TRUE</code>. (Note that <code>SQLNET.KERBEROS5_CONF_MIT</code> is deprecated, but is retained for backward compatibility for <code>okdstry</code>.)

Table 22-4 Options for the okdstry Utility

Option	Description
— A	Destroys all caches in the collection, if a cache collection is available
-q	Runs quietly. Normally okdstry beeps if it fails to destroy the user's tickets. This flag suppresses this behavior.
-c cache_name	Uses $cache_name$ as the credentials (ticket) cache name and location. For UNIX, the default is $/tmp/krb5cc_uid$. You can also specify the alternate credential cache by using the SQLNET.KERBEROS5_CC_NAME parameter in the sqlnet.ora file.

22.2.4 okcreate Utility Options for Automatic Keytab Creation

The okcreate utility automates the creation of keytabs from either the KDC or a service endpoint.

The following table lists the available okcreate options.

Table 22-5 okcreate Utility Options for Automatic Keytab Creation

Option	Description
-name service_name	Specifies the service name of the kerberized service for which to get a keytab. The default is oracle.
-hosts path- to_hosts_list	Specifies either a comma-separated list of hosts for which to get the keytab, or the path to a text file that contains a list of the hosts. The default is none.
-out path_to_output	Specifies the output path to store the resulting keytabs. The default is the current directory.
	Ensure that this directory is readable only by the root user. Never send keytabs over the network in clear text.
-k	For use if the operation is performed on the KDC. Do not use this option if you are using $-s$.
-s	For use if the operation is performed on a Kerberized service. Do not use this option if you are using $-\mathbf{k}$.
-u KDC_username	Specifies the user name for the KDC. Only use this setting on a Kerberized service endpoint.
	If you specify the $-s$ and omit this setting, then okcreate prompts for the ${\it KDCuser@KDCmachine}.$
-r	Specifies the Kerberos realm
-p	Specifies the Kerberos principal
-q	Specifies the Kerberos query
-d	Specifies the KDC database name
-e	Specifies the salt list to be used for any new keys that are created
—m	Specifies to prompt for the KDC master password

22.3 Connecting to an Oracle Database Server Authenticated by Kerberos

After Kerberos is configured, you can connect to an Oracle database server without using a user name or password.

 Use the following syntax to connect to the database without using a user name or password:

\$ sqlplus /@net_service_name

In this specification, net service name is an Oracle Net Services service name. For example:

\$ sqlplus /@oracle dbname



Oracle Database Heterogeneous Connectivity User's Guide for information about external authentication



22.4 Configuring Interoperability with a Windows 2008 Domain Controller KDC

You can configure Oracle Database to interoperate with a Microsoft Windows 2008 domain controller key distribution center (KDC).

- About Configuring Interoperability with a Microsoft Windows Server Domain Controller KDC
 - Oracle Database complies with MIT Kerberos.
- Step 1: Configure Oracle Kerberos Client for Windows 2008 Domain Controller You can configure the Oracle Kerberos client to interoperate with a Microsoft Windows 2008 Domain Controller KDC.
- Step 2: Configure a Microsoft Windows Server Domain Controller KDC for the Oracle Client
 - Next, you are ready to configure a Microsoft Windows Server Domain Controller KDC to interoperate with an Oracle Client.
- Step 3: Configure Oracle Database for a Microsoft Windows Server Domain Controller KDC
 - You must configure the Oracle database for the domain controller on the host computer where the Oracle database is installed.
- Step 4: Obtain an Initial Ticket for the Kerberos/Oracle User
 Before a client can connect to the database, the client must request an initial ticket.

22.4.1 About Configuring Interoperability with a Microsoft Windows Server Domain Controller KDC

Oracle Database complies with MIT Kerberos.

This enables Oracle Database to interoperate with tickets that are issued by a Kerberos Key Distribution Center (KDC) on a Microsoft Windows Server domain controller. This process enables Kerberos authentication with an Oracle database.

22.4.2 Step 1: Configure Oracle Kerberos Client for Windows 2008 Domain Controller

You can configure the Oracle Kerberos client to interoperate with a Microsoft Windows 2008 Domain Controller KDC.

- Step 1A: Create the Client Kerberos Configuration Files
 You must configure a set of client Kerberos configuration files that refer to the Windows
 2008 domain controller as the Kerberos KDC.
- Step 1B: Specify the Oracle Configuration Parameters in the sqlnet.ora File
 Configuring an Oracle client to interoperate with a Microsoft Windows Server Domain
 Controller Kerberos Key Distribution Center (KDC) uses the same sqlnet.ora file
 parameters that are used for configuring Kerberos on the client and on the database
 server.
- Step 1C: Optionally, Specify Additional Kerberos Principals Using tnsnames.ora
 You can configure additional Kerberos principal users to connect from an Oracle Database client.



Step 1D: Specify the Listening Port Number
 The Microsoft Windows Server domain controller KDC listens on UDP/TCP port 88.

22.4.2.1 Step 1A: Create the Client Kerberos Configuration Files

You must configure a set of client Kerberos configuration files that refer to the Windows 2008 domain controller as the Kerberos KDC.

Create the krb.conf and krb5.realms files. Oracle Database provides a default krb5.conf file, which you must modify for your site.

The krb5.conf file is located in the location indicated by the SQLNET.KERBEROS_CONF parameter.

For example, assuming that the Windows 2008 domain controller is running on a node named sales3854.us.example.com:

krb.conf file

For example:

```
SALES3854.US.EXAMPLE.COM
SALES3854.US.EXAMPLE.COM
sales3854.us.example.com admin server
```

krb5.conf file

For example:

```
[libdefaults]
default_realm=SALES.US.EXAMPLE.COM
[realms]
SALES.US.EXAMPLE.COM= { kdc=sales3854.us.example.com:88 }
[domain_realm]
.us.example.com=SALES.US.EXAMPLE.COM
```

krb5.realms file

For example:

us.example.com SALES.US.EXAMPLE.COM

22.4.2.2 Step 1B: Specify the Oracle Configuration Parameters in the sqlnet.ora File

Configuring an Oracle client to interoperate with a Microsoft Windows Server Domain Controller Kerberos Key Distribution Center (KDC) uses the same sqlnet.ora file parameters that are used for configuring Kerberos on the client and on the database server.

Set the following parameters in the sqlnet.ora file on the client:

```
SQLNET.KERBEROS5_CONF=pathname_to_Kerberos_configuration_file
SQLNET.KERBEROS5_CONF_MIT=TRUE
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=Kerberos_service_name
SQLNET.AUTHENTICATION_SERVICES=(BEQ, KERBEROS5)
```

Note the following:

- The SQLNET.KERBEROS5_CONF_MIT parameter has been deprecated, but is retained for backward compatibility for the okint, oklist, and okdstry utilities.
- Ensure that the SQLNET.KERBEROS5_CONF_MIT parameter is set to TRUE because the Windows Server operating system is designed to interoperate only with security services that are based on MIT Kerberos version 5.

 If you want to use multiple Kerberos principal users, then you can specify them as part of a connect string or in tnsnames.ora.

Related Topics

- Step 6A: Configure Kerberos on the Client and on the Database Server
 First, you must configure Kerberos authentication service parameters on the client and on
 the database server.
- Step 1C: Optionally, Specify Additional Kerberos Principals Using the things of the second sec

22.4.2.3 Step 1C: Optionally, Specify Additional Kerberos Principals Using tnsnames.ora

You can configure additional Kerberos principal users to connect from an Oracle Database client.

• Add the KERBEROS5_CC_NAME and KERBEROS5_PRINCIPAL settings to the tnsnames.ora connect string.

KERBEROS5_CC_NAME is mandatory for all additional Kerberos users and principals, but the KERBEROS5_PRINCIPAL setting is optional. Oracle Database checks KERBEROS5_PRINCIPAL against the value that is retrieved from the credential cache. If the two values do not match, then the user is not authenticated.

For example:

Related Topics

Oracle Database Net Services Reference

22.4.2.4 Step 1D: Specify the Listening Port Number

The Microsoft Windows Server domain controller KDC listens on UDP/TCP port 88.

1. Ensure that the system file entry for kerberos5 is set to UDP/TCP port 88.

2. Note:

This step is only required when using the KERBEROS5PRE adapter. This step can be skipped when using the KERBEROS5 adapter.

The use of the $\tt KERBEROS5PRE$ adapter is deprecated with Oracle Database 21c. Oracle recommends that you use the $\tt KERBEROS5$ adapter instead.



For the UNIX environment, ensure that the first kerberos5 entry in the /etc/services file is set to 88.

22.4.3 Step 2: Configure a Microsoft Windows Server Domain Controller KDC for the Oracle Client

Next, you are ready to configure a Microsoft Windows Server Domain Controller KDC to interoperate with an Oracle Client.

- Step 2A: Create the User Account
 You must create a user account for the Microsoft Windows Server Domain Controller KDC.
- Step 2B: Create the Oracle Database Principal User Account and Keytab
 After you create the user account, you are ready to create the Oracle Database principal
 user account.



Microsoft documentation for information about how to create users in Active Directory.

22.4.3.1 Step 2A: Create the User Account

You must create a user account for the Microsoft Windows Server Domain Controller KDC.

 On the Microsoft Windows Server domain controller, create a new user account for the Oracle client in Microsoft Active Directory.

22.4.3.2 Step 2B: Create the Oracle Database Principal User Account and Keytab

After you create the user account, you are ready to create the Oracle Database principal user account.

After you create this account on the Windows Server domain controller, you must use the <code>okcreate</code> utility to register it with the principal keytab. You can run this utilty on the same KDC to create all the service keytabs rather than creating them individually, or you can run <code>okcreate</code> from a service endpoint that connects to the KDC, run the ncessary commands, and then copy the resulting keytab back to the service endpoint.

- 1. Create a new user account for the Oracle database in Microsoft Active Directory.
 - For example, if the Oracle database runs on the host sales3854.us.example.com, then use Active Directory to create a user with the user name sales3854.us.example.com.
 - Do not create a user as host/hostname.dns.com, such as oracle/sales3854.us.example.com, in Active Directory. Microsoft's KDC does not support multipart names like an MIT KDC does. An MIT KDC allows multipart names to be used for service principals because it treats all principals as user names. However, Microsoft's KDC does not.
- 2. Run the okcreate command to create a keytab that will use this user account. The syntax is as follows:

```
okcreate (-s [-u KDCuser@KDCmachine] | -k)
[-name service_name] [-hosts path_to_host_list]
```

```
[-out path_to_output] [-r realm] [-p principal]
[-q query] [-d dbname] [-e enc:salt...] [-m]
[-x db args]
```

For example:

```
okcreate -s -u kdcuser1@kdcmachine1 -name oracle
  -hosts sales3854.us.example.com
  -out /OSsecured/keytablocation
```

Copy the extracted keytab file to the host computer where the Oracle database is installed.

For example, the keytab that was created in the previous step can be copied to /krb5/v5svrtab.

22.4.4 Step 3: Configure Oracle Database for a Microsoft Windows Server Domain Controller KDC

You must configure the Oracle database for the domain controller on the host computer where the Oracle database is installed.

- Step 3A: Set Configuration Parameters in the sqlnet.ora File You must first set configuration parameters for the database.
- Step 3B: Create an Externally Authenticated Oracle User
 After you set the configuration parameters, you are ready to create an externally authenticated Oracle user.

22.4.4.1 Step 3A: Set Configuration Parameters in the sqlnet.ora File

You must first set configuration parameters for the database.

Specify values for the following parameters in the sqlnet.ora file for the database server:

```
SQLNET.KERBEROS5_CONF=pathname_to_Kerberos_configuration_file
SQLNET.KERBEROS5_KEYTAB=pathname_to_Kerberos_principal/key_table
SQLNET.KERBEROS5_CONF_MIT=TRUE
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=Kerberos_service_name
SQLNET.AUTHENTICATION_SERVICES=(BEQ,KERBEROS5)
```

Note:

- The SQLNET.KERBEROS5_CONF_MIT parameter has been deprecated, but is retained for backward compatibility for the okint, oklist, and okdstry utilities.
- Ensure that the SQLNET.KERBEROS5_CONF_MIT parameter is set to TRUE because the Windows Server operating system is designed to interoperate only with security services that are based on MIT Kerberos version 5.
- Be aware that the settings in the sqlnet.ora file apply to all PDBs. However, this
 does not mean that all PDBs must authenticate with one KDC if using Kerberos;
 the settings in the sqlnet.ora file and Kerberos configuration files can support
 multiple KDCs.



22.4.4.2 Step 3B: Create an Externally Authenticated Oracle User

After you set the configuration parameters, you are ready to create an externally authenticated Oracle user.

• Follow the procedure under Step 8: Create an Externally Authenticated Oracle User to create an externally authenticated Oracle user.

Ensure that you create the username in all uppercase characters (for example, ORAKRB@SALES.US.EXAMPLE.COM).



Step 6: Configure Kerberos Authentication for information about using Oracle Net Manager to set the sqlnet.ora file parameters.

22.4.5 Step 4: Obtain an Initial Ticket for the Kerberos/Oracle User

Before a client can connect to the database, the client must request an initial ticket.

 To request an initial ticket, follow the task information for Step 9: Get an Initial Ticket for the Kerberos/Oracle User.

The user does not need to explicitly request for an initial ticket, using the <code>okinit</code> command, when using the Windows native cache.

If the Oracle client is running on Microsoft Windows Server or later, then the Kerberos ticket is automatically retrieved when the user logs in to Windows.

See also the Microsoft documentation for details about the Kerbtray.exe utility, which can be used to display Kerberos ticket information for a system.

2. For each Kerberos principal user that you have added to the snames.ora, run the okinit command in the client.

For example:

okinit krbprinc1@example.com

22.5 Configuring Kerberos Authentication Fallback Behavior

You can configure fallback behavior (password-based authentication) in case the Kerberos authentication fails.

After you have configured Kerberos authentication for Oracle clients to use Kerberos authentication to authenticate to an Oracle database, there are cases where you may want to fall back to password-based authentication. An example would be if you have fixed user database links in the Oracle database.

 To enable Kerberos authentication to fall back to password-based authentication, set the SQLNET.FALLBACK_AUTHENTICATION parameter to TRUE in the sqlnet.ora files on both the client and server.

The default of this parameter is FALSE. This means that by default, the connection fails when Kerberos authentication fails.



Oracle Database Net Services Reference for more information about the SQLNET.FALLBACK AUTHENTICATION parameter

22.6 Troubleshooting the Oracle Kerberos Authentication Configuration

Oracle provides guidance for common Kerberos configuration problems.

Common problems are as follows:

- If you cannot get your ticket-granting ticket using okinit:
 - Ensure that the default realm is correct by examining the krb.conf file.
 - Ensure that the KDC is running on the host specified for the realm.
 - Ensure that the KDC has an entry for the user principal and that the passwords match.
 - Ensure that the krb.conf and krb.realms files are readable by Oracle.
 - Ensure that the TNS_ADMIN environment variable is pointing to the directory containing the sqlnet.ora configuration file.
- If you have an initial ticket but still cannot connect:
 - After trying to connect, check for a service ticket.
 - Check that the sqlnet.ora file on the database server side has a service name that corresponds to a service known by Kerberos.
 - Check that the clocks on all systems involved are set to times that are within a few minutes of each other or change the SQLNET.KERBEROS5_CLOCKSKEW parameter in the sqlnet.ora file.
- If you have a service ticket and you still cannot connect:
 - Check the clocks on the client and database server.
 - Check that the v5srvtab file exists in the correct location and is readable by Oracle.
 Remember to set the sqlnet.ora parameters.
 - Check that the v5srvtab file has been generated for the service named in the sqlnet.ora file on the database server side.
- If everything seems to work fine, but then you issue another query and it fails:
 - Check that the initial ticket is forwardable. You must have obtained the initial ticket by running the okinit utility.
 - Check the expiration date on the credentials. If the credentials have expired, then
 close the connection and run okinit to get a new initial ticket.
- Common Kerberos Configuration Problems
 Oracle provides guidance for common Kerberos configuration problems.
- ORA-12631 Errors in the Kerberos Configuration
 The ORA-12631: username retrieval failed error can result from the wrong or incorrectly formatted principal being used for the Kerberos authentication



ORA-28575 Errors in the Kerberos Configuration

The ORA-28575: unable to open RPC connection to external procedure agent error can occur when the client is remote and the EXTPROC process is spawned.

ORA-01017 Errors in the Kerberos Configuration

The ORA-01017: invalid username/password; logon denied error can result if okinit fails and there is no valid ticket in the SQL*Plus connection.

Enabling Tracing for Kerberos okinit Operations

The KRB5 TRACE environment variable enables you to trace Kerberos okinit operations.

22.6.1 Common Kerberos Configuration Problems

Oracle provides guidance for common Kerberos configuration problems.

Common problems are as follows:

- If you cannot get your ticket-granting ticket using okinit:
 - Ensure that the default realm is correct by examining the krb.conf file.
 - Ensure that the KDC is running on the host specified for the realm.
 - Ensure that the KDC has an entry for the user principal and that the passwords match.
 - Ensure that the krb.conf and krb.realms files are readable by Oracle.
 - Ensure that the TNS_ADMIN environment variable is pointing to the directory containing the sqlnet.ora configuration file.
- If you have an initial ticket but still cannot connect, try the following:
 - After trying to connect, check for a service ticket.
 - Check that the sqlnet.ora file on the database server side has a service name that corresponds to a service known by Kerberos.
 - Check that the clocks on all systems involved are set to times that are within a few minutes of each other or change the SQLNET.KERBEROS5_CLOCKSKEW parameter in the sqlnet.ora file.
- If you have a service ticket and you still cannot connect:
 - Check the clocks on the client and database server.
 - Check that the v5srvtab file exists in the correct location and is readable by Oracle.
 Remember to set the sqlnet.ora parameters.
 - Check that the v5srvtab file has been generated for the service named in the sqlnet.ora file on the database server side.
- If everything seems to work well, but then you issue another query and it fails, then try the following:
 - Check that the initial ticket is forwardable. You must have obtained the initial ticket by running the okinit utility.
 - Check the expiration date on the credentials. If the credentials have expired, then
 close the connection and run okinit to get a new initial ticket.

22.6.2 ORA-12631 Errors in the Kerberos Configuration

The ORA-12631: username retrieval failed error can result from the wrong or incorrectly formatted principal being used for the Kerberos authentication



Check the sqlnet server trace files for Wrong principal in request in the output.

To remedy this problem, edit the krb5.conf file and check the [domain_realm] settings. These settings are case sensitive, so even if the domain_realm name is correct, it will fail to parse correctly if it is lower case. Ensure that this setting is upper case. For example:

```
[domain_realm]
.country.<DOMAIN_NAME> = SECWIN.LOCAL
country.<DOMAIN NAME> = SECWIN.LOCAL
```

22.6.3 ORA-28575 Errors in the Kerberos Configuration

The ORA-28575: unable to open RPC connection to external procedure agent error can occur when the client is remote and the EXTPROC process is spawned.

There is no need to have Kerberos authentication with an external procedure call. To remedy this problem, add BEQ in front of the KERBEROS5 and KERBEROS5PRE parameters in the sqlnet.ora file.

22.6.4 ORA-01017 Errors in the Kerberos Configuration

The ORA-01017: invalid username/password; logon denied error can result if okinit fails and there is no valid ticket in the SQL*Plus connection.

The okinit trace file will show the following errors:

```
nauk51_sendto_kdc: entry
snauk51_sendto_kdc: exit
snauk51_sendto_kdc: exit
nauk51a_get_in_tkt: Returning 25: Additional pre-authentication required
.
snauk51_sendto_kdc: exit
snauk51_sendto_kdc: exit
nauk51a_get_in_tkt: Returning 24: Preauthentication failed
.
nauk51a_get_in_tkt: exit
nauk5zi_kinit: Getting TGT failed: Preauthentication failed
.
nauk5fq_free_principal: entry
nauk5fq_free_principal: exit
nauk5fq_free_principal: entry
nauk5fq_free_principal: exit
nauk5fq_free_principal: exit
nauk5fq_free_principal: exit
nauk5zi_kinit: Returning 24: Preauthentication failed
.
nauk5zi_kinit: exit
```

To remedy this problem:

1. Set the default_tkt_enctypes parameter in the krb5.conf file. This enables you to control the encryption types that are requested from the client. For example:

```
default_tgs_enctypes = aes256-cts-hmac-sha1-96
default tkt enctypes = aes256-cts-hmac-sha1-96
```

Test okinit with the following option:

```
okinit user name
```

If DES encryption algorithm is not implemented in an Active Directory server, the okinit fails:

```
okinit user_name

Kerberos Utilities for Solaris: Version 23.0.0.0.0 - Production on 15-MAY-2023 11:50:39

Copyright (c) 1996, 2023 Oracle. All rights reserved.

Password for user_name@domain:
okinit: KDC has no support for encryption type

okinit user_name

Kerberos Utilities for Solaris: Version 23.0.0.0.0 - Production on 15-MAY-2023 11:50:39

Copyright (c) 1996, 2023 Oracle. All rights reserved.

Password for user_name@domain:
okinit: Preauthentication failed
```

However, the following succeeds:

```
okinit user_name
Kerberos Utilities for Solaris: Version 23.0.0.0.0 - Production on 15-
MAY-2023 11:50:39
Copyright (c) 1996, 2023 Oracle. All rights reserved.
Password for user_name@domain:
```

The oklist utility lists the user principal from the ticket and as long as a valid ticket is present one can connect in the usual way. After okinit has completed successfully, you can connect to an Oracle Database server without using a user name or password, as follows:.

```
% sqlplus /@service name
```

22.6.5 Enabling Tracing for Kerberos okinit Operations

The KRB5_TRACE environment variable enables you to trace Kerberos okinit operations.

You can use this method verifying any encryption type that has been set using the default tkt enctypes setting in the krb.conf.

1. Run the export command on the KRB5 TRACE environment variable.

For example, for a trace file named krb5.trc:

```
export KRB5 TRACE="/oracle/work/krb5.trc"
```

2. Run the okinit command as follows:

```
okinit user name
```



Output similar to the following appears:

```
Kerberos Utilities for Linux: Version 23.0.0.0.0 - Development on 15-MAY-2023 21:37:39

Copyright (c) 1996, 2023 Oracle. All rights reserved.

Configuration file: /oracle/work/krb/krb.conf.

Password for user_name@US.EXAMPLE.COM:
pfitch@sales us:/oracle/work/
```

3. Use the grep command to find the default that enctype setting in the trace file.

For example:

```
/oracle/work/fgrep aes256-cts krb5.trc
[4072148] 1683321391.149999: Selected etype info: etype aes256-cts, salt
"US.EXAMPLE.COMoratst", params ""
[4072148] 1683321393.375503: AS key obtained from gak fct: aes256-cts/95C0
[4072148] 1683321393.375504: Decrypted AS reply; session key is: aes256-
[4072182] 1683321415.915360: Selected etype info: etype aes256-cts, salt
"US.EXAMPLE.COMoratst", params ""
[4072182] 1683321417.701784: AS key obtained from gak fct: aes256-cts/95C0
[4072182] 1683321417.701785: Decrypted AS reply; session key is: aes256-
cts/859E
[4075441] 1683322653.162464: Selected etype info: etype aes256-cts, salt
"US.EXAMPLE.COMoratst", params ""
[4075441] 1683322656.084028: AS key obtained from gak fct: aes256-cts/1938
[4075455] 1683322659.360899: Selected etype info: etype aes256-cts, salt
"US.EXAMPLE.COMoratst", params ""
[4075455] 1683322661.242404: AS key obtained from gak fct: aes256-cts/95C0
[4075455] 1683322661.242405: Decrypted AS reply; session key is: aes256-
cts/3580
```



Configuring Transport Layer Security Authentication

You can configure Oracle Database to use Transport Layer Security authentication.

- Transport Layer Security and Secure Sockets Layer
 Netscape Communications Corporation designed Transport Layer Security (TLS), previously called Secure Sockets Layer (SSL), to secure network connections.
- How Oracle Database Uses Transport Layer Security for Authentication
 Transport Layer Security works with the core Oracle Database features such as encryption
 and data access controls.
- How Transport Layer Security Works in an Oracle Environment: The TLS Handshake
 When a network connection over Transport Layer Security is initiated, the client and server
 perform a TLS handshake before performing the authentication.
- Public Key Infrastructure in an Oracle Environment
 A public key infrastructure (PKI) is a substrate of network components that provide a security underpinning, based on trust assertions, for an entire organization.
- Transport Layer Security Combined with Other Authentication Methods
 You can configure Oracle Database to use TLS concurrently with database user names
 and passwords, RADIUS, and Kerberos.
- Transport Layer Security and Firewalls
 Oracle Database supports two application proxy-based and stateful packet inspection of firewalls.
- Transport Layer Security Usage Issues
 You should be aware of TLS usage issues, such as communication with other Oracle
 products and types of supported authentication and encryption methods.
- Transport Layer Security Connection without a Client Wallet
 A Transport Layer Security (TLS) connection that uses a common root certificate for the
 database server does not require a client wallet.
- Transport Layer Security Connection with a Client Wallet
 You must configure Transport Layer Security on the server, and then the client.
- Transport Layer Security Oracle Wallet Search Order
 Oracle Database provides several routes for finding the wallet on a server in a Transport Layer Security (TLS) environment.
- Transport Layer Security Connections in an Oracle Real Application Clusters Environment You can configure Transport Layer Security (TLS) connections in an Oracle Real Application Clusters (Oracle RAC) environment by using Oracle RAC tools and modifying Oracle Database configuration files.
- Configuring Transport Layer Security for Client Authentication and Encryption Using Microsoft Certificate Store
 - To perform this configuration with Microsoft Certificate Store (MCS), you use the orapki command-line tool to generate certificates and manipulate the Oracle wallets.
- Troubleshooting the Transport Layer Security Configuration
 Common errors may occur while you use the Oracle Database SSL adapter.

- Certificate Validation with Certificate Revocation Lists
 Oracle provides tools that enable you to validate certificates using certificate revocation lists.
- Configuring Your System to Use Hardware Security Modules
 Oracle Database supports hardware security modules that use APIs that conform to the
 RSA Security, Inc., PKCS #11 specification.

23.1 Transport Layer Security and Secure Sockets Layer

Netscape Communications Corporation designed Transport Layer Security (TLS), previously called Secure Sockets Layer (SSL), to secure network connections.

- The Difference Between Transport Layer Security and Secure Sockets Layer
 Transport Layer Security (TLS) is an incremental version of Secure Sockets Layer (SSL)
 version 3.0.
- Using Transport Layer Security in a Multitenant Environment
 Transport Layer Security (TLS) can be used in a multitenant environment for application containers.

23.1.1 The Difference Between Transport Layer Security and Secure Sockets Layer

Transport Layer Security (TLS) is an incremental version of Secure Sockets Layer (SSL) version 3.0.

Although SSL was primarily developed by Netscape Communications Corporation, the Internet Engineering Task Force (IETF) took over development of it, and renamed it Transport Layer Security (TLS). TLS is an IETF standard.

Oracle Database Security Guide uses the terms Transport Layer Security and TLS instead of Secure Sockets Layer and SSL since the Oracle Database has implemented TLS. However, other documentation in the Oracle Database library may still use the earlier terms Secure Socket Layer and SSL. Where distinctions occur between how you use or configure these protocols, Oracle Database Security Guide specifies what is appropriate for either SSL or TLS.

The Oracle Database software still uses some of the older terminology. For example, the <code>netmgr</code> tool still uses the terms Secure Socket Layer and SSL. Many SSL parameters, such as <code>SSL_SERVER_CERT_DN</code>, use the older terminology. The names of cipher suites and the wording in error messages also use the SSL terminology. However, all these features work with and apply to Transport Layer Security.

23.1.2 Using Transport Layer Security in a Multitenant Environment

Transport Layer Security (TLS) can be used in a multitenant environment for application containers.

If you want to use Transport Layer Security (TLS) in a multitenant environment for an application container, then you must ensure that each PDB is able to use its own wallet with its own certificates for TLS authentication.

- Connect to the PDB that uses the wallet.
- Place the wallet in a subdirectory of the wallet directory where the name of the subdirectory is the GUID of the PDB that uses the wallet.



You must do this because there is no individual sqlnet.ora file for each PDB. For example, suppose the WALLET LOCATION parameter in sqlnet.ora is set as follows:

```
(SOURCE=(METHOD=FILE) (METHOD_DATA=
    (DIRECTORY=/home/oracle/wallet)))
```

Place each PDB's wallet in the /home/oracle/wallet/PDB_GUID directory. You can find the existing PDBs and their GUIDs by querying the DBA PDBS data dictionary view.

If the WALLET_LOCATION parameter is not specified, then you must place the PDB wallet in a leaf subdirectory of the default wallet path where the name of the subdirectory is the GUID of the PDB, and the name of the leaf subdirectory is TLS. For example:

```
$ORACLE_BASE/admin/db_unique_name/PDB_GUID/TLS
```

Or if the ORACLE BASE environment variable is not set, then you can use the Oracle home:

```
$ORACLE HOME/admin/db unique name/PDB GUID/TLS
```

These default locations correspond to the default that is used by Oracle Enterprise User Security to locate wallets for authentication to LDAP.

To enable the PDB to use separate certificates, create a subdirectory under the <code>\$WALLET_LOCATION/PDB_GUID/TLS</code> directory, and then copy the wallets to this subdirectory.

3. Close and then reopen PDB.

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

23.2 How Oracle Database Uses Transport Layer Security for Authentication

Transport Layer Security works with the core Oracle Database features such as encryption and data access controls.

By using Oracle Database TLS functionality to secure communications between clients and servers, you can

- Use TLS to encrypt the connection between clients and servers
- Authenticate any client or server, such as Oracle Application Server 10g, to any Oracle database server that is configured to communicate over TLS

You can use TLS features by themselves or in combination with other authentication methods supported by Oracle Database. For example, you can use the encryption provided by TLS in combination with the authentication provided by Kerberos. TLS supports any of the following authentication modes:

- Only the server authenticates itself to the client
- Both client and server authenticate themselves to each other
- Neither the client nor the server authenticates itself to the other, thus using the TLS encryption feature by itself



The TLS Protocol, version 3.0, published by the Internet Engineering Task Force, for a more detailed discussion of TLS

23.3 How Transport Layer Security Works in an Oracle Environment: The TLS Handshake

When a network connection over Transport Layer Security is initiated, the client and server perform a TLS handshake before performing the authentication.

The handshake process is as follows:

- The client and server establish which cipher suites to use. This includes which encryption algorithms are used for data transfers.
- The server sends its certificate to the client, and the client verifies that the server's certificate was signed by a trusted CA. This step verifies the identity of the server.
- 3. Similarly, if client authentication is required, the client sends its own certificate to the server, and the server verifies that the client's certificate was signed by a trusted CA.
- 4. The client and server exchange key information using public key cryptography. Based on this information, each generates a session key. A key is shared by at least two parties (usually a client and a server) that is used for data encryption for the duration of a single communication session. Session keys are typically used to encrypt network traffic; a client and a server can negotiate a session key at the beginning of a session, and that key is used to encrypt all network traffic between the parties for that session. If the client and server communicate again in a new session, they negotiate a new session key. All subsequent communications between the client and the server is encrypted and decrypted by using this session key and the negotiated cipher suite.

The authentication process is as follows:

- On a client, the user initiates an Oracle Net connection to the server by using TLS.
- 2. TLS performs the handshake between the client and the server.
- If the handshake is successful, then the server verifies that the user has the appropriate authorization to access the database.

23.4 Public Key Infrastructure in an Oracle Environment

A public key infrastructure (PKI) is a substrate of network components that provide a security underpinning, based on trust assertions, for an entire organization.

- About Public Key Cryptography
 Traditional private-key or symmetric-key cryptography requires a single, secret key shared by two or more parties to establish a secure communication.
- Public Key Infrastructure Components in an Oracle Environment
 Public key infrastructure (PKI) components in an Oracle environment include a certificate authority, certificates, certificate revocation lists, and wallets.



23.4.1 About Public Key Cryptography

Traditional private-key or symmetric-key cryptography requires a single, secret key shared by two or more parties to establish a secure communication.

This key is used to both encrypt and decrypt secure messages sent between the parties, requiring prior, secure distribution of the key to each party. The problem with this method is that it is difficult to securely transmit and store the key.

Public-key cryptography provides a solution to this problem, by employing public and private key pairs and a secure method for key distribution. The freely available public key is used to encrypt messages that can *only* be decrypted by the holder of the associated private key. The private key is securely stored, together with other security credentials, in an encrypted container called a wallet.

Public-key algorithms can guarantee the secrecy of a message, but they do not necessarily guarantee secure communications because they do not verify the identities of the communicating parties. To establish secure communications, it is important to verify that the public key used to encrypt a message does in fact belong to the target recipient. Otherwise, a third party can potentially eavesdrop on the communication and intercept public key requests, substituting its own public key for a legitimate key (the third-party attack).

In order to avoid such an attack, it is necessary to verify the owner of the public key, a process called authentication. Authentication can be accomplished through a certificate authority (CA), which is a third party that is trusted by both of the communicating parties.

The CA issues public key certificates that contain an entity's name, public key, and certain other security credentials. Such credentials typically include the CA name, the CA signature, and the certificate effective dates (From Date, To Date).

The CA uses its private key to encrypt a message, while the public key is used to decrypt it, thus verifying that the message was encrypted by the CA. The CA public key is well known and does not have to be authenticated each time it is accessed. Such CA public keys are stored in wallets.

23.4.2 Public Key Infrastructure Components in an Oracle Environment

Public key infrastructure (PKI) components in an Oracle environment include a certificate authority, certificates, certificate revocation lists, and wallets.

Certificate Authority

A certificate authority (CA) is a trusted third party that certifies the identity of entities, such as users, databases, administrators, clients, and servers.

Certificates

A certificate is created when an entity's public key is signed by a trusted certificate authority (CA).

• Certificate Revocation Lists

When a CA signs a certificate binding a public key pair to a user identity, the certificate is valid for a specified time.

Wallets

A wallet is a container that stores authentication and signing credentials, including private keys, certificates, and trusted certificates TLS needs.

Hardware Security Modules

The hardware security modules for SSL include devices to handle various functions and hardware devices to store cryptographic information.

23.4.2.1 Certificate Authority

A certificate authority (CA) is a trusted third party that certifies the identity of entities, such as users, databases, administrators, clients, and servers.

When an entity requests certification, the CA verifies its identity and grants a certificate, which is signed with the CA's private key.

Different CAs may have different identification requirements when issuing certificates. Some CAs may verify a requester's identity with a driver's license, some may verify identity with the requester's fingerprints, while others may require that requesters have their certificate request form notarized.

The CA publishes its own certificate, which includes its public key. Each network entity has a list of trusted CA certificates. Before communicating, network entities exchange certificates and check that each other's certificate is signed by one of the CAs on their respective trusted CA certificate lists.

Network entities can obtain their certificates from the same or different CAs.

Related Topics

Wallets

A wallet is a container that stores authentication and signing credentials, including private keys, certificates, and trusted certificates TLS needs.

23.4.2.2 Certificates

A certificate is created when an entity's public key is signed by a trusted certificate authority (CA).

A certificate ensures that an entity's identification information is correct and that the public key actually belongs to that entity.

A certificate contains the entity's name, public key, and an expiration date, as well as a serial number and certificate chain information. (A certificate chain is an ordered list of certificates containing an end-user or subscriber certificate and its certificate authority certificate.) It can also contain information about the privileges associated with the certificate.

When a network entity receives a certificate, it verifies that it is a trusted certificate, that is, one that has been issued and signed by a trusted certificate authority. A certificate remains valid until it expires or until it is revoked.

23.4.2.3 Certificate Revocation Lists

When a CA signs a certificate binding a public key pair to a user identity, the certificate is valid for a specified time.

However, certain events, such as user name changes or compromised private keys, can render a certificate invalid before the validity period expires. When this happens, the CA revokes the certificate and adds its serial number to a Certificate Revocation List (CRL). The CA periodically publishes CRLs to alert the user population when it is no longer acceptable to use a particular public key to verify its associated user identity.

When servers or clients receive user certificates in an Oracle environment, they can validate the certificate by checking its expiration date, signature, and revocation status. Certificate revocation status is checked by validating it against published CRLs. If certificate revocation status checking is turned on, then the server searches for the appropriate CRL depending on



how this feature has been configured. The server searches for CRLs in the following locations in this order:

- 1. Local file system
- 2. Oracle Internet Directory
- 3. CRL Distribution Point (CRL DP), a location specified in the CRL Distribution Point (CRL DP) X.509, version 3, certificate extension when the certificate is issued. A CRL DP is an optional extension specified by the X.509 version 3 certificate standard, which indicates the location of the Partitioned CRL where revocation information for a certificate is stored. Typically, the value in this extension is in the form of a URL. CRL DPs allow revocation information within a single certificate authority domain to be posted in multiple CRLs. CRL DPs subdivide revocation information into more manageable pieces to avoid proliferating voluminous CRLs, thereby providing performance benefits. For example, a CRL DP is specified in the certificate and can point to a file on a Web server from which that certificate's revocation information can be downloaded.



To use CRLs with other Oracle products, refer to the specific product documentation. This implementation of certificate validation with CRLs is only available in the Oracle Database 12c release 1 (12.1) and later SSL adapter.

Related Topics

Certificate Validation with Certificate Revocation Lists
 Oracle provides tools that enable you to validate certificates using certificate revocation lists.

23.4.2.4 Wallets

A wallet is a container that stores authentication and signing credentials, including private keys, certificates, and trusted certificates TLS needs.

In an Oracle environment, every entity that communicates over TLS must have a wallet containing an X.509 version 3 certificate, private key, and list of trusted certificates, with the exception of Diffie-Hellman.

Security administrators use Oracle Wallet Manager to manage security credentials on the server. Wallet owners use it to manage security credentials on clients. Specifically, you use Oracle Wallet Manager to do the following:

- Generate a public-private key pair and create a certificate request
- Store a user certificate that matches with the private key
- Configure trusted certificates



See Also:

- Oracle Database Enterprise User Security Administrator's Guide for information about Oracle Wallet Manager
- Oracle Database Enterprise User Security Administrator's Guide for information about creating a new Oracle wallet
- Oracle Database Enterprise User Security Administrator's Guide for information about managing trusted certificates in Oracle wallets

23.4.2.5 Hardware Security Modules

The hardware security modules for SSL include devices to handle various functions and hardware devices to store cryptographic information.

Oracle Database uses these devices for the following functions:

- Store cryptographic information, such as private keys
- Perform cryptographic operations to off load RSA operations from the server, freeing the CPU to respond to other transactions

Cryptographic information can be stored on two types of hardware devices:

- (Server-side) Hardware boxes where keys are stored in the box, but managed by using tokens.
- (Client-side) Smart card readers, which support storing private keys on tokens.

An Oracle environment supports hardware devices using APIs that conform to the RSA Security, Inc., Public-Key Cryptography Standards (PKCS) #11 specification.



Currently, SafeNET, nCipher, and Utimaco devices are certified with Oracle Database

Related Topics

Configuring Your System to Use Hardware Security Modules
 Oracle Database supports hardware security modules that use APIs that conform to the
 RSA Security, Inc., PKCS #11 specification.

23.5 Transport Layer Security Combined with Other Authentication Methods

You can configure Oracle Database to use TLS concurrently with database user names and passwords, RADIUS, and Kerberos.

Architecture: Oracle Database and Transport Layer Security
 It is important to understand the architecture of how Oracle Database works with TLS.



How Transport Layer Security Works with Other Authentication Methods
 Transport Layer Security can be used with other authentication methods that Oracle
 Database supports.

23.5.1 Architecture: Oracle Database and Transport Layer Security

It is important to understand the architecture of how Oracle Database works with TLS.

Figure 20-4, which displays the Oracle Database implementation of Transport Layer Security architecture, shows that Oracle Databases operates at the session layer on top of TLS and uses TCP/IP at the transport layer. The session layer is a network layer that provides the services needed by the presentation layer entities that enable them to organize and synchronize their dialogue and manage their data exchange. This layer establishes, manages, and terminates network sessions between the client and server. The transport layer is a networking layer that maintains end-to-end reliability through data flow control and error recovery methods. Oracle Net Services uses Oracle protocol supports for the transport layer.

This separation of functionality lets you employ TLS concurrently with other supported protocols.

Related Topics

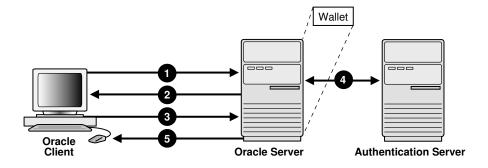
Oracle Database Net Services Administrator's Guide

23.5.2 How Transport Layer Security Works with Other Authentication Methods

Transport Layer Security can be used with other authentication methods that Oracle Database supports.

Figure 23-1 illustrates a configuration in which Transport Layer Security is used in combination with another authentication method.

Figure 23-1 Transport Layer Security in Relation to Other Authentication Methods



In this example, Transport Layer Security is used to establish the initial handshake (server authentication), and an alternative authentication method is used to authenticate the client. The process is as follows:

- The client seeks to connect to the Oracle database server.
- 2. Transport Layer Security performs a handshake during which the server authenticates itself to the client and both the client and server establish which cipher suite to use.
- 3. Once the Transport Layer Security handshake is successfully completed, the user seeks access to the database.



- The Oracle database server authenticates the user with the authentication server using a non-TLS authentication method such as Kerberos or RADIUS.
- Upon validation by the authentication server, the Oracle database server grants access and authorization to the user, and then the user can access the database securely by using TLS.

Related Topics

How Transport Layer Security Works in an Oracle Environment: The TLS Handshake
When a network connection over Transport Layer Security is initiated, the client and server
perform a TLS handshake before performing the authentication.

23.6 Transport Layer Security and Firewalls

Oracle Database supports two application proxy-based and stateful packet inspection of firewalls.

These firewalls are as follows:

- Application proxy-based firewalls: Examples are Network Associates Gauntlet, or Axent Raptor.
- Stateful packet inspection firewalls: Examples are Check Point Firewall-1, or Cisco PIX Firewall

When you enable TLS, stateful inspection firewalls behave like application proxy firewalls because they do not decrypt encrypted packets.

Firewalls do not inspect encrypted traffic. When a firewall encounters data addressed to a TLS port on an intranet server, it checks the target IP address against its access rules and lets the TLS packet pass through to permitted TLS ports, rejecting all others.

23.7 Transport Layer Security Usage Issues

You should be aware of TLS usage issues, such as communication with other Oracle products and types of supported authentication and encryption methods.

Consider the following issues when using TLS:

- TLS use enables secure communication with other Oracle products, such as Oracle Internet Directory.
- Because TLS supports both authentication and encryption, the client/server connection is somewhat slower than the standard Oracle Net TCP/IP transport (using native encryption).
- Each TLS authentication mode requires configuration settings.



If you configure TLS encryption, you must disable non-TLS encryption.

Related Topics

 Configuring Your System to Use Hardware Security Modules
 Oracle Database supports hardware security modules that use APIs that conform to the RSA Security, Inc., PKCS #11 specification. Disabling Strong Authentication and Native Network Encryption
 You can use Oracle Net Manager to disable strong authentication and native network encryption.

23.8 Transport Layer Security Connection without a Client Wallet

A Transport Layer Security (TLS) connection that uses a common root certificate for the database server does not require a client wallet.

- About Transport Layer Security Connections without a Client Wallet
 You can configure a Transport Layer Security (TLS) connection without a client wallet if
 your environment meets certain requirements.
- Configuring a Transport Layer Security Connection without a Client Wallet
 Before you can configure Transport Layer Security (TLS) without using client wallets, you must ensure that the database does not require client authentication.

23.8.1 About Transport Layer Security Connections without a Client Wallet

You can configure a Transport Layer Security (TLS) connection without a client wallet if your environment meets certain requirements.

Consider using a TLS connection without a client wallet if your environment meets these requirements:

- The client certificate is not used as a means of user authentication to the database. Only the server certificate is required to establish a TLS connection.
- The server certificate was issued by a certificate authority (CA) whose certificate is available in the system's default certificate store (common root certificate).
- The Oracle database server is configured to allow TLS connections. (Set SSL CLIENT AUTHENTICATION=FALSE).

This is the most common type of configuration as long as the root certificate for the database server already exists in the local system certificate store. This configuration can be used for both cloud and on-premises databases. This configuration enables the client to verify server certificates without having to configure its own wallet.

Note the following:

- For the C and Instant Client database drivers (and therefore, SQL*Plus), the walletless feature is only available on Microsoft Windows and Linux x64.
- For the JDBC-thin driver, the walletless feature is available on all platforms.

Related Topics

Oracle Database Net Services Administrator's Guide



23.8.2 Configuring a Transport Layer Security Connection without a Client Wallet

Before you can configure Transport Layer Security (TLS) without using client wallets, you must ensure that the database does not require client authentication.

For the C and Instant Client database drivers (and therefore, SQL*Plus), the walletless feature is only available on Microsoft Windows and Linux x64. For the JDBC-thin driver, the walletless feature is available on all platforms.

- 1. Log in to the server where the Oracle database resides.
- 2. Check the sqlnet.ora file for the following settings:
 - AUTHENTICATION SERVICES=(tcps), which will require TLS for database connections.
 - SSL_CLIENT_AUTHENTICATION=FALSE, whose default is TRUE, which will require mTLS
 (mutual TLS requiring a client certificate in a client wallet). Setting this parameter to
 FALSE enables both TLS and mTLS connections from the client. If it is set to TRUE, then
 it is always mTLS.

By default, the sqlnet.ora file is located in the $poracle_home/dbs$ directory or in the location set by the TNS ADMIN environment variable.

- 3. Ensure that the server wallet exists in the default location, defined by the WALLET_ROOT system parameter, or in the WALLET LOCATION sqlnet.ora parameter.
- 4. Check the listener.ora file to ensure TLS is specified.

```
LISTENER = (ADDRESS=(PROTOCOL=tcps) (HOST=) (PORT=1234))
```

5. Ensure that the listener wallet also exists the default location, set by the WALLET_ROOT system parameter, or in the WALLET LOCATION sqlnet.ora parameter.

If you are creating a new client connection, then edit the listener.ora file to have the following setting:

```
ADDRESS=(PROTOCOL=tcps)
```

By default, listener.ora is located in the <code>\$ORACLE HOME/network/admin directory</code>.

- Log in to the client for the Oracle database.
- 7. Modify the client sqlnet.ora and tnsnames.ora files.
 - Edit the SQLNET.SSL_CLIENT_AUTHENTICATION setting in the sqlnet.ora file.

 Set the SQLNET.SSL_CLIENT_AUTHENTICATION=FALSE, because the default is TRUE.

 FALSE enables the client to create the connection using either TLS or mTLS. A setting of FALSE, will not send information about the client side private certificate. Because this applies to every connection, you can change the SSL_CLIENT_AUTHENTICATION parameter in the tnsnames.ora connection string using the same parameter setting. If SSL_CLIENT_AUTHENTICATION=TRUE, then it can only configure mTLS. This setting is optional.
 - If you connect to multiple databases and some require mTLS with a client wallet, then you can have two options for setting different connections with and without a client wallet, as follows:



Option 1: Set WALLET_LOCATION in sqlnet.ora for a common wallet. Then use WALLET_LOCATION in your connect string (in thishames.ora or directly on the command line) to override the setting in sqlnet.ora. You can specify a different wallet location for a connection or tell the connection to use the system default keystore instead. Use the following parameter to change wallet location to the system default keystore:

```
net_service_name = (DESCRIPTION=(ADDRESS = (PROTOCOL=tcps)
(HOST=host_name) (PORT=port)) (SECURITY=(WALLET_LOCATION=SYSTEM))
(CONNECT_DATA=(SERVICE_NAME=service_name)))
```

The default certificate store is located in /etc/pki/tls/cert.pem for Linux or in the Microsoft Certificate Store for Microsoft Windows. You cannot change the default location of the certificate store. By default, tnsnames.ora is located in the \$ORACLE HOME/network/admin directory.

Option 2: Only specify WALLET_LOCATION as part of the connections that need to use a client wallet. Do not specify WALLET_LOCATION in sqlnet.ora. Connections that do not need to use a client wallet will automatically use the local default system keystore if WALLET_LOCATION is not specified in the sqlnet.ora file. For example:

```
net_service_name = (DESCRIPTION=(ADDRESS = (PROTOCOL=tcps)
(HOST=host_name) (PORT=port))
(SECURITY=(WALLET_LOCATION=wallet_file_directory))
(CONNECT_DATA=(SERVICE_NAME=service_name)))
```

8. In SQL*Plus, to determine if the database connections are using TLS, check the connections by performing the following guery.

```
SELECT SYS CONTEXT ('USERENV', 'NETWORK PROTOCOL') FROM DUAL;
```

Output similar to the following should appear:

```
SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
-----tcps
```

Related Topics

Oracle Database Reference

23.9 Transport Layer Security Connection with a Client Wallet

You must configure Transport Layer Security on the server, and then the client.

- Step 1: Configure Transport Layer Security on the Server
 During installation, Oracle sets defaults on the Oracle database server and the Oracle client for TLS parameters, except the Oracle wallet location.
- Step 2: Configure Transport Layer Security on the Client
 When you configure SSL on the client, you configure the server DNs and use TCP/IP with TLS on the client.

Step 3: Log in to the Database Instance
 After you have completed the configuration, you are ready to log in to the database.

23.9.1 Step 1: Configure Transport Layer Security on the Server

During installation, Oracle sets defaults on the Oracle database server and the Oracle client for TLS parameters, except the Oracle wallet location.

- Step 1A: Confirm Wallet Creation on the Server
 Before proceeding to the next step, confirm that a wallet has been created and that it has a
 certificate.
- Step 1B: Specify the Database Wallet Location on the Server
 Next, you are ready to specify a location on the server for the wallet.
- Step 1C: Set the Transport Layer Security Cipher Suites on the Server (Optional)
 Optionally, you can set the Transport Layer Security cipher suites.
- Step 1D: Set the Required Transport Layer Security Version on the Server (Optional)

 The SSL_VERSION parameter defines the version of TLS that must run on the systems with which the server communicates.
- Step 1E: Set Transport Layer Security Client Authentication on the Server (Optional)
 The SSL_CLIENT_AUTHENTICATION parameter controls whether the client is authenticated using TLS.
- Step 1F: Set Transport Layer Security as an Authentication Service on the Server (Optional)

The SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file sets the TLS authentication service.

- Step 1G: Disable SSLv3 on the Server and Client (Optional) SSLv3 refers to Secure Sockets Layer version 3.
- Step 1H: Create a Listening Endpoint that Uses TCP/IP with Transport Layer Security on the Server

You can configure a listening endpoint to use TCP/IP with TLS on the server.

• Step 1H: Restart the Database
To complete the configuration of Transport Layer Security on the server, you must restart
the database.

23.9.1.1 Step 1A: Confirm Wallet Creation on the Server

Before proceeding to the next step, confirm that a wallet has been created and that it has a certificate.

- Start Oracle Wallet Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command:
 - (Windows) Select Start, Programs, Oracle-HOME_NAME, Integrated Management Tools, Wallet Manager

Oracle Wallet Manager is deprecated with Oracle Database 21c. Instead of using Oracle Wallet Manager, Oracle recommends that you use the command line tools <code>orapki</code> and <code>mkstore</code>.

2. From the Wallet menu, select Open.



The wallet should contain a certificate with a status of Ready and auto-login turned on. If auto-login is not on, then select it from the **Wallet** menu and save the wallet again. This turns auto-login on.

Related Topics

Oracle Database Enterprise User Security Administrator's Guide

23.9.1.2 Step 1B: Specify the Database Wallet Location on the Server

Next, you are ready to specify a location on the server for the wallet.

- 1. Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- 3. From the Naming list, select Network Security.

The Network Security tabbed window appears.

- Select the SSL tab (which works for TLS) and then select Configure SSL for: Server.
- In the Wallet Directory box, enter the directory in which the Oracle wallet is located or click Browse to find it by searching the file system.

Note that if you are configuring the database-to-directory TLS connection for Enterprise User Security, then Database Configuration Assistant automatically creates a database wallet while registering the database with the directory. You must use that wallet to store the database PKI credentials for TLS-authenticated Enterprise User Security.

Important:

- Use Oracle Wallet Manager to create the wallet. See Oracle Database Enterprise User Security Administrator's Guide for information about creating a new Oracle wallet.
- Use Oracle Net Manager to set the wallet location in the sqlnet.ora file. Be aware that
 in a multitenant environment, the settings in the sqlnet.ora file apply to all pluggable
 databases (PDBs).

Ensure that you enter the same wallet location when you create it and when you set the location in the sqlnet.ora file.

6. From the File menu, select Save Network Configuration.

The sqlnet.ora and listener.ora files are updated with the following entries:

```
wallet_location =
  (SOURCE=
   (METHOD=File)
  (METHOD_DATA=
      (DIRECTORY=wallet_location)))
```



Note:

The listener uses the wallet defined in the <code>listener.ora</code> file. It can use any database wallet. When SSL is configured for a server using Net Manager, the wallet location is entered into the <code>listener.ora</code> and the <code>sqlnet.ora</code> files. The <code>listener.ora</code> file is not relevant to the Oracle client.

To change the listener wallet location so that the listener has its own wallet, you can edit listener.ora to enter the new location.

23.9.1.3 Step 1C: Set the Transport Layer Security Cipher Suites on the Server (Optional)

Optionally, you can set the Transport Layer Security cipher suites.

- About the Transport Layer Security Cipher Suites
 A cipher suite is a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network entities.
- TLS Cipher Suite Authentication, Encryption, Integrity, and TLS Versions
 Oracle Database supports a set of cipher suites that are set by default when you install
 Oracle Database.
- Specifying Transport Layer Security Cipher Suites for the Database Server
 First, you must specify the Transport Layer Security cipher suites for the database server.

23.9.1.3.1 About the Transport Layer Security Cipher Suites

A cipher suite is a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network entities.

During a Transport Layer Security handshake, two entities negotiate to see which cipher suite they will use when transmitting messages back and forth.

When you install Oracle Database, the Transport Layer Security cipher suites are set for you by default and negotiated in the order they are listed. You can override the default order by setting the <code>SSL_CIPHER_SUITES</code> parameter. Ensure that you enclose the <code>SSL_CIPHER_SUITES</code> parameter setting in parentheses (for example,

SSL_CIPHER_SUITES=(tls_rsa_with_aes_128_cbc_sha256)). Otherwise, the cipher suite setting will not parse correctly.

You can prioritize the cipher suites. When the client negotiates with servers regarding which cipher suite to use, it follows the prioritization you set. When you prioritize the cipher suites, consider the following:

- **Compatibility.** Server and client must be configured to use compatible cipher suites for a successful connection.
- **Cipher priority and strength.** Prioritize cipher suites starting with the strongest and moving to the weakest to ensure the highest level of security possible.
- The level of security you want to use.
- The impact on performance.



Note:

Regarding Diffie-Hellman anonymous authentication:

- If you set the server to employ this cipher suite, then you must also set the same cipher suite on the client. Otherwise, the connection fails.
- If you use a cipher suite employing Diffie-Hellman anonymous, then you
 must configure the TLS client authentication on the server by setting the
 SSL CLIENT AUTHENTICATION parameter to FALSE.
- There is a known bug in which an OCI client requires a wallet even when using a cipher suite with DH_ANON, which does not authenticate the client.

Related Topics

- TLS Cipher Suite Authentication, Encryption, Integrity, and TLS Versions
 Oracle Database supports a set of cipher suites that are set by default when you install
 Oracle Database.
- Step 1E: Set Transport Layer Security Client Authentication on the Server (Optional)
 The SSL_CLIENT_AUTHENTICATION parameter controls whether the client is authenticated using TLS.

23.9.1.3.2 TLS Cipher Suite Authentication, Encryption, Integrity, and TLS Versions

Oracle Database supports a set of cipher suites that are set by default when you install Oracle Database.

Table 23-1 lists the authentication, encryption, and data integrity types each cipher suite uses.

Table 23-1 Transport Layer Security Cipher Suites

Cipher Suites	Authentication	Encryption	Data Integrity	TLS Compatibilit y
TLS_ECDHE_ECDSA_WITH_AES_128_GCM_SHA256	ECDHE_ECDSA	AES 128 GCM	SHA256 (SHA-2)	TLS 1.2 only
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA	ECDHE_ECDSA	AES 128 CBC	SHA-1	TLS 1.0 and later
TLS_ECDHE_ECDSA_WITH_AES_128_CBC_SHA256	ECDHE_ECDSA	AES 128 CBC	SHA256 (SHA-2)	TLS 1.2 only
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA	ECDHE_ECDSA	AES 256 CBC	SHA-1	TLS 1.0 and later
TLS_ECDHE_ECDSA_WITH_AES_256_CBC_SHA384	ECDHE_ECDSA	AES 256 CBC	SHA384 (SHA-2)	TLS 1.2 only
TLS_ECDHE_ECDSA_WITH_AES_256_GCM_SHA384	ECDHE_ECDSA	AES 256 GCM	SHA384 (SHA-2)	TLS 1.2 only
TLS_RSA_WITH_AES_128_CBC_SHA256	RSA	AES 128 CBC	SHA256 (SHA-2)	TLS 1.2 only
TLS_RSA_WITH_AES_128_GCM_SHA256	RSA	AES 128 GCM	SHA256 (SHA-2)	TLS 1.2 only
TLS_RSA_WITH_AES_128_CBC_SHA	RSA	AES 128 CBC	SHA-1	TLS 1.0 only



Table 23-1 (Con	:.) Transport	Layer Security	Cipher Suites
-----------------	---------------	----------------	---------------

Cipher Suites	Authentication	Encryption	Data Integrity	TLS Compatibilit y
TLS_RSA_WITH_AES_256_CBC_SHA	RSA	AES 256 CBC	SHA-1	TLS 1.0 and later
TLS_RSA_WITH_AES_256_CBC_SHA256	RSA	AES 256 CBC	SHA256 (SHA-2)	TLS 1.2 only
TLS_RSA_WITH_AES_256_GCM_SHA384	RSA	AES 256 GCM	SHA384 (SHA-2)	TLS 1.2 only

Table 23-2 lists cipher suites that you can use, but be aware that they do not the provide authentication of the communicating parties, and hence can be vulnerable to third-party attacks. Oracle recommends that you do not use these cipher suites to protect sensitive data. However, they are useful if the communicating parties want to remain anonymous or simply do not want the overhead caused by mutual authentication.

Table 23-2 SSL_DH Transport Layer Security Cipher Suites

Cipher Suites	Authentication	Encryption	Data Integrity	TLS Compatibilit y
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH anon	3DES EDE CBC	SHA-1	TLS 3.0 and later

23.9.1.3.3 Specifying Transport Layer Security Cipher Suites for the Database Server

First, you must specify the Transport Layer Security cipher suites for the database server.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- 3. From the Naming list, select Network Security.

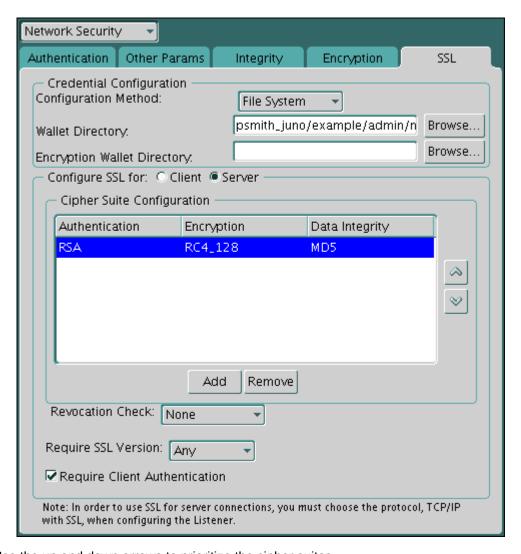
The Network Security tabbed window appears.

- Select the SSL tab and then select Configure SSL for: Server.
- 5. In the Cipher Suite Configuration area, click Add.

A dialog box displays available cipher suites. To see the US domestic cipher suites, click the **Show US Domestic Cipher Suits** check box.

6. Select a suite and click **OK**.

The **Cipher Suite Configuration** list is updated:



- Use the up and down arrows to prioritize the cipher suites.
- 8. From the **File** menu, select **Save Network Configuration**.

The sqlnet.ora file is updated with the following entry:

SSL CIPHER SUITES= (SSL cipher suite1 [,SSL cipher suite2])

23.9.1.4 Step 1D: Set the Required Transport Layer Security Version on the Server (Optional)

The $\tt SSL_VERSION$ parameter defines the version of TLS that must run on the systems with which the server communicates.

Optionally, you can set the SSL VERSION parameter in the sqlnet.ora or the listener.ora file.

You can require these systems to use any valid version. The default setting for this parameter in sqlnet.ora is undetermined, which is set by selecting **Any** from the list in the SSL tab of the Network Security window.

- In the Require SSL Version list, the default is Any.
 Accept this default or select the SSL version you want to use.
- 2. From the File menu, select Save Network Configuration.

If you chose **Any**, then the sqlnet.ora file is updated with the following entry:

SSL VERSION=UNDETERMINED



SSL 2.0 is not supported on the server side.

Related Topics

Oracle Database Net Services Reference

23.9.1.5 Step 1E: Set Transport Layer Security Client Authentication on the Server (Optional)

The SSL_CLIENT_AUTHENTICATION parameter controls whether the client is authenticated using TLS.

You must set this parameter in the sqlnet.ora file on the server. The default value of SSL CLIENT AUTHENTICATION parameter is TRUE.

You can set the SSL_CLIENT_AUTHENTICATION to FALSE if you are using a cipher suite that contains Diffie-Hellman anonymous authentication (DH anon).

Also, you can set this parameter to FALSE for the client to authenticate itself to the server by using any of the non-SSL authentication methods supported by Oracle Database, such as Kerberos or RADIUS.

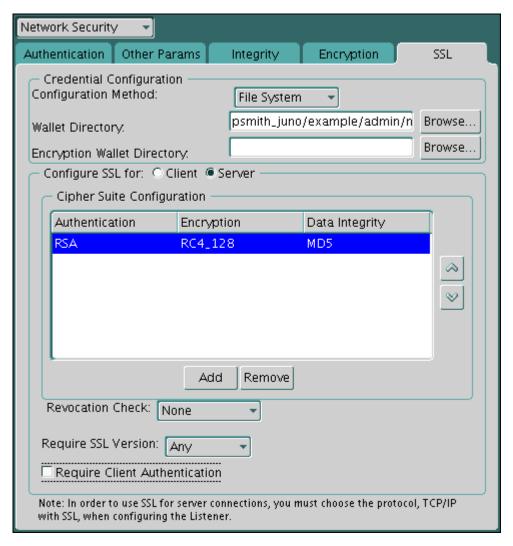


There is a known bug in which an OCI client requires a wallet even when using a cipher suite with DH_ANON, which does not authenticate the client.

To set SSL CLIENT AUTHENTICATION to FALSE on the server:

1. In the SSL page Oracle Net Manager, deselect Require Client Authentication.





2. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entry:

SSL CLIENT AUTHENTICATION=FALSE

23.9.1.6 Step 1F: Set Transport Layer Security as an Authentication Service on the Server (Optional)

The SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file sets the TLS authentication service.

Set this parameter if you want to use TLS authentication in conjunction with another authentication method supported by Oracle Database. For example, use this parameter if you want the server to authenticate itself to the client by using TLS and the client to authenticate itself to the server by using Kerberos.

• To set the SQLNET.AUTHENTICATION_SERVICES parameter on the server, add TCP/IP with TLS (TCPS) to this parameter in the sqlnet.ora file by using a text editor. For example, if you want to use SSL authentication in conjunction with RADIUS authentication, set this parameter as follows:

SQLNET.AUTHENTICATION SERVICES = (TCPS, radius)

If you do not want to use TLS authentication in conjunction with another authentication method, then do not set this parameter.

23.9.1.7 Step 1G: Disable SSLv3 on the Server and Client (Optional)

SSLv3 refers to Secure Sockets Layer version 3.

Applications that support Secure Sockets Layer version 3 (SSLv3) are vulnerable to Padding Oracle On Downgraded Legacy Encryption (POODLE) attacks, even if they use the most recent version of Transport Layer Security (TLS). To prevent POODLE attacks, you should set the ADD_SSLV3_TO_DEFAULT sqlnet.ora parameter to FALSE on both the server and the client. ADD_SSLV3_TO_DEFAULT only applies if the SSL_VERSION parameter is not set. (which means that you are using the default list of SSL versions).

- Log in to the database server or the client server.
- 2. Edit the sqlnet.ora parameter file, which by default is located in the <code>\$ORACLE_HOME/network/admin</code> directory, to include the <code>ADD_SSLV3_TO_DEFAULT</code> parameter, as follows:

```
ADD SSLV3 TO DEFAULT=false
```

ADD_SSLV3_TO_DEFAULT defaults to FALSE, and has no effect if SSL_VERSION is explicitly set. So, by default, without SSL_VERSION explicitly set, SSLv3 connections will not be allowed. Setting ADD_SSLV3_TO_DEFAULT to TRUE would allow SSLv3 connections to continue working by default.

23.9.1.8 Step 1H: Create a Listening Endpoint that Uses TCP/IP with Transport Layer Security on the Server

You can configure a listening endpoint to use TCP/IP with TLS on the server.

- Configure the listener in the listener.ora file. Oracle recommends using port number 2484 for typical Oracle Net clients.
- 2. Restart the database.

Related Topics

Oracle Database Net Services Reference

23.9.1.9 Step 1H: Restart the Database

To complete the configuration of Transport Layer Security on the server, you must restart the database.

For example:

SHUTDOWN IMMEDIATE STARTUP

23.9.2 Step 2: Configure Transport Layer Security on the Client

When you configure SSL on the client, you configure the server DNs and use TCP/IP with TLS on the client.



- Step 2A: Confirm Client Wallet Creation
 - You must confirm that a wallet has been created on the client and that the client has a valid certificate.
- Step 2B: Configure Server DN Matching and Use TCP/IP with TLS on the Client
 Next, you are ready to configure server DN matching and use TCP/IP with Transport Layer
 Security (TLS) on the client.
- Step 2C: Specify Required Client TLS Configuration (Wallet Location)
 You can use Oracle Net Manager to specify the required client TLS configuration.
- Step 2D: Set the Client Transport Layer Security Cipher Suites (Optional)
 Optionally, you can set the Transport Layer Security cipher suites. Oracle Database provides default cipher suite settings.
- Step 2E: Set the Required TLS Version on the Client (Optional)

 The SSL_VERSION parameter defines the version of TLS that must run on the systems with which the client communicates.
- Step 2F: Set TLS as an Authentication Service on the Client (Optional)

 The SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file sets the TLS authentication service.
- Step 2G: Specify the Certificate to Use for Authentication on the Client (Optional)

 If you have multiple certificates, then you can set the SQLNET.SSL_EXTENDED_KEY_USAGE parameter in the sqlnet.ora file to specify the correct certificate.
- Step 2H: Check That the Connections Are Using Transport Layer Security
 You can query the V\$SESSION and V\$SESSION_CONNECT_INFO dynamic views to ensure that
 the client connections are using Transport Layer Security (TLS).
- Step 2I: Restart the Database
 To complete the configuration of Transport Layer Security on the client, you must restart
 the database.

23.9.2.1 Step 2A: Confirm Client Wallet Creation

You must confirm that a wallet has been created on the client and that the client has a valid certificate.

 Use Oracle Wallet Manager to check that the wallet has been created by selecting the Wallet menu, then Open.

Oracle recommends that you use Oracle Wallet Manager to remove the trusted certificate in your Oracle wallet that is associated with each certificate authority that you do not use.

Related Topics

- Step 1A: Confirm Wallet Creation on the Server
 Before proceeding to the next step, confirm that a wallet has been created and that it has a
 certificate.
- Oracle Database Enterprise User Security Administrator's Guide

23.9.2.2 Step 2B: Configure Server DN Matching and Use TCP/IP with TLS on the Client

Next, you are ready to configure server DN matching and use TCP/IP with Transport Layer Security (TLS) on the client.



- About Configuring the Server DN Matching and Using TCP/IP with TLS on the Client
 In addition to validating the server certificate's certificate chain, you can perform an extra
 check through server DN matching.
- Configuring the Server DN Matching and Using TCP/IP with TLS on the Client You must edit the tnsnames.ora and listener.ora files to configure the server DN matching and user TCP/IP with TLS on the client.
- Use of the SSL_ALLOW_WEAK_DN_MATCH Parameter to Control
 SSL_SERVER_DN_MATCH
 The SSL_ALLOW_WEAK_DN_MATCH parameter controls how the SSL_SERVER_DN_MATCH parameter allows the service name for partial distinguished name matching and check the database server certificate.

23.9.2.2.1 About Configuring the Server DN Matching and Using TCP/IP with TLS on the Client

In addition to validating the server certificate's certificate chain, you can perform an extra check through server DN matching.

You can configure the Oracle Net Service name to include server DN matching and to use TCP/IP with TLS on the client. To accomplish this, you must specify the server's distinguished name (DN) and TCPS as the protocol in the client network configuration files to enable server DN matching and TCP/IP with TLS connections. Server DN matching is optional, but Oracle recommends it because it adds a layer of security to the client: the client can then perform this check against the server.

Due to changes in the CA certificate format where the Organization Unit (OU) field will be removed starting in 2022, you may need to update your server certificate DN if you have set <code>SSL_SERVER_DN_MATCH</code> to <code>TRUE</code>. After you receive the new server certificate with the OU removed from the DN, you must update the client <code>SSL_SERVER_CERT_DN</code> parameter to match the new DN.

You can configure either partial DN matching or full DN matching. After you set the SSL_SERVER_DN_MATCH parameter to TRUE, then partial DN matching is performed automatically. The client will then check the server certificate for the DN information. Full DN matching enables the client to match against the complete DN of the server. If you want to perform a full DN match, then you must specify the server's DN in the SSL_SERVER_CERT_DN parameter. To enable either of these methods, you must set the SSL SERVER DN MATCH parameter to TRUE.

The ability to use either partial or full DN matching enables more flexibility based on how you create and manage host certificates. For example, suppose the client tries to connect to a server with hostname=finance.us.example.com. With partial DN matching, the client checks the server's certificate to verify that CN=finance.us.example.com is part of the server's DN.

You must manually edit the <code>tnsnames.ora</code> client network configuration file to specify the server's DN and the TCP/IP with SSL protocol. The <code>tnsnames.ora</code> file can be located on the client or in the LDAP directory. If it is located on the server, then it typically resides in the same directory as the <code>listener.ora</code> file. The <code>tnsnames.ora</code> file is typically located in the setting specified by the <code>TNS_ADMIN</code> environment variable. If <code>TNS_ADMIN</code> is not set, then <code>tnsnames.ora</code> resides in the following directory locations:

- (UNIX) \$ORACLE HOME/network/admin/
- (Windows) ORACLE BASE\ORACLE HOME\network\admin\



23.9.2.2.2 Configuring the Server DN Matching and Using TCP/IP with TLS on the Client

You must edit the tnsnames.ora and listener.ora files to configure the server DN matching and user TCP/IP with TLS on the client.

- In the client tnsnames.ora file, search for the SSL_SERVER_CERT_DN parameter and do the following:
 - If you want to use full DN matching, then set SSL_SERVER_CERT_DN to the complete DN, similar to the following:

```
(SECURITY=
(SSL SERVER CERT DN="finance,cn=OracleContext,c=us,o=example"))
```

The client uses this information to obtain the list of DNs it expects for each of the servers, enforcing the server's DN to match its service name. The following example shows an entry for the finance database in the tnsnames.ora file.

```
finance=
(DESCRIPTION=
(ADDRESS_LIST=
(ADDRESS= (PROTOCOL = tcps) (HOST = finance) (PORT = 1575)))
(CONNECT_DATA=
(SERVICE_NAME= finance.us.example.com))
(SECURITY=
(SSL SERVER CERT DN="cn=finance,cn=OracleContext,c=us,o=example"))
```

By default, the tnsnames.ora and listener.ora files are in the <code>\$ORACLE_HOME/</code> network/admin directory on UNIX systems and in <code>ORACLE_HOME/</code> network/admin on Windows

- If you plan to use partial DN matching, then do not include the SSL_SERVER_CERT_DN parameter in tnsnames.ora.
- 2. In the client thin the same some file, enter tops as the PROTOCOL in the ADDRESS parameter.

This specifies that the client will use TCP/IP with TLS to connect to the database that is identified in the <code>SERVICE_NAME</code> parameter. The following also shows an entry that specifies TCP/IP with TLS as the connecting protocol in the <code>tnsnames.ora</code> file.

```
LISTENER=
(DESCRIPTION_LIST=
(DESCRIPTION=
(ADDRESS= (PROTOCOL = tcps) (HOST = finance) (PORT = 1575))))
```

3. In the listener.ora file, enter tops as the PROTOCOL in the ADDRESS parameter.

23.9.2.2.3 Use of the SSL_ALLOW_WEAK_DN_MATCH Parameter to Control SSL SERVER DN MATCH

The SSL_ALLOW_WEAK_DN_MATCH parameter controls how the SSL_SERVER_DN_MATCH parameter allows the service name for partial distinguished name matching and check the database server certificate.

Starting in Oracle Database 23c, the behavior of the <code>SSL_SERVER_DN_MATCH</code> parameter has changed. Previously, only the database server certificate was checked for DN matching. With Oracle Database 23c, the listener and server certificates are both checked. Also, the <code>SERVICE_NAME</code> setting is not used to check during partial DN match anymore. The <code>HOST</code> setting can still be used for partial DN matching with the certificate DN and subject alternative name (SAN), on both the listener and server certificates.

You can set SSL ALLOW WEAK DN MATCH as follows:

- TRUE enables SSL_SERVER_DN_MATCH to check the database server certificate (but not the listener) and enable the service name to be used for partial DN matching. The search order on the client side is as follows: first, the host name, then the subject alternative name (SAN), and then the service name.
- FALSE (the default) disables SSL_SERVER_DN_MATCH from checking service name matching. Instead, matching on the client side is based on a search for the HOST setting in the certificate DN, and if that is not available, then in the subject alternative name (SAN) field (but not the service name). The DN check is performed on the listener and the server.

If you used the service name for partial DN matching previously, then you must either get a new certificate or set <code>SSL_ALLOW_WEAK_DN_MATCH</code> to <code>TRUE</code> to revert to the pre-release 23c behavior. You are most likely using the same certificate for both the database server and listener, but if you are not, then you will either need to do one of the following:

- Get a new certificate (use the orapki cert create command for self-signed certificates).
- Change or remove the DN matching strategy.
- Set the SSL_ALLOW_WEAK_DN_MATCH parameter to TRUE to revert SSL_SERVER_DN_MATCH to its
 older behavior.

When you set SSL ALLOW WEAK DN MATCH to TRUE, note the following:

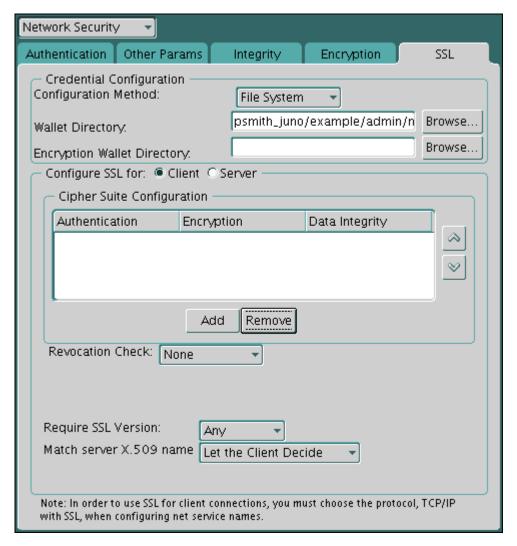
- When the client performs a full DN match (SSL_SERVER_MATCH=TRUE, SSL_SERVER_CERT_DN="certificate_DN"), then only the database server certificate DN will need to match the SSL_SERVER_CERT_DN value.
- When the client performs a partial DN match (SSL_SERVER_MATCH=TRUE, SSL_SERVER_CERT_DN is not set), then Oracle Database will compare the connect string parameter HOST to the common name (CN) of the database server certificate DN and the certificate subject alternate names field (SAN). If there is no partial match, then Oracle Database will continue and check the SERVICE_NAME parameter with the CN.

23.9.2.3 Step 2C: Specify Required Client TLS Configuration (Wallet Location)

You can use Oracle Net Manager to specify the required client TLS configuration.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.The Network Security tabbed window appears.
- 4. Select the SSL tab.
- Select Configure SSL for: Client.





- 6. In the Wallet Directory box, enter the directory in which the Oracle wallet is located, or click **Browse** to find it by searching the file system.
- 7. From the Match server X.509 name list, select one of the following options:
 - Yes: Requires that the server's distinguished name (DN) match its service name. TLS
 ensures that the certificate is from the server and connections succeed only if there is
 a match.

This check can be made only when RSA ciphers are selected, which is the default setting.

- No (default): TLS checks for a match between the DN and the service name, but does
 not enforce it. Connections succeed regardless of the outcome but an error is logged if
 the match fails.
- Let Client Decide: Enables the default.

The following alert is displayed when you select **No**:

```
Security Alert
Not enforcing the server X.509 name match allows a server to potentially fake its identity. Oracle recommends selecting YES for this option so that connections are refused when there is a mismatch.
```

8. From the File menu, select Save Network Configuration.

The sglnet.ora file on the client is updated with the following entries:

```
SSL_CLIENT_AUTHENTICATION =TRUE
wallet_location =
  (SOURCE=
   (METHOD=File)
   (METHOD_DATA=
        (DIRECTORY=wallet_location)))
SSL_SERVER_DN_MATCH=(ON/OFF)
```

Related Topics

Transport Layer Security X.509 Server Match Parameters

The SSL_SERVER_DN_MATCH and SSL_SERVER_CERT_DN parameters validate the identity of the server to which a client connects.

23.9.2.4 Step 2D: Set the Client Transport Layer Security Cipher Suites (Optional)

Optionally, you can set the Transport Layer Security cipher suites. Oracle Database provides default cipher suite settings.

- About Setting the Client Transport Layer Security Cipher Suites
 A cipher suite is a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network entities.
- Setting the Client Transport Layer Security Cipher Suites
 You can use Oracle Net Manager to set the client TLS cipher suites.

23.9.2.4.1 About Setting the Client Transport Layer Security Cipher Suites

A cipher suite is a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network entities.

During an SSL handshake, two entities negotiate to see which cipher suite they will use when transmitting messages back and forth.

When you install Oracle Database, the TLS cipher suites are set for you by default. This table lists them in the order they are tried when two entities are negotiating a connection. You can override the default by setting the <code>SSL_CIPHER_SUITES</code> parameter. For example, if you use Oracle Net Manager to add the cipher suite <code>SSL_RSA_WITH_RC4_128_SHA</code>, all other cipher suites in the default setting are ignored.

You can prioritize the cipher suites. When the client negotiates with servers regarding which cipher suite to use, it follows the prioritization you set. When you prioritize the cipher suites, consider the following:

- The level of security you want to use. For example, AES encryption is stronger than DES.
- The impact on performance. For example, triple-DES encryption is slower than DES.
- Administrative requirements. The cipher suites selected for a client must be compatible
 with those required by the server. For example, in the case of an Oracle Call Interface
 (OCI) user, the server requires the client to authenticate itself. You cannot, in this case, use
 a cipher suite employing Diffie-Hellman anonymous authentication, which disallows the
 exchange of certificates.

You typically prioritize cipher suites starting with the strongest and moving to the weakest.

The currently supported Transport Layer Security cipher suites are set by default when you install Oracle Database. The table also lists the authentication, encryption, and data integrity types each cipher suite uses.



If the ${\tt SSL_CLIENT_AUTHENTICATION}$ parameter is set to true in the ${\tt sqlnet.ora}$ file, then disable all cipher suites that use Diffie-Hellman anonymous authentication. Otherwise, the connection fails.

Related Topics

TLS Cipher Suite Authentication, Encryption, Integrity, and TLS Versions
 Oracle Database supports a set of cipher suites that are set by default when you install
 Oracle Database.

23.9.2.4.2 Setting the Client Transport Layer Security Cipher Suites

You can use Oracle Net Manager to set the client TLS cipher suites.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- 3. From the **Naming** list, select **Network Security**.

The Network Security tabbed window appears.

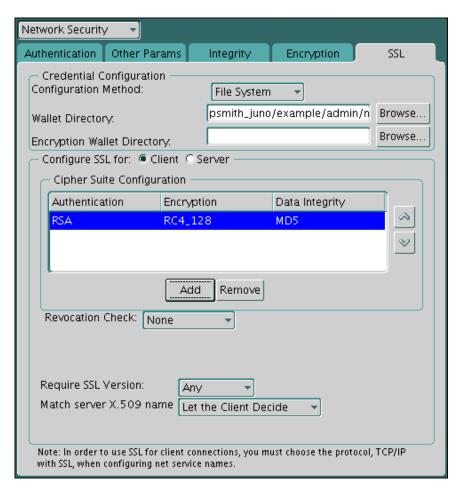
- Select the SSL tab.
- 5. In the Cipher Suite Configuration region, click Add.

A dialog box displays available cipher suites.

6. Select a suite and click OK.

The **Cipher Suite Configuration** list is updated as shown as follows:





- 7. Use the up and down arrows to prioritize the cipher suites.
- 8. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entry:

SSL_CIPHER_SUITES= (SSL_cipher_suite1 [,SSL_cipher_suite2])

23.9.2.5 Step 2E: Set the Required TLS Version on the Client (Optional)

The SSL_VERSION parameter defines the version of TLS that must run on the systems with which the client communicates.

You must set the SSL_VERSION parameter in the sqlnet.ora file. You can require these systems to use any valid version.

The default setting for this parameter in sqlnet.ora is undetermined, which is set by selecting **Any** from the list in the **SSL** tab of the Network Security window. When **Any** is selected, TLS 1.0 is tried first, then TLS 3.0, and TLS 2.0 are tried in that order. Ensure that the client TLS version is compatible with the version the server uses.

- In the Require SSL Version list, select the SSL version that you want to configure.
 The default setting is Any.
- 2. From the File menu, select, Save Network Configuration.

The sqlnet.ora file is updated. If you selected **Any**, then it is updated with the following entry:

SSL VERSION=UNDETERMINED

23.9.2.6 Step 2F: Set TLS as an Authentication Service on the Client (Optional)

The SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file sets the TLS authentication service.

- About the SQLNET.AUTHENTICATION_SERVICES Parameter
 The SQLNET.AUTHENTICATION_SERVICES parameter enables TLS authentication in
 conjunction with another authentication method supported by Oracle Database.
- Setting the SQLNET.AUTHENTICATION_SERVICES Parameter You can set the SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file.

23.9.2.6.1 About the SQLNET.AUTHENTICATION_SERVICES Parameter

The SQLNET.AUTHENTICATION_SERVICES parameter enables TLS authentication in conjunction with another authentication method supported by Oracle Database.

For example, use this parameter if you want the server to authenticate itself to the client by using TLS and the client to authenticate itself to the server by using RADIUS.

To set the SQLNET.AUTHENTICATION_SERVICES parameter, you must edit the sqlnet.ora file, which is located in the same directory as the other network configuration files.

Depending on the platform, the sqlnet.ora file is in the following directory location:

- (UNIX) \$ORACLE HOME/network/admin
- (Windows) ORACLE BASE\ORACLE HOME\network\admin\

23.9.2.6.2 Setting the SQLNET.AUTHENTICATION SERVICES Parameter

You can set the SQLNET.AUTHENTICATION SERVICES parameter in the sqlnet.ora file.

• To set the client SQLNET.AUTHENTICATION_SERVICES parameter, add TCP/IP with TLS (TCPS) to this parameter in the sglnet.ora file by using a text editor.

For example, if you want to use TLS authentication in conjunction with RADIUS authentication, then set this parameter as follows:

```
SQLNET.AUTHENTICATION SERVICES = (TCPS, radius)
```

If you do not want to use TLS authentication in conjunction with another authentication method, then do not set this parameter.

23.9.2.7 Step 2G: Specify the Certificate to Use for Authentication on the Client (Optional)

If you have multiple certificates, then you can set the $SQLNET.SSL_EXTENDED_KEY_USAGE$ parameter in the sqlnet.ora file to specify the correct certificate.

- About the SQLNET.SSL_EXTENDED_KEY_USAGE Parameter
 The SQLNET.SSL_EXTENDED_KEY_USAGE parameter in the sqlnet.ora file specifies which certificate to use in authenticating to the database server
- Setting the SQLNET.SSL_EXTENDED_KEY_USAGE Parameter You can set the SQLNET.SSL EXTENDED KEY USAGE to set the client authentication.



23.9.2.7.1 About the SQLNET.SSL EXTENDED KEY USAGE Parameter

The SQLNET.SSL_EXTENDED_KEY_USAGE parameter in the sqlnet.ora file specifies which certificate to use in authenticating to the database server

You should set the SQLNET.SSL_EXTENDED_KEY_USAGE parameter if you have multiple certificates in the security module, but there is only one certificate with extended key usage field of client authentication, and this certificate is exactly the one you want to use to authenticate to the database.

For example, use this parameter if you have multiple certificates in a smart card, only one of which has an extended key usage field of client authentication, and you want to use this certificate $\mathbb C$ to authenticate to the database. By setting this parameter on a Windows client to client authentication, the MSCAPI certificate selection box will not appear, and the certificate $\mathbb C$ is automatically used for the Transport Layer Security authentication of the client to the server.

23.9.2.7.2 Setting the SQLNET.SSL_EXTENDED_KEY_USAGE Parameter

You can set the SQLNET.SSL EXTENDED KEY USAGE to set the client authentication.

• To set the client SQLNET.SSL_EXTENDED_KEY_USAGE parameter, edit the sqlnet.ora file to have the following line:

```
SQLNET.SSL EXTENDED KEY USAGE = "client authentication"
```

If you do not want to use the certificate filtering, then remove the SQLNET.SSL EXTENDED KEY USAGE parameter setting from the sqlnet.ora file.

23.9.2.8 Step 2H: Check That the Connections Are Using Transport Layer Security

You can query the V\$SESSION and V\$SESSION_CONNECT_INFO dynamic views to ensure that the client connections are using Transport Layer Security (TLS).

In SQL*Plus, perform the following query.

```
SELECT SYS CONTEXT ('USERENV', 'NETWORK PROTOCOL') FROM DUAL;
```

Output similar to the following should appear:

```
SYS_CONTEXT('USERENV','NETWORK_PROTOCOL')
----tcps
```

23.9.2.9 Step 2I: Restart the Database

To complete the configuration of Transport Layer Security on the client, you must restart the database.

• For example:

SHUTDOWN IMMEDIATE STARTUP



23.9.3 Step 3: Log in to the Database Instance

After you have completed the configuration, you are ready to log in to the database.

- Start SQL*Plus and then enter one of the following connection commands:
 - If you are using Transport Layer Security authentication for the client (SSL_CLIENT_AUTHENTICATION=true in the sqlnet.ora file):

```
CONNECT/@net_service_name
```

 If you are not using Transport Layer Security authentication (SSL CLIENT AUTHENTICATION=false in the sqlnet.ora file):

```
CONNECT username@net_service_name
Enter password: password
```

Related Topics

Certificate Validation with Certificate Revocation Lists
 Oracle provides tools that enable you to validate certificates using certificate revocation lists

23.10 Transport Layer Security Oracle Wallet Search Order

Oracle Database provides several routes for finding the wallet on a server in a Transport Layer Security (TLS) environment.

The Oracle Database server retrieves the wallet by searching in these locations, in the following order:

- WALLET ROOT in the init.ora file
- 2. WALLET LOCATION in the sqlnet.ora file
- Default wallet location:
 - Linux: /etc/ORACLE/WALLETS/user_name
 - Windows: C:\Users\user name\\ORACLE\WALLETS

The TNS listener retrieves the wallet location by searching in these locations, in the following order:

- WALLET_LOCATION in the listener.ora file
- 2. Default wallet location:
 - Linux: /etc/ORACLE/WALLETS/user name
 - Windows: C:\Users\user_name\\ORACLE\WALLETS

Oracle Database Client retrieves the wallet by searching in these locations, in the following order:

- Connect string
- 2. WALLET LOCATION in the sqlnet.ora file
- 3. Default wallet location:
 - Linux: /etc/ORACLE/WALLETS/user name
 - Windows: C:\Users\user name\\ORACLE\WALLETS

- **4.** System wallets located in the certificate store location. The default certificate store location depends on the platform, as follows:
 - Linux: RHEL/Oracle Linux: /etc/pki/tls/cert.pem
 - Microsoft Windows: Microsoft Certificate Store

23.11 Transport Layer Security Connections in an Oracle Real Application Clusters Environment

You can configure Transport Layer Security (TLS) connections in an Oracle Real Application Clusters (Oracle RAC) environment by using Oracle RAC tools and modifying Oracle Database configuration files.

- Step 1: Configure TCPS Protocol Endpoints
 - In Oracle Real Application Clusters (Oracle RAC), clients access one of three scan listeners and are then routed to database listeners. To support Transport Layer Security (TLS), all of these listeners must have TCPS protocol endpoints.
- Step 2: Ensure That the LOCAL_LISTENER Parameter Is Correctly Set on Each Node
 The Oracle Agent automatically sets the LOCAL_LISTENER parameter on each node, but you
 should double-check to ensure that it is correct.
- Step 3: Create Transport Layer Security Wallets and Certificates
 You must create Transport Layer Security (TLS) wallets and certificates for the cluster and also for clients that will connect to the cluster over TLS.
- Step 5: Define Wallet Locations in the listener.ora and sqlnet.ora Files
 To enable the database server and listeners to access the wallets, you must define the wallet locations in the listener.ora and sqlnet.ora files.
- Step 6: Restart the Database Instances and Listeners
 With the wallets in place and the *.ora files edited, you must restart the database server
 and listener processes so that they pick up the new settings.
- Step 7: Test the Cluster Node Configuration
 To test the cluster node configuration, you can create a connect descriptor for the node and then try to connect to this node.
- Step 8: Test the Remote Client Configuration
 After you have tested the wallet on the Oracle Real Applications (Oracle RAC) cluster
 nodes, you area ready to test the remote client configuration.

23.11.1 Step 1: Configure TCPS Protocol Endpoints

In Oracle Real Application Clusters (Oracle RAC), clients access one of three scan listeners and are then routed to database listeners. To support Transport Layer Security (TLS), all of these listeners must have TCPS protocol endpoints.

- Log in to the cluster that hosts the Oracle RAC database.
- 2. Check the listener resources to find if they support TCP endpoints.



For example:

```
$ srvctl config listener -h
```

Output similar to the following appears:

Name: LISTENER Subnet: 192.0.2.195 Type: type Owner: pfitch Home: Grid_home End points: TCP:1521

The following command displays information about the scan listener:

```
$ srvctl config scan listener -h
```

Output similar to the following appears:

```
SCAN Listener LISTENER_SCAN1 exists. Port: TCP:1529
Registration invited nodes:
Registration invited subnets:
SCAN Listener is enabled.
SCAN Listener is individually enabled on nodes:
SCAN Listener is individually disabled on nodes:
```

3. Add TCPS endpoints to the database listeners.

For example:

```
$ srvctl modify listener -endpoints "TCP:port_1/TCPS:port_2"
```

4. Check the listener configuration.

For example:

```
$ srvctl config listener

Name: LISTENER
Network: 1, Owner: oracle
Home: CRS_home
End points: TCP:port_1/TCPS:port_2

$ lsnrctl status

Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc) (KEY=LISTENER)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=IP_address) (PORT=port_2)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=IP_address) (PORT=port_1)))
```

5. Add TCPS endpoints to the scan listeners.

For example:

```
$ srvctl modify scan listener -endpoints "TCP:port 1/TCPS:port 2"
```

6. Check the scan listener configuration.

For example:

```
$ srvctl config scan_listener

SCAN Listener LISTENER_SCAN1 exists. Port: TCP:port_1/TCPS:port_2

SCAN Listener LISTENER_SCAN2 exists. Port: TCP:port_1/TCPS:port_2

SCAN Listener LISTENER_SCAN3 exists. Port: TCP:port_1/TCPS:port_2

$ lsnrctl status listener_scan3

Listening Endpoints Summary...

(DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=LISTENER_SCAN3)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=IP_address)(PORT=port_1)))

(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=IP_address)(PORT=port_2)))
```

23.11.2 Step 2: Ensure That the LOCAL_LISTENER Parameter Is Correctly Set on Each Node

The Oracle Agent automatically sets the <code>LOCAL_LISTENER</code> parameter on each node, but you should double-check to ensure that it is correct.

- 1. Log in any Oracle Real Application Clusters (Oracle RAC) node.
- In SQL*Plus, as a user with the SYSDBA administrative privilege, check the LOCAL_LISTENER parameter.

```
show parameter local listener;
```

Output similar to the following appears:

3. If the output is not what you want, then restart each Oracle RAC instance.

23.11.3 Step 3: Create Transport Layer Security Wallets and Certificates

You must create Transport Layer Security (TLS) wallets and certificates for the cluster and also for clients that will connect to the cluster over TLS.

- Oracle Real Application Clusters Components That Need Certificates
 Specific components in Oracle Real Application Clusters (Oracle RAC) need certificates
 when you configure Transport Layer Security (TLS) connections.
- Creating Transport Layer Security Wallets and Certificates
 To create the Transport Layer Security wallets and certificates, you first create the root CA certificate, followed by the cluster and client wallets.

23.11.3.1 Oracle Real Application Clusters Components That Need Certificates

Specific components in Oracle Real Application Clusters (Oracle RAC) need certificates when you configure Transport Layer Security (TLS) connections.

- Each cluster node (server) and listener must have a wallet with the user certificate and CA certificates.
- The client only needs CA certificates of the listeners and servers (either in wallet or system's certificate store) if one-way TLS is configured.
- The client needs a wallet with its user certificate and CA certificates of the listeners and servers if mTLS is configured.

23.11.3.2 Creating Transport Layer Security Wallets and Certificates

To create the Transport Layer Security wallets and certificates, you first create the root CA certificate, followed by the cluster and client wallets.

- 1. Create the root CA certificate.
 - a. Log in to any Oracle Real Application Clusters (Oracle RAC) cluster node.
 - **b.** Use the orapki utility to create the CA wallet in a directory for the CA.

```
$ orapki wallet create -wallet CA home wallet file directory
```

c. Create a self-signed root certificate for the CA wallet.

For example:

```
$ orapki wallet add -wallet CA_home_wallet_file_directory -self_signed -
dn "CN=test CA,O=test,C=c" -keysize 2048 -validity 3650 -sign_alg sha256
Enter wallet password: password
```

d. Extract the root CA certificate from the wallet.

This root certificate will be used as the trusted CA certificate in cluster and client wallets and can be distributed or published for users who are building PKCS#12 wallets. For example:

```
$ orapki wallet export -wallet CA_home_wallet_file_directory -dn
"CN=test CA,O=test,C=c" -cert testCAroot.cer
Enter wallet password: password
```

At this stage, the CA_home_wallet_file_directory directory will contain the new wallet (ewallet.p12) and certificate (testCAroot.cer).

To check the configuration:

```
$ orapki wallet display -wallet CA_home_wallet_file_directory -summary
```

Output similar to the following appears:

```
Requested Certificates:
User Certificates:
Subject: CN=test CA,O=test,C=c
Trusted Certificates:
Subject: CN=test CA,O=test,C=c
```

2. Create the cluster wallet.

Next, you are ready follow the remaining steps in this procedure to sign the user certificate requests and provide authorized digital user certificates for different entities and processes in your environments. Repeat this process for each entity in the test environment that participates in the public key infrastructure functionality. A valid wallet consists of a root CA certificate and the signed user certificate.

a. Create a wallet that is in a different location from the from the CA home directory.

For example:

```
$ orapki wallet create -wallet cluster_wallet_file_directory
Enter password: password
Enter password again: password
```

b. Create a user identity (user dn) and then a certificate request.

For example:

```
$ orapki wallet add -wallet cluster_wallet_file_directory -dn
"CN=testuser" -keysize 2048
Enter wallet password: password
$ orapki wallet export -wallet cluster_wallet_file_directory -dn
"CN=testuser" -request cluster_wallet_file_directory/testuser.req
Enter wallet password: password
```

At this stage, the <code>cluster_wallet_file_directory</code> directory will contain the SSO wallet (<code>cwallet.sso</code>), the wallet (<code>cwallet.p12</code>) and the certificate request (<code>testuser.req</code>). The certificate request can be signed by the CA generated above.

For example:

```
$ orapki cert create -wallet cluster_wallet_file_directory -request
cluster_wallet_file_directory/testuser.req -cert
user_wallet_file_directory/testuser.cer -validity 3650 -sign_alg sha256
Enter wallet password: password
```

The $cluster_wallet_file_directory$ directory now has the testuser.req certificate request file.

c. Import the root certificate (testCAroot.cer) and the signed user certificate (testuser.cer) into the user wallet.

For example:

```
$ orapki wallet add -wallet cluster_wallet_file_directory -trusted_cert
-cert CA_home_wallet_file_directory/testCAroot.cer -pwd
Enter wallet password: user password
```



```
$ orapki wallet add -wallet cluster_wallet_file_directory -user_cert -
cert cluster_wallet_file_directory/testuser.cer
Enter wallet password: user password
```

d. Check the finished cluster wallet.

For example:

```
$ orapki wallet display -wallet cluster_wallet_file_directory -summary
Requested Certificates:
User Certificates:
Subject: CN=testuser
Trusted Certificates:
Subject: CN=test CA,O=test,C=c
```

At this point, you are ready to copy the finished cluster wallet to each node of the cluster.

- 3. Create the client wallet.
 - a. Create a client wallet with the root certificate (testCAroot.cer).

To make a successful TLS connection, the client only requires the CA certificates of the server's certificate.

For example:

```
$ orapki wallet create -wallet client_wallet_file_directory -auto_login
$ orapki wallet add -wallet client_wallet_file_directory -trusted_cert -
cert CA_home_wallet_file_directory/testCAroot.cer
```

b. Check the finished client wallet.

For example:

```
$ orapki wallet display -wallet client_wallet_file_directory -summary
Requested Certificates:
User Certificates:
Trusted Certificates:
Subject: CN=test CA,O=test,C=c
```

23.11.4 Step 4: Create a Wallet in Each Node of the Oracle RAC Cluster

After you have created the cluster wallet, you can copy it to each node of the Oracle Real Applications (Oracle RAC) cluster.

Ensure that each node is accessible by both the Oracle Real Application Clusters (Oracle RAC) database server (process monitor) and by the scan and local listeners that normally run from the GI home.

- 1. Copy the PKCS#12 wallet (ewallet.p12) file that you created in the previous section to each node in the cluster.
- 2. In each node, create an auto-login wallet (cwallet.sso).

The <code>cwallet.sso</code> file is an obfuscated mirror copy of the <code>ewallet.p12</code> and is the file that the database server and its listeners accesses. If you create the <code>cwallet.sso</code> on the

Oracle RAC cluster, then you can copy it along with the <code>ewallet.p12</code> file to the wallet directory on each node. You can also create the <code>cwallet.sso</code> file on each node separately if <code>ewallet.p12</code> file is already in place. Run the following command in the same location as the <code>ewallet.p12</code> file:

```
$ orapki wallet create -wallet wallet_file_location -auto_login
Enter wallet password: ewallet password
```

Related Topics

Oracle Real Application Clusters Components That Need Certificates
 Specific components in Oracle Real Application Clusters (Oracle RAC) need certificates
 when you configure Transport Layer Security (TLS) connections.

23.11.5 Step 5: Define Wallet Locations in the listener.ora and sqlnet.ora Files

To enable the database server and listeners to access the wallets, you must define the wallet locations in the listener.ora and sqlnet.ora files.

1. Modify the listener.ora file in the Grid home of every node.

2. In the sqlnet.ora file in the Oracle Database home, and the Grid home, of each cluster node, add the following information:

23.11.6 Step 6: Restart the Database Instances and Listeners

With the wallets in place and the *.ora files edited, you must restart the database server and listener processes so that they pick up the new settings.

The restart process will also enable the Oracle Real Application Clusters (Oracle RAC) instances where you set the LOCAL LISTENER parameter earlier.

 In any cluster node, use the srvctl utility to restart the database server and listener processes.

For example:

```
$ srvctl stop listener
$ srvctl start listener
$ srvctl stop scan_listener
$ srvctl start scan_listener
$ srvctl stop database -d db_name
$ srvctl start database -d db name
```

23.11.7 Step 7: Test the Cluster Node Configuration

To test the cluster node configuration, you can create a connect descriptor for the node and then try to connect to this node.

1. In any cluster node, create a connect descriptor in the tnsnames.ora file that uses the scan listener TCPS endpoint.

For example, for a TCPS endpoint called dbssl:

```
DBSSL =
  (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCPS) (HOST = scan_name) (PORT = port_2))
      (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = service_name)
      )
    )
```

Use SQL*Plus to try to connect to this TCPS endpoint.

For example:

```
sqlplus user_name/@dbssl
Enter password: password
```

23.11.8 Step 8: Test the Remote Client Configuration

After you have tested the wallet on the Oracle Real Applications (Oracle RAC) cluster nodes, you area ready to test the remote client configuration.

1. In every remote client sqlnet.ora file on the cluster node, define a wallet directory.



2. Move the client wallet that you created earlier, when you created the SSL wallets and certificates, to the client wallet directory.

```
$ wallet create -wallet wallet_file_location -auto_login
Enter wallet password: password
```

The wallet file location should have an ewallet.p12 file and a cwallet.sso file.

In the tnsnames.ora file, create a connect descriptor that uses the scan listener TCPS endpoint.

For example:

```
DBSSL =
  (DESCRIPTION =
      (ADDRESS = (PROTOCOL = TCPS) (HOST = scan_name) (PORT = port_2))
      (CONNECT_DATA =
            (SERVER = DEDICATED)
            (SERVICE_NAME = service_name)
      )
    )
```

4. Use SQL*Plus to try to connect to this TCPS endpoint.

For example:

```
sqlplus user_name/@dbssl
Enter password: password
```

23.12 Configuring Transport Layer Security for Client Authentication and Encryption Using Microsoft Certificate Store

To perform this configuration with Microsoft Certificate Store (MCS), you use the orapki command-line tool to generate certificates and manipulate the Oracle wallets.

 About Configuring Transport Layer Security for Client Authentication and Encryption Using Microsoft Certificate Store

This type of configuration is the foundation of the Common Access Cards and PIV cards authentication.

- Step 1: Create and Configure the Server Wallet
 You must use orapki to create a server wallet and the server's self-signed certificate.
- Step 2: Create and Configure the Client Wallet
 You must use orapki to create a client wallet and a certificate request.
- Step 3: Create an External User in the Oracle Database
 You must create an external user to be used with the client and server connection.
- Step 4: Configure the Server listener.ora File
 Next, you must check and then restart the server listener.ora file.
- Step 5: Configure the Server sqlnet.ora File
 You must ensure that the sqlnet.ora file points to the server wallet that you created
 earlier.

- Step 6: Import the Client Wallet into the Microsoft Certificate Store
 You must use the Microsoft Management Console (MMC) to perform this import operation.
- Step 7: Configure the Client sqlnet.ora File
 You must configure the client sqlnet.ora file to use Microsoft Certificate Store for the client wallet.
- Step 8: Configure the Oracle Database
 In the Oracle database, configure the OS AUTHENT PRE and REMOTE OS AUTH parameters.
- Step 9: Test the Client and Server Connection
 After you complete the Microsoft Certificate Store configuration, you should test the and server connection.

23.12.1 About Configuring Transport Layer Security for Client Authentication and Encryption Using Microsoft Certificate Store

This type of configuration is the foundation of the Common Access Cards and PIV cards authentication.

As long as the software libraries that are delivered with the Common Access Cards and PIV cards are capable of transparently loading the necessary certificates into the Microsoft Certificate Store, then the Transport Layer Security (TLS) authentication that you configure will be transparently performed.

It is important to note that all the signing certificates of the user certificate that is loaded to the PIV card must be manually loaded into the server's wallet as part of the TLS configuration at the server level.

23.12.2 Step 1: Create and Configure the Server Wallet

You must use orapki to create a server wallet and the server's self-signed certificate.

- Log in to the Oracle Database server.
- Create a directory for the server wallet.

For example:

```
mkdir /home/oracle/wallet tls/server
```

Go to this directory.

```
cd /home/oracle/wallet tls/server
```

4. Create the server wallet.

```
orapki wallet create -wallet . -auto login -pwd password
```

Check the directory.

For example:

ls -la



Output similar to the following appears:

```
total 16
drwxr-xr-x. 2 oracle oinstall 4096 Oct 28 07:18 .
drwxr-xr-x. 6 oracle oinstall 4096 Oct 28 07:17 ..
-rw----- 1 oracle oinstall 120 Oct 28 07:18 cwallet.sso
-rw-rw-rw-. 1 oracle oinstall 0 Oct 28 07:18 cwallet.sso.lck
-rw----- 1 oracle oinstall 75 Oct 28 07:18 ewallet.p12
-rw-rw-rw-. 1 oracle oinstall 0 Oct 28 07:18 ewallet.p12.lck
```

6. Create the server's self-signed certificate.

```
orapki wallet add -wallet . -dn "cn=server" -self_signed -keysize 2048 - sign alg sha256 -validity 365 -pwd password
```

23.12.3 Step 2: Create and Configure the Client Wallet

You must use orapki to create a client wallet and a certificate request.

- 1. Log in to the Oracle Database client.
- 2. Create a directory for the client wallet.

For example:

```
mkdir /home/oracle/wallet tls/client
```

3. Go to this directory.

```
cd /home/oracle/wallet tls/client
```

4. Create the client wallet.

```
orapki wallet create -wallet . -auto login -pwd password
```

5. Create a request for a user certificate and export the request.

```
orapki wallet add -wallet . -dn "cn=client" -keysize 2048 -sign_alg sha256 -pwd password orapki wallet export -wallet . -dn "cn=client" -request req.txt -pwd password
```

6. Copy the certificate request from the client directory to the server directory.

For example:

```
cp req.txt ../server/
cd ../server/
```

7. Sign the certificate of the client and also export server's CA certificate.

For example:

```
orapki cert create -wallet . -request req.txt -cert sign.txt -validity 1000 -pwd password orapki wallet export -wallet . -dn "cn=server" -cert server.txt
```

```
cp server.txt ../client
cp sign.txt ../client
orapki wallet add -wallet . -trusted_cert -cert server.txt -pwd password
orapki wallet add -wallet . -user_cert -cert sign.txt -pwd password
cp sign.txt server.txt ../client/
cd ../client
```

23.12.4 Step 3: Create an External User in the Oracle Database

You must create an external user to be used with the client and server connection.

- 1. As a user who can create users and grant them privileges, log in to the PDB that will use this external user account.
- Create the external user.

For example:

```
CREATE USER tlsuser IDENTIFIED EXTERNALLY AS 'cn=client';
```

3. Grant this account the CONNECT privilege.

```
GRANT CONNECT TO tlsuser;
```

23.12.5 Step 4: Configure the Server listener.ora File

Next, you must check and then restart the server listener.ora file.

- Log in to the Oracle Database server.
- 2. Check the server listener.ora file to ensure that it is correctly configured.

For example:

```
cat /u01/app/oracle/product/release/dbhome 1/network/admin/listener.or
```

Output similar to the following appears:



3. Restart the listener and check if the database is registered to this listener.

```
su - oracle
./lsnrctl start
```

Output similar to the following appears:

```
Listener Parameter File /u01/app/oracle/product/release/dbhome_1/network/
admin/listener.ora
Listener Log File /u01/app/oracle/diag/tnslsnr/service/instance/alert/
log.xml
Listening Endpoints Summary...
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp) (HOST=domain.com) (PORT=1523)))
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcps) (HOST=domain.com) (PORT=1525)))
Services Summary...
Service "service" has 1 instance(s).
Instance "instance", status READY, has 1 handler(s) for this service...
Service "serviceDB" has 1 instance(s).
Instance "instance", status READY, has 1 handler(s) for this service...
The command completed successful
```

23.12.6 Step 5: Configure the Server sqlnet.ora File

You must ensure that the sqlnet.ora file points to the server wallet that you created earlier.

- 1. Log in to the Oracle Database server.
- 2. Check the sqlnet.ora file to ensure that it points to the server wallet.

For example:

```
NAMES.DIRECTORY_PATH=(TNSNAMES)
SQLNET.AUTHENTICATION_SERVICES=(BEQ,TCPS)
SSL_CLIENT_AUTHENTICATION = TRUE
WALLET_LOCATION =
  (SOURCE =
   (METHOD = FILE)
    (METHOD_DATA =
        (DIRECTORY = /home/oracle/wallet_tls/server)
    )
)
```

23.12.7 Step 6: Import the Client Wallet into the Microsoft Certificate Store

You must use the Microsoft Management Console (MMC) to perform this import operation.

- Start the MMC (mmc.exe).
- 2. Select File, then add/remove snap-in.
- Select Certificates, then Add.
- 4. Select My user account, then Finish, and then OK.
- Go to the Console Root, then Certificates Current User, then Personal, then Certificates.

- Right-click All Tasks, then select Import, then Next, then Browse.
- Select the certificate file that contains the certificate needed for the connection (for example, ewallet.p12).
- 8. Select Open, then Next.
- 9. Enter the wallet password.
- 10. Select the Mark this as exportable checkbox.
- 11. Select the Include all extended properties checkbox.
- 12. Select Place all certificates in the following store: Personal.
- 13. Select Next, then Finish.
- 14. Ensure that the client's certificate was added to the MY store, by going to Console Root, and then selecting Certificates Current User, then Personal, then Certificates.
- 15. Ensure that the CA certificates were added to the ROOT store by going to Console Root, and then selecting Certificates Current User, then Trusted Root Certification Authorities, then Certificates.

23.12.8 Step 7: Configure the Client sqlnet.ora File

You must configure the client sqlnet.ora file to use Microsoft Certificate Store for the client wallet.

- 1. Log in to the Oracle Database client.
- 2. Check the client side sqlnet.ora file.

For example:

```
WALLET LOCATION = (SOURCE = (METHOD=MCS))
```

23.12.9 Step 8: Configure the Oracle Database

In the Oracle database, configure the OS AUTHENT PRE and REMOTE OS AUTH parameters.

- Log in SQL*Plus on the Oracle Database server as a user who has the ALTER SYSTEM system privilege.
- Set the OS AUTHENT PRE and REMOTE OS AUTH parameters.

```
ALTER SYSTEM SET REMOTE_OS_AUTHENT=FALSE SCOPE=SPFILE;
ALTER SYSTEM SET OS_AUTHENT_PREFIX='' SCOPE=SPFILE;
```

3. Restart the database instance.

23.12.10 Step 9: Test the Client and Server Connection

After you complete the Microsoft Certificate Store configuration, you should test the and server connection.

1. To verify that the MCS is used for the TLS connection, enable the client trace by adding the following lines in the client's sqlnet.ora file.

```
trace_level_client=16
trace_directory_client=trace_directory
DIAG_ADR_ENABLED=OFF
```

Make a connection to the server using SQL*Plus and then ensure that the certificates are loaded successfully from MCS.

```
nztwOpenWallet: [enter]
nztwOpenWallet: WRL mcs:, type = 24
nztwOpenWallet: Loading the EXTKS provider for MCS type wallet
nztwOpenWallet: [exit] OK
```

23.13 Troubleshooting the Transport Layer Security Configuration

Common errors may occur while you use the Oracle Database SSL adapter.

It may be necessary to enable Oracle Net tracing to determine the cause of an error. For information about setting tracing parameters to enable Oracle Net tracing, refer to *Oracle Database Net Services Administrator's Guide*.

ORA-28759: Failure to Open File

Cause: The system could not open the specified file. Typically, this error occurs because the wallet cannot be found.

Action: Check the following:

- Ensure that the correct wallet location is specified in the sqlnet.ora file. This should be the same directory location where you saved the wallet.
- Enable Oracle Net tracing to determine the name of the file that cannot be opened and the reason.
- Ensure that auto-login was enabled when you saved the wallet. See *Oracle Database Enterprise User Security Administrator's Guide*.

ORA-28786: Decryption of Encrypted Private Key Failure

Cause: An incorrect password was used to decrypt an encrypted private key. Frequently, this happens because an auto-login wallet is not being used.

Action: Use Oracle Wallet Manager to turn the auto-login feature on for the wallet. Then save the wallet again. See *Oracle Database Enterprise User Security Administrator's Guide* If the auto-login feature is not being used, then enter the correct password.

ORA-28858: SSL Protocol Error

Cause: This is a generic error that can occur during TLS handshake negotiation between two processes.

Action: Enable Oracle Net tracing and attempt the connection again to produce trace output. Then contact Oracle customer support with the trace output.

ORA-28859 SSL Negotiation Failure

Cause: An error occurred during the negotiation between two processes as part of the TLS protocol. This error can occur when two sides of the connection do not support a common cipher suite.

Action: Check the following:

- Use Oracle Net Manager to ensure that the TLS versions on both the client and the server match, or are compatible. For example, if the server accepts only TLS 3.0 and the client accepts only TLS 1.1, then the TLS connection will fail.
- Use Oracle Net Manager to check what cipher suites are configured on the client and the server, and ensure that compatible cipher suites are set on both.

If the error still persists, then enable Oracle Net tracing and attempt the connection again. Contact Oracle customer support with the trace output.



Step 2D: Set the Client Transport Layer Security Cipher Suites (Optional) for details about setting compatible cipher suites on the client and the server

Note:

If you do not configure any cipher suites, then all available cipher suites are enabled.

ORA-28862: SSL Connection Failed

Cause: This error occurred because the peer closed the connection.

Action: Check the following:

- Ensure that the correct wallet location is specified in the sqlnet.ora file so the system can find the wallet.
- Use Oracle Net Manager to ensure that cipher suites are set correctly in the sqlnet.ora
 file. Sometimes this error occurs because the sqlnet.ora has been manually edited and
 the cipher suite names are misspelled. Ensure that case sensitive string matching is used
 with cipher suite names.
- Use Oracle Net Manager to ensure that the TLS versions on both the client and the server
 match or are compatible. Sometimes this error occurs because the TLS version specified
 on the server and client do not match. For example, if the server accepts only TLS 3.0 and
 the client accepts only TLS 1.0, then the TLS connection will fail.
- For more diagnostic information, enable Oracle Net tracing on the peer.

ORA-28865: SSL Connection Closed

Cause: The TLS connection closed because of an error in the underlying transport layer, or because the peer process guit unexpectedly.

Action: Check the following:



- Use Oracle Net Manager to ensure that the TLS versions on both the client and the server match, or are compatible. Sometimes this error occurs because the TLS version specified on the server and client do not match. For example, if the server accepts only TLS 3.0 and the client accepts only TLS 1.0, then the TLS connection will fail.
- If you are using a Diffie-Hellman anonymous cipher suite and the SSL_CLIENT_AUTHENTICATION parameter is set to true in the server's listener.ora file, then the client does not pass its certificate to the server. When the server does not receive the client's certificate, it (the server) cannot authenticate the client so the connection is closed. To resolve this use another cipher suite, or set this listener.ora parameter to false.
- Enable Oracle Net tracing and check the trace output for network errors.
- For details, refer to Actions listed for ORA-28862: SSL Connection Failed

ORA-28868: Peer Certificate Chain Check Failed

Cause: When the peer presented the certificate chain, it was checked and that check failed. This failure can be caused by a number of problems, including:

- One of the certificates in the chain has expired.
- A certificate authority for one of the certificates in the chain is not recognized as a trust point.
- The signature in one of the certificates cannot be verified.

Action: See *Oracle Database Enterprise User Security Administrator's Guide* to use Oracle Wallet Manager to open your wallet and check the following:

- Ensure that all of the certificates installed in your wallet are current (not expired).
- Ensure that a certificate authority's certificate from your peer's certificate chain is added as
 a trusted certificate in your wallet. See Oracle Database Enterprise User Security
 Administrator's Guide to use Oracle Wallet Manager to import a trusted certificate.

ORA-28885: No certificate with the required key usage found.

Cause: Your certificate was not created with the appropriate X.509 version 3 key usage extension.

Action: Use Oracle Wallet Manager to check the certificate's key usage. See *Oracle Database Enterprise User Security Administrator's Guide* for information about key usage values.

ORA-29024: Certificate Validation Failure

Cause: The certificate sent by the other side could not be validated. This may occur if the certificate has expired, has been revoked, or is invalid for any other reason.

Action: Check the following:

- Check the certificate to determine whether it is valid. If necessary, get a new certificate, inform the sender that her certificate has failed, or resend.
- Check to ensure that the server's wallet has the appropriate trust points to validate the
 client's certificate. If it does not, then use Oracle Wallet Manager to import the appropriate
 trust point into the wallet. See Oracle Database Enterprise User Security Administrator's
 Guide for details about importing a trusted certificate.
- Ensure that the certificate has not been revoked and that certificate revocation list (CRL) checking is turned on. For details, refer to Configuring Certificate Validation with Certificate Revocation Lists



ORA-29223: Cannot Create Certificate Chain

Cause: A certificate chain cannot be created with the existing trust points for the certificate being installed. Typically, this error is returned when the peer does not give the complete chain and you do not have the appropriate trust points to complete it.

Action: Use Oracle Wallet Manager to install the trust points that are required to complete the chain. See *Oracle Database Enterprise User Security Administrator's Guide* for details about importing a trusted certificate.

23.14 Certificate Validation with Certificate Revocation Lists

Oracle provides tools that enable you to validate certificates using certificate revocation lists.

- About Certificate Validation with Certificate Revocation Lists
 The process of determining whether a given certificate can be used in a given context is referred to as certificate validation.
- What CRLs Should You Use?
 You should have CRLs for all of the trust points that you honor.
- How CRL Checking Works
 Oracle Database checks the certificate revocation status against CRLs.
- Configuring Certificate Validation with Certificate Revocation Lists
 You can edit the sqlnet.ora file to configure certificate validation with certificate revocation lists
- Certificate Revocation List Management
 Certificate revocation list management entails ensuring that the CRLs are the correct format before you enable certificate revocation checking.
- Troubleshooting CRL Certificate Validation
 To determine whether certificates are being validated against CRLs, you can enable Oracle Net tracing.
- Oracle Net Tracing File Error Messages Associated with Certificate Validation
 Oracle generates trace messages that are relevant to certificate validation.

23.14.1 About Certificate Validation with Certificate Revocation Lists

The process of determining whether a given certificate can be used in a given context is referred to as certificate validation.

Certificate validation includes determining that the following takes place:

- A trusted certificate authority (CA) has digitally signed the certificate
- The certificate's digital signature corresponds to the independently-calculated hash value of the certificate itself and the certificate signer's (CA's) public key
- The certificate has not expired
- The certificate has not been revoked

The Transport Layer Security network layer automatically performs the first three validation checks, but you must configure certificate revocation list (CRL) checking to ensure that certificates have not been revoked. CRLs are signed data structures that contain a list of revoked certificates. They are usually issued and signed by the same entity who issued the original certificate.



23.14.2 What CRLs Should You Use?

You should have CRLs for all of the trust points that you honor.

The trust points are the trusted certificates from a third party identity that is qualified with a level of trust.

Typically, the certificate authorities you trust are called trust points.

23.14.3 How CRL Checking Works

Oracle Database checks the certificate revocation status against CRLs.

These CRLs are located in file system directories, Oracle Internet Directory, or downloaded from the location specified in the CRL Distribution Point (CRL DP) extension on the certificate.

Typically, CRL definitions are valid for a few days. If you store your CRLs on the local file system or in the directory, then you must update them regularly. If you use a CRL Distribution Point (CRL DP), then CRLs are downloaded each time a certificate is used, so there is no need to regularly refresh the CRLs.

The server searches for CRLs in the following locations in the order listed. When the system finds a CRL that matches the certificate CA's DN, it stops searching.

1. Local file system

The system checks the sqlnet.ora file for the SSL_CRL_FILE parameter first, followed by the SSL_CRL_PATH parameter. If these two parameters are not specified, then the system checks the wallet location for any CRLs.

Note: if you store CRLs on your local file system, then you must use the <code>orapki</code> utility to periodically update them (for example, renaming CRLs with a hash value for certificate validation).

2. Oracle Internet Directory

If the server cannot locate the CRL on the local file system and directory connection information has been configured in an ldap.ora file, then the server searches in the directory. It searches the CRL subtree by using the CA's distinguished name (DN) and the DN of the CRL subtree.

The server must have a properly configured <code>ldap.ora</code> file to search for CRLs in the directory. It cannot use the Domain Name System (DNS) discovery feature of Oracle Internet Directory. Also note that if you store CRLs in the directory, then you must use the <code>orapki</code> utility to periodically update them.

3. CRL DP

If the CA specifies a location in the CRL DP X.509, version 3, certificate extension when the certificate is issued, then the appropriate CRL that contains revocation information for that certificate is downloaded. Currently, Oracle Database supports downloading CRLs over LDAP.

Note the following:

- For performance reasons, only user certificates are checked.
- Oracle recommends that you store CRLs in the directory rather than the local file system.



Related Topics

- Uploading CRLs to Oracle Internet Directory
 Publishing CRLs in the directory enables CRL validation throughout your enterprise,
 eliminating the need for individual applications to configure their own CRLs.
- Renaming CRLs with a Hash Value for Certificate Validation
 When the system validates a certificate, it must locate the CRL issued by the CA who created the certificate.

23.14.4 Configuring Certificate Validation with Certificate Revocation Lists

You can edit the sqlnet.ora file to configure certificate validation with certificate revocation lists.

- About Configuring Certificate Validation with Certificate Revocation Lists

 The SSL_CERT_REVOCATION parameter must be set to REQUIRED or REQUESTED in the sqlnet.ora file to enable certificate revocation status checking.
- Enabling Certificate Revocation Status Checking for the Client or Server
 You can enable certificate the revocation status checking for a client or a server.
- Disabling Certificate Revocation Status Checking You can disable certificate revocation status checking.

23.14.4.1 About Configuring Certificate Validation with Certificate Revocation Lists

The SSL_CERT_REVOCATION parameter must be set to REQUIRED or REQUESTED in the sqlnet.ora file to enable certificate revocation status checking.

The SSL_CERT_REVOCATION parameter must be set to REQUIRED or REQUESTED in the sqlnet.ora file to enable certificate revocation status checking.

By default this parameter is set to NONE indicating that certificate revocation status checking is turned off.



If you want to store CRLs on your local file system or in Oracle Internet Directory, then you must use the command line utility, orapki, to rename CRLs in your file system or upload them to the directory.

Related Topics

Certificate Revocation List Management
 Certificate revocation list management entails ensuring that the CRLs are the correct format before you enable certificate revocation checking.

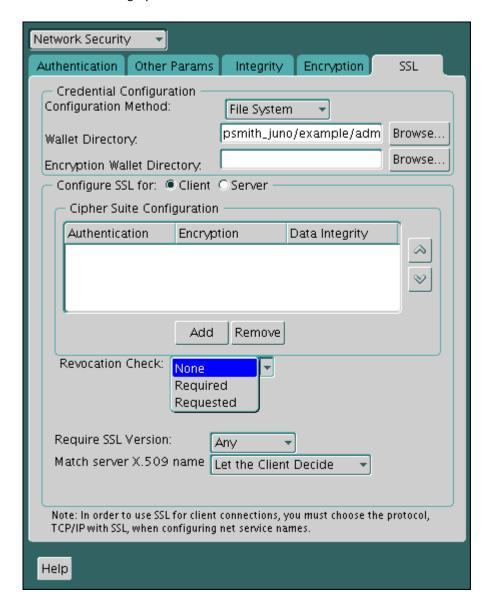
23.14.4.2 Enabling Certificate Revocation Status Checking for the Client or Server

You can enable certificate the revocation status checking for a client or a server.

- 1. Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr



- (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.The Network Security tabbed window appears.
- 4. Select the **SSL** tab.
- 5. Select one of the following options from the Revocation Check list:



- Required: Requires certificate revocation status checking. The TLS connection is
 rejected if a certificate is revoked or no CRL is found. TLS connections are accepted
 only if it can be verified that the certificate has not been revoked.
- Requested: Performs certificate revocation status checking if a CRL is available. The
 TLS connection is rejected if a certificate is revoked. TLS connections are accepted if
 no CRL is found or if the certificate has not been revoked.

For performance reasons, only user certificates are checked for revocation.

- 6. (Optional) If CRLs are stored on your local file system, then set one or both of the following fields that specify where they are stored. These fields are available only when Revocation Check is set to Required or REQUESTED.
 - Certificate Revocation Lists Path: Enter the path to the directory where CRLs are stored or click Browse to find it by searching the file system. Specifying this path sets the SSL_CRL_PATH parameter in the sqlnet.ora file. If a path is not specified for this parameter, then the default is the wallet directory. Both DER-encoded (binary format) and PEM-encoded (BASE64) CRLs are supported.
 - Certificate Revocation Lists File: Enter the path to a comprehensive CRL file (where PEM-encoded (BASE64) CRLs are concatenated in order of preference in one file) or click Browse to find it by searching the file system. Specifying this file sets the SSL_CRL_FILE parameter in the sqlnet.ora file. If this parameter is set, then the file must be present in the specified location, or else the application will error out during startup.
 - If you want to store CRLs in a local file system directory by setting the **Certificate Revocation Lists Path**, then you must use the orapki utility to rename them so the system can locate them.
- 7. (Optional) If CRLs are fetched from Oracle Internet Directory, then directory server and port information must be specified in an ldap.ora file.
 - When configuring your ldap.ora file, you should specify only a non-TLS port for the directory. CRL download is done as part of the TLS protocol, and making a TLS connection within a TLS connection is not supported.
 - Oracle Database CRL functionality will not work if the Oracle Internet Directory non-TLS port is disabled.
- 8. Select File, Save Network Configuration. The sqlnet.ora file is updated.

Related Topics

Renaming CRLs with a Hash Value for Certificate Validation
 When the system validates a certificate, it must locate the CRL issued by the CA who created the certificate.

23.14.4.3 Disabling Certificate Revocation Status Checking

You can disable certificate revocation status checking.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- 3. From the Naming list, select Network Security.
 - The Network Security tabbed window appears.
- Select the SSL tab.
- 5. Select NONE from the Revocation Check list.
- 6. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entry:



SSL CERT REVOCATION=NONE

Related Topics

Troubleshooting CRL Certificate Validation
 To determine whether certificates are being validated against CRLs, you can enable Oracle Net tracing.

23.14.5 Certificate Revocation List Management

Certificate revocation list management entails ensuring that the CRLs are the correct format before you enable certificate revocation checking.

- About Certificate Revocation List Management
 Oracle Database provides a command-line utility, orapki, that you can use to manage
 certificates.
- Displaying orapki Help for Commands That Manage CRLs
 You can display all the orapki commands that are available for managing CRLs.
- Renaming CRLs with a Hash Value for Certificate Validation
 When the system validates a certificate, it must locate the CRL issued by the CA who created the certificate.
- Uploading CRLs to Oracle Internet Directory
 Publishing CRLs in the directory enables CRL validation throughout your enterprise, eliminating the need for individual applications to configure their own CRLs.
- Listing CRLs Stored in Oracle Internet Directory
 You can display a list of all CRLs stored in the directory with orapki, which is useful for browsing to locate a particular CRL to view or download to your local computer.
- Viewing CRLs in Oracle Internet Directory
 Oracle Internet Directory CRLS are available in a summarized format; you also can
 request a listing of revoked certificates for a CRL.
- Deleting CRLs from Oracle Internet Directory
 The user who deletes CRLs from the directory by using orapki must be a member of the directory group CRLAdmins.

23.14.5.1 About Certificate Revocation List Management

Oracle Database provides a command-line utility, orapki, that you can use to manage certificates.

Before you can enable certificate revocation status checking, you must ensure that the CRLs you receive from the CAs you use are in a form (renamed with a hash value) or in a location (uploaded to the directory) where your computer can use them.

You can also use LDAP command-line tools to manage CRLs in Oracle Internet Directory.



CRLs must be updated at regular intervals (before they expire) for successful validation. You can automate this task by using orapki commands in a script



23.14.5.2 Displaying orapki Help for Commands That Manage CRLs

You can display all the orapki commands that are available for managing CRLs.

 To display all the orapki available CRL management commands and their options, enter the following at the command line:

```
orapki crl help
```



Using the -summary, -complete, or -wallet command options is always optional. A command will still run if these command options are not specified.

23.14.5.3 Renaming CRLs with a Hash Value for Certificate Validation

When the system validates a certificate, it must locate the CRL issued by the CA who created the certificate.

The system locates the appropriate CRL by matching the issuer name in the certificate with the issuer name in the CRL.

When you specify a CRL storage location for the **Certificate Revocation Lists Path** field in Oracle Net Manager, which sets the SSL_CRL_PATH parameter in the sqlnet.ora file, use the orapki utility to rename CRLs with a hash value that represents the issuer's name. Creating the hash value enables the server to load the CRLs.

On UNIX operating systems, <code>orapki</code> creates a symbolic link to the CRL. On Windows operating systems, it creates a copy of the CRL file. In either case, the symbolic link or the copy created by <code>orapki</code> are named with a hash value of the issuer's name. Then when the system validates a certificate, the same hash function is used to calculate the link (or copy) name so the appropriate CRL can be loaded.

- Depending on the operating system, enter one of the following commands to rename CRLs stored in the file system:
 - To rename CRLs stored in UNIX file systems:

```
orapki crl hash -crl crl_filename [-wallet wallet_location] -symlink
crl directory [-summary]
```

To rename CRLs stored in Windows file systems:

```
orapki crl hash -crl crl_filename [-wallet wallet_location] -copy crl_directory
[-summary]
```

In this specification, <code>crl_filename</code> is the name of the CRL file, <code>wallet_location</code> is the location of a wallet that contains the certificate of the CA that issued the CRL, and <code>crl_directory</code> is the directory where the CRL is located.

Using -wallet and -summary are optional. Specifying -wallet causes the tool to verify the validity of the CRL against the CA's certificate prior to renaming the CRL. Specifying the -summary option causes the tool to display the CRL issuer's name.



23.14.5.4 Uploading CRLs to Oracle Internet Directory

Publishing CRLs in the directory enables CRL validation throughout your enterprise, eliminating the need for individual applications to configure their own CRLs.

All applications can use the CRLs stored in the directory where they can be centrally managed, greatly reducing the administrative overhead of CRL management and use. The user who uploads CRLs to the directory by using orapki must be a member of the directory group CRLAdmins (cn=CRLAdmins, cn=groups, %s_OracleContextDN%). This is a privileged operation because these CRLs are accessible to the entire enterprise. Contact your directory administrator to get added to this administrative directory group.

To upload CRLs to the directory, enter the following at the command line:

```
orapki crl upload -crl crl_location -ldap hostname:ssl_port -user username [-wallet wallet location] [-summary]
```

In this specification, <code>crl_location</code> is the file name or URL where the CRL is located, <code>hostname</code> and <code>ssl_port</code> (TLS port with no authentication) are for the system on which your directory is installed, <code>username</code> is the directory user who has permission to add CRLs to the CRL subtree, and <code>wallet_location</code> is the location of a wallet that contains the certificate of the CA that issued the CRL.

Using -wallet and -summary are optional. Specifying -wallet causes the tool to verify the validity of the CRL against the CA's certificate prior to uploading it to the directory. Specifying the -summary option causes the tool to print the CRL issuer's name and the LDAP entry where the CRL is stored in the directory.

The following example illustrates uploading a CRL with the orapki utility:

orapki crl upload -crl /home/user1/wallet/crldir/crl.txt -ldap host1.example.com:3533 - user cn=orcladmin

Note:

- The orapki utility will prompt you for the directory password when you perform this operation.
- Ensure that you specify the directory SSL port on which the Diffie-Hellman-based TLS server is running. This is the TLS port that does not perform authentication. Neither the server authentication nor the mutual authentication TLS ports are supported by the orapki utility.

23.14.5.5 Listing CRLs Stored in Oracle Internet Directory

You can display a list of all CRLs stored in the directory with orapki, which is useful for browsing to locate a particular CRL to view or download to your local computer.

This command displays the CA who issued the CRL (Issuer) and its location (DN) in the CRL subtree of your directory.

To list CRLs in Oracle Internet Directory, enter the following at the command line:

```
orapki crl list -ldap hostname:ssl port
```



where the <code>hostname</code> and <code>ssl_port</code> are for the system on which your directory is installed. Note that this is the directory SSL port with no authentication as described in the preceding section.

23.14.5.6 Viewing CRLs in Oracle Internet Directory

Oracle Internet Directory CRLS are available in a summarized format; you also can request a listing of revoked certificates for a CRL.

You can view CRLs stored in Oracle Internet Directory in a summarized format or you can request a complete listing of revoked certificates for a CRL. A summary listing provides the CRL issuer's name and its validity period. A complete listing provides a list of all revoked certificates contained in the CRL.

 To view a summary listing of a CRL in Oracle Internet Directory, enter the following at the command line:

```
orapki crl display -crl crl location [-wallet wallet location] -summary
```

In this specification, <code>crl_location</code> is the location of the CRL in the directory. It is convenient to paste the CRL location from the list that displays when you use the <code>orapkicrl list</code> command.

To view a list of all revoked certificates contained in a specified CRL, which is stored in Oracle Internet Directory, you can enter the following at the command line:

```
orapki crl display -crl crl location [-wallet wallet location] -complete
```

For example, the following orapki command:

```
orapki crl display -crl T_WORK/pki/wlt_crl/nzcrl.txt -wallet T_WORK/pki/wlt_crl -complete
```

produces the following output, which lists the CRL issuer's DN, its publication date, date of its next update, and the revoked certificates it contains:

```
issuer = CN=root,C=us, thisUpdate = Sun Nov 16 10:56:58 PST 2003, nextUpdate = Mon
Sep 30 11:56:58 PDT 2013, revokedCertificates = { (serialNo =
153328337133459399575438325845117876415, revocationDate - Sun Nov 16 10:56:58 PST
2003) }
CRL is valid
```

Using the -wallet option causes the orapki crl display command to validate the CRL against the CA's certificate.

Depending on the size of your CRL, choosing the -complete option may take a long time to display.

You can also use Oracle Directory Manager, a graphical user interface tool that is provided with Oracle Internet Directory, to view CRLs in the directory. CRLs are stored in the following directory location:

cn=CRLValidation,cn=Validation,cn=PKI,cn=Products,cn=OracleContext

Related Topics

Listing CRLs Stored in Oracle Internet Directory

You can display a list of all CRLs stored in the directory with <code>orapki</code>, which is useful for browsing to locate a particular CRL to view or download to your local computer.

23.14.5.7 Deleting CRLs from Oracle Internet Directory

The user who deletes CRLs from the directory by using orapki must be a member of the directory group CRLAdmins.

To delete CRLs from the directory, enter the following at the command line:

```
orapki crl delete -issuer issuer name -ldap host:ssl port -user username [-summary]
```

In this specification, <code>issuer_name</code> is the name of the CA who issued the CRL, the <code>hostname</code> and <code>ssl_port</code> are for the system on which your directory is installed, and <code>username</code> is the directory user who has permission to delete CRLs from the CRL subtree. Ensure that this must be a directory SSL port with no authentication.

Using the -summary option causes the tool to print the CRL LDAP entry that was deleted.

For example, the following orapki command:

```
orapki crl delete -issuer "CN=root,C=us" -ldap machine1:3500 -user cn=orcladmin -summary
```

produces the following output, which lists the location of the deleted CRL in the directory:

```
Deleted CRL at cn=root cd45860c.rN,cn=CRLValidation,cn=Validation,cn=PKI,cn=Products,cn=OracleContext
```

Related Topics

Uploading CRLs to Oracle Internet Directory
 Publishing CRLs in the directory enables CRL validation throughout your enterprise, eliminating the need for individual applications to configure their own CRLs.

23.14.6 Troubleshooting CRL Certificate Validation

To determine whether certificates are being validated against CRLs, you can enable Oracle Net tracing.

When a revoked certificate is validated by using CRLs, then you will see the following entries in the Oracle Net tracing file without error messages logged between entry and exit:

```
nzcrlVCS_VerifyCRLSignature: entry
nzcrlVCS_VerifyCRLSignature: exit

nzcrlVCD_VerifyCRLDate: entry
nzcrlVCD_VerifyCRLDate: exit

nzcrlCCS_CheckCertStatus: entry
nzcrlCCS_CheckCertStatus: Certificate is listed in CRL
nzcrlCCS CheckCertStatus: exit
```

Note:

Note that when certificate validation fails, the peer in the SSL handshake sees an ORA-29024: Certificate Validation Failure.

Related Topics

- Oracle Net Tracing File Error Messages Associated with Certificate Validation
 Oracle generates trace messages that are relevant to certificate validation.
- Oracle Database Net Services Administrator's Guide

23.14.7 Oracle Net Tracing File Error Messages Associated with Certificate Validation

Oracle generates trace messages that are relevant to certificate validation.

These trace messages may be logged between the <code>entry</code> and <code>exit</code> entries in the Oracle Net tracing file. Oracle SSL looks for CRLs in multiple locations, so there may be multiple errors in the trace.

You can check the following list of possible error messages for information about how to resolve them.

CRL signature verification failed with RSA status

Cause: The CRL signature cannot be verified.

Action: Ensure that the downloaded CRL is issued by the peer's CA and that the CRL was not corrupted when it was downloaded. Note that the <code>orapki</code> utility verifies the CRL before renaming it with a hash value or before uploading it to the directory. See Certificate Revocation List Management for information about using <code>orapki</code> for CRL management.

CRL date verification failed with RSA status

Cause: The current time is later than the time listed in the next update field. You should not see this error if CRL DP is used. The systems searches for the CRL in the following order:

- File system
- Oracle Internet Directory
- 3. CRL DP

The first CRL found in this search may not be the latest.

Action: Update the CRL with the most recent copy.

CRL could not be found

Cause: The CRL could not be found at the configured locations. This will return error ORA-29024 if the configuration specifies that certificate validation is require.

Action: Ensure that the CRL locations specified in the configuration are correct by performing the following steps:

- Use Oracle Net Manager to check if the correct CRL location is configured. Refer to Configuring Certificate Validation with Certificate Revocation Lists
- 2. If necessary, use the orapki utility to configure CRLs for system use as follows:
 - For CRLs stored on your local file system, refer to Renaming CRLs with a Hash Value for Certificate Validation
 - CRLs stored in the directory, refer to Uploading CRLs to Oracle Internet Directory



Oracle Internet Directory host name or port number not set

Cause: Oracle Internet Directory connection information is not set. Note that this is not a fatal error. The search continues with CRL DP.

Action: If you want to store the CRLs in Oracle Internet Directory, then use Oracle Net Configuration Assistant to create and configure an ldap.ora file for your Oracle home.

Fetch CRL from CRL DP: No CRLs found

Cause: The CRL could not be fetched by using the CRL Distribution Point (CRL DP). This happens if the certificate does not have a location specified in its CRL DP extension, or if the URL specified in the CRL DP extension is incorrect.

Action: Ensure that your certificate authority publishes the CRL to the URL that is specified in the certificate's CRL DP extension.

Manually download the CRL. Then depending on whether you want to store it on your local file system or in Oracle Internet Directory, perform the following steps:

If you want to store the CRL on your local file system:

- Use Oracle Net Manager to specify the path to the CRL directory or file. Refer to Configuring Certificate Validation with Certificate Revocation Lists
- 2. Use the orapki utility to configure the CRL for system use. Refer to Renaming CRLs with a Hash Value for Certificate Validation

If you want to store the CRL in Oracle Internet Directory:

- 1. Use Oracle Net Configuration Assistant to create and configure an ldap.ora file with directory connection information.
- Use the orapki utility to upload the CRL to the directory. Refer to Uploading CRLs to Oracle Internet Directory

23.15 Configuring Your System to Use Hardware Security Modules

Oracle Database supports hardware security modules that use APIs that conform to the RSA Security, Inc., PKCS #11 specification.

Typically, these hardware devices are used to securely store and manage private keys in tokens or smart cards, or to accelerate cryptographic processing.

- General Guidelines for Using Hardware Security Modules for TLS
 Oracle provides a set of guidelines to follow if you are using a hardware security module with Oracle Database.
- Configuring Your System to Use nCipher Hardware Security Modules
 You can configure your system to use nCipher hardware security modules for
 cryptographic processing.
- Configuring Your System to Use SafeNET Hardware Security Modules
 You can configure your system to use SafeNET hardware security modules for
 cryptographic processing.
- Troubleshooting Using Hardware Security Modules
 Oracle provides troubleshooting advice for hardware security modules.



23.15.1 General Guidelines for Using Hardware Security Modules for TLS

Oracle provides a set of guidelines to follow if you are using a hardware security module with Oracle Database.

- Contact your hardware device vendor to obtain the necessary hardware, software, and PKCS #11 libraries.
- 2. Install the hardware, software, and libraries where appropriate for the hardware security module you are using.
- 3. Test your hardware security module installation to ensure that it is operating correctly. Refer to your device documentation for instructions.
- 4. Create a wallet of the type PKCS11 by using Oracle Wallet Manager and specify the absolute path to the PKCS #11 library (including the library name) if you wish to store the private key in the token. Oracle PKCS11 wallets contain information that points to the token for private key access.

You can use the wallet containing PKCS #11 information just as you would use any Oracle wallet, except the private keys are stored on the hardware device and the cryptographic operations are performed on the device as well.

Related Topics

Oracle Database Enterprise User Security Administrator's Guide

23.15.2 Configuring Your System to Use nCipher Hardware Security Modules

You can configure your system to use nCipher hardware security modules for cryptographic processing.

- About Configuring Your System to Use nCipher Hardware Security Modules
 Hardware security modules made by nCipher Corporation are certified to operate with
 Oracle Database.
- Oracle Components Required To Use an nCipher Hardware Security Module
 To use an nCipher hardware security module, you must have a special set of components.
- Directory Path Requirements for Installing an nCipher Hardware Security Module
 The nCipher hardware security module uses the nCipher PKCS #11 library.

23.15.2.1 About Configuring Your System to Use nCipher Hardware Security Modules

Hardware security modules made by nCipher Corporation are certified to operate with Oracle Database.

These modules provide a secure way to store keys and off-load cryptographic processing. Primarily, these devices provide the following benefits:

- Off-load cryptographic processing that frees your server to respond to other requests
- Secure private key storage on the device
- Allow key administration through the use of smart cards





You must contact your nCipher representative to obtain certified hardware and software to use with Oracle Database.

23.15.2.2 Oracle Components Required To Use an nCipher Hardware Security Module

To use an nCipher hardware security module, you must have a special set of components.

These components are as follows:

- nCipher Hardware Security Module
- Supporting nCipher PKCS #11 library

The following platform-specific PKCS#11 library is required:

- libcknfast.so library for UNIX 32-Bit
- libcknfast-64.so library for UNIX 64-Bit
- cknfast.dll library for Windows

Note:

You must contact your nCipher representative to have the hardware security module or the secure accelerator installed, and to acquire the necessary library.

These tasks must be performed before you can use an nCipher hardware security module with Oracle Database.

23.15.2.3 Directory Path Requirements for Installing an nCipher Hardware Security Module

The nCipher hardware security module uses the nCipher PKCS #11 library.

To use the secure accelerator, you must provide the absolute path to the directory that contains the nCipher PKCS #11 library (including the library name) when you create the wallet by using Oracle Wallet Manager. This enables the library to be loaded at runtime.

Typically, the nCipher card is installed at the following locations:

- /opt/nfast for UNIX
- C:\nfast for Windows

The nCipher PKCS #11 library is located at the following location for typical installations:

- /opt/nfast/toolkits/pkcs11/libcknfast.so for UNIX 32-Bit
- /opt/nfast/toolkits/pkcs11/libcknfast-64.so for UNIX 64-Bit
- C:\nfast\toolkits\pkcs11\cknfast.dll for Windows



Note:

Use the 32-bit library version when using the 32-bit release of Oracle Database and use the 64-bit library version when using the 64-bit release of Oracle Database. For example, use the 64-bit nCipher PKCS #11 library for the Oracle Database for Solaris Operating System (SPARC 64-bit).

23.15.3 Configuring Your System to Use SafeNET Hardware Security Modules

You can configure your system to use SafeNET hardware security modules for cryptographic processing.

- About Configuring Your System to Use SafeNET Hardware Security Modules
 Hardware security modules made by SafeNET Incorporated are certified to operate with
 Oracle Database.
- Oracle Components Required for SafeNET Luna SA Hardware Security Modules
 To use a SafeNET Luna SA hardware security module, you must have a special set of
 components.
- Directory Path Requirements for Installing a SafeNET Hardware Security Module The SafeNET hardware security module uses the SafeNET PKCS #11 library.

23.15.3.1 About Configuring Your System to Use SafeNET Hardware Security Modules

Hardware security modules made by SafeNET Incorporated are certified to operate with Oracle Database.

These modules provide a secure way to store keys and off-load cryptographic processing. Primarily, these devices provide the following benefits:

- Off-load of cryptographic processing to free your server to respond to more requests
- Secure private key storage on the device

Note

You must contact your SafeNET representative to obtain certified hardware and software to use with Oracle Database.

23.15.3.2 Oracle Components Required for SafeNET Luna SA Hardware Security Modules

To use a SafeNET Luna SA hardware security module, you must have a special set of components.

These components are as follows:

SafeNET Luna SA Hardware Security Module



Supporting SafeNET Luna SA PKCS #11 library

The following platform-specific PKCS#11 library is required:

- libCryptoki2.so library for UNIX
- cryptoki.dll library for Windows

Note:

You must contact your SafeNET representative to have the hardware security module or the secure accelerator installed, and to acquire the necessary library.

These tasks must be performed before you can use a SafeNET hardware security module with Oracle Database.

23.15.3.3 Directory Path Requirements for Installing a SafeNET Hardware Security Module

The SafeNET hardware security module uses the SafeNET PKCS #11 library.

To use the secure accelerator, you must provide the absolute path to the directory that contains the SafeNET PKCS #11 library (including the library name) when you create the wallet using Oracle Wallet Manager. This enables the library to be loaded at runtime.

Typically, the SafeNET Luna SA client is installed at the following location:

- /usr/lunasa for UNIX
- C:\Program Files\LunaSA for Windows

The SafeNET Luna SA PKCS #11 library is located at the following location for typical installations:

- /usr/lunasa/lib/libCryptoki2.so for UNIX
- C:\Program Files\LunaSA\cryptoki2.dll for Windows

23.15.4 Troubleshooting Using Hardware Security Modules

Oracle provides troubleshooting advice for hardware security modules.

- Errors in the Oracle Net Trace Files
 To detect whether the module is being used, you can turn on Oracle Net tracing.
- Error Messages Associated with Using Hardware Security Modules
 Errors that are associated with using PKCS #11 hardware security modules can appear.

23.15.4.1 Errors in the Oracle Net Trace Files

To detect whether the module is being used, you can turn on Oracle Net tracing.

If the wallet contains PKCS #11 information and the private key on the module is being used, then you will see the following entries in the Oracle Net tracing file without error messages logged between entry and exit:

```
nzpkcs11_Init: entry
nzpkcs11CP_ChangeProviders: entry
```

```
nzpkcs11CP_ChangeProviders: exit
nzpkcs11GPK_GetPrivateKey: entry
nzpkcs11GPK_GetPrivateKey: exit
nzpkcs11_Init: exit
...
nzpkcs11_Decrypt: entry
nzpkcs11_Decrypt: exit
nzpkcs11_Sign: entry
nzpkcs11_Sign: exit
```

See Also:

Oracle Database Net Services Administrator's Guide for information about setting tracing parameters to enable Oracle Net tracing

23.15.4.2 Error Messages Associated with Using Hardware Security Modules

Errors that are associated with using PKCS #11 hardware security modules can appear.

ORA-43000: PKCS11: library not found

Cause: The system cannot locate the PKCS #11 library at the location specified when the wallet was created. This happens only when the library is moved after the wallet is created.

Action: Copy the PKCS #11 library back to its original location where it was when the wallet was created.

ORA-43001: PKCS11: token not found

Cause: The smart card that was used to create the wallet is not present in the hardware security module slot.

Action: Ensure that the smart card that was used when the wallet was created is present in the hardware security module slot.

ORA-43002: PKCS11: passphrase is wrong

Cause: This can occur when an incorrect password is specified at wallet creation, or the PKCS #11 device password is changed after the wallet is created and not updated in the wallet by using Oracle Wallet Manager.

Action: Depending on the cause, take one of the following actions:

If you see this error during wallet creation, then check to ensure that you have the correct password and reenter it.

If the password changed after wallet creation, then use Oracle Wallet Manager to open the wallet and enter a new password.

See Also:

Oracle Database Enterprise User Security Administrator's Guide about creating an Oracle wallet to store hardware security credentials





The nCipher log file is in the directory where the module is installed at the following location:

/log/logfile

See Also:

nCipher and SafeNET documentation for more information about troubleshooting nCipher and SafeNET devices



Configuring RADIUS Authentication

RADIUS is a client/server security protocol widely used to enable remote authentication and access.

- About Configuring RADIUS Authentication
 - An Oracle Database network can use any authentication method that supports the RADIUS standard.
- RADIUS Components
 - RADIUS has a set of authentication components that enable you to manage configuration settings.
- RADIUS Authentication Modes
 - User authentication can take place either through synchronous authentication mode or challenge-response (asynchronous) authentication mode.
- Using RADIUS to Log in to a Database
 - You can use RADIUS to log into a database by using either synchronous authentication mode or challenge-response mode.
- RSA ACE/Server Configuration Checklist
 If you are using an RSA ACE/Server RADIUS server, check the host agent and SecurID tokens for this server before making the initial connection.

24.1 About Configuring RADIUS Authentication

An Oracle Database network can use any authentication method that supports the RADIUS standard.

The supported RADIUS standard includes token cards and smart cards when you install and configure the RADIUS protocol. Oracle Database uses RADIUS in a client/server network environment. Moreover, when you use RADIUS, you can change the authentication method without modifying either the Oracle client or the Oracle database server.

From an end user's perspective, the entire authentication process is transparent. When the user seeks access to an Oracle database server, the Oracle database server, acting as the RADIUS client, notifies the RADIUS server. The RADIUS server then:

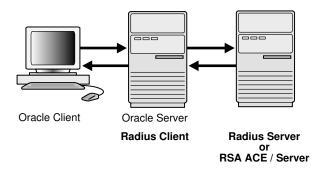
- Looks up the user's security information
- Passes authentication and authorization information between the appropriate authentication server or servers and the Oracle database server
- Grants the user access to the Oracle database server
- Logs session information, including when, how often, and for how long the user was connected to the Oracle database server

Note:

Oracle Database does not support RADIUS authentication over database links.

Figure 24-1 illustrates the Oracle Database-RADIUS environment.

Figure 24-1 RADIUS in an Oracle Environment



The Oracle Database server acts as the RADIUS client, passing information between the Oracle client and the RADIUS server. Similarly, the RADIUS server passes information between the Oracle database server and the appropriate authentication servers.

A RADIUS server vendor is often the authentication server vendor as well. In this case authentication can be processed on the RADIUS server. For example, the RSA ACE/Server is both a RADIUS server and an authentication server. It thus authenticates the user's pass code.

Note:

SecurID, an authentication product of RSA Security, Inc., though not directly supported by Oracle Database, has been certified as RADIUS-compliant. You can therefore, run SecurID under RADIUS.

Refer to the RSA Security SecurID documentation for further information.

See Also:

 ${\it Oracle\ Database\ Net\ Services\ Reference\ for\ information\ about\ the\ {\tt sqlnet.ora}\ file}$

24.2 RADIUS Components

RADIUS has a set of authentication components that enable you to manage configuration settings.

Table 24-1 lists the authentication components.

Table 24-1 RADIUS Authentication Components

Component	Stored Information
Oracle client	Configuration setting for communicating through RADIUS.
Oracle database server/ RADIUS client	Configuration settings for passing information between the Oracle client and the RADIUS server.
	The secret key file.
RADIUS server	Authentication and authorization information for all users.
	Each client's name or IP address.
	Each client's shared secret.
	Unlimited number of menu files enabling users already authenticated to select different login options without reconnecting.
Authentication server or servers	User authentication information such as pass codes and PINs, depending on the authentication method in use.
	Note: The RADIUS server can also be the authentication server.

24.3 RADIUS Authentication Modes

User authentication can take place either through synchronous authentication mode or challenge-response (asynchronous) authentication mode.

- Synchronous Authentication Mode
 In the synchronous mode, RADIUS lets you use various authentication methods, including passwords and SecurID token cards.
- Challenge-Response (Asynchronous) Authentication Mode
 When the system uses the asynchronous mode, the user does not need to enter a user name and password at the SQL*Plus CONNECT string.

24.3.1 Synchronous Authentication Mode

In the synchronous mode, RADIUS lets you use various authentication methods, including passwords and SecurID token cards.

- Sequence for Synchronous Authentication Mode
 The sequence of synchronous authentication mode is comprised of six steps.
- Example: Synchronous Authentication with SecurID Token Cards
 With SecurID authentication, each user has a token card that displays a dynamic number that changes every sixty seconds.

24.3.1.1 Sequence for Synchronous Authentication Mode

The sequence of synchronous authentication mode is comprised of six steps.

Figure 24-2 shows the sequence in which synchronous authentication occurs.

Client Oracle server/RADIUS client Server Server

Figure 24-2 Synchronous Authentication Sequence

The following steps describe the synchronous authentication sequence:

- A user logs in by entering a connect string, pass code, or other value. The client system
 passes this data to the Oracle database server.
- 2. The Oracle database server, acting as the RADIUS client, passes the data from the Oracle client to the RADIUS server.
- The RADIUS server passes the data to the appropriate authentication server, such as Smart Card or SecurID ACE for validation.
- 4. The authentication server sends either an Access Accept or an Access Reject message back to the RADIUS server.
- The RADIUS server passes this response to the Oracle database server/RADIUS client.
- 6. The Oracle database server/RADIUS client passes the response back to the Oracle client.

24.3.1.2 Example: Synchronous Authentication with SecurID Token Cards

With SecurID authentication, each user has a token card that displays a dynamic number that changes every sixty seconds.

To gain access to the Oracle database server/RADIUS client, the user enters a valid pass code that includes both a personal identification number (PIN) and the dynamic number currently displayed on the user's SecurID card. The Oracle database server passes this authentication information from the Oracle client to the RADIUS server, which in this case is the authentication server for validation. Once the authentication server (RSA ACE/Server) validates the user, it sends an *accept* packet to the Oracle database server, which, in turn, passes it to the Oracle client. The user is now authenticated and able to access the appropriate tables and applications.



See Also:

Documentation provided by RSA Security, Inc.

24.3.2 Challenge-Response (Asynchronous) Authentication Mode

When the system uses the asynchronous mode, the user does not need to enter a user name and password at the SQL*Plus CONNECT string.

- Sequence for Challenge-Response (Asynchronous) Authentication Mode
 The sequence for challenge-response (asynchronous) authentication mode is comprised of
 12 steps.
- Example: Asynchronous Authentication with Smart Cards
 With smart card authentication, the user logs in by inserting the smart card into a smart card reader that reads the smart card.
- Example: Asynchronous Authentication with ActivCard Tokens
 One particular ActivCard token is a hand-held device with a keypad and which displays a dynamic password.

24.3.2.1 Sequence for Challenge-Response (Asynchronous) Authentication Mode

The sequence for challenge-response (asynchronous) authentication mode is comprised of 12 steps.

Note:

Challenge-response (Asynchronous) authentication mode is not supported on database servers that run on the Microsoft Windows platform, regardless of the client connection to the database server.

Figure 24-3 shows the sequence in which challenge-response (asynchronous) authentication occurs.

Note:

If the RADIUS server is the authentication server, Steps 3, 4, and 5, and Steps 9, 10, and 11 in Figure 24-3 are combined.



RADIUS Client Oracle Authentication server/ RADIUS Server Server client 2

Figure 24-3 Asynchronous Authentication Sequence

The following steps describe the asynchronous authentication sequence:

- A user initiates a connection to an Oracle database server. The client system passes the data to the Oracle database server.
- 2. The Oracle database server, acting as the RADIUS client, passes the data from the Oracle client to the RADIUS server.
- The RADIUS server passes the data to the appropriate authentication server, such as a Smart Card, SecurID ACE, or token card server.
- The authentication server sends a challenge, such as a random number, to the RADIUS server.
- 5. The RADIUS server passes the challenge to the Oracle database server/RADIUS client.
- The Oracle database server/RADIUS client, in turn, passes it to the Oracle client. A graphical user interface presents the challenge to the user.

- 7. The user provides a response to the challenge. To formulate a response, the user can, for example, enter the received challenge into the token card. The token card provides a dynamic password that is entered into the graphical user interface. The Oracle client passes the user's response to the Oracle database server/RADIUS client.
- The Oracle database server/RADIUS client sends the user's response to the RADIUS server.
- The RADIUS server passes the user's response to the appropriate authentication server for validation.
- The authentication server sends either an Access Accept or an Access Reject message back to the RADIUS server.
- 11. The RADIUS server passes the response to the Oracle database server/RADIUS client.
- 12. The Oracle database server/RADIUS client passes the response to the Oracle client.

24.3.2.2 Example: Asynchronous Authentication with Smart Cards

With smart card authentication, the user logs in by inserting the smart card into a smart card reader that reads the smart card.

The smart card is a plastic card, like a credit card, with an embedded integrated circuit for storing information.

The Oracle client sends the login information contained in the smart card to the authentication server by way of the Oracle database server/RADIUS client and the RADIUS server. The authentication server sends back a challenge to the Oracle client, by way of the RADIUS server and the Oracle database server, prompting the user for authentication information. The information could be, for example, a PIN as well as additional authentication information contained on the smart card.

The Oracle client sends the user's response to the authentication server by way of the Oracle database server and the RADIUS server. If the user has entered a valid number, the authentication server sends an *accept* packet back to the Oracle client by way of the RADIUS server and the Oracle database server. The user is now authenticated and authorized to access the appropriate tables and applications. If the user has entered incorrect information, the authentication server sends back a message rejecting user's access.

24.3.2.3 Example: Asynchronous Authentication with ActivCard Tokens

One particular ActivCard token is a hand-held device with a keypad and which displays a dynamic password.

When the user seeks access to an Oracle database server by entering a password, the information is passed to the appropriate authentication server by way of the Oracle database server/RADIUS client and the RADIUS server. The authentication server sends back a challenge to the client, by way of the RADIUS server and the Oracle database server. The user types that challenge into the token, and the token displays a number for the user to send in response.

The Oracle client then sends the user's response to the authentication server by way of the Oracle database server and the RADIUS server. If the user has typed a valid number, the authentication server sends an *accept* packet back to the Oracle client by way of the RADIUS server and the Oracle database server. The user is now authenticated and authorized to access the appropriate tables and applications. If the user has entered an incorrect response, the authentication server sends back a message rejecting the user's access.



24.4 Enabling RADIUS Authentication, Authorization, and Accounting

To enable RADIUS authentication, authorization, and accounting, you can use Oracle Net Manager.

- Step 1: Configure RADIUS Authentication
 - To configure RADIUS authentication, you must first configure it on the Oracle client, then the server. Afterward, you can configure additional RADIUS features.
- Step 2: Create a User and Grant Access
 After you complete the RADIUS authentication, you must create an Oracle Database user who is responsible for the RADIUS configuration.
- Step 3: Configure External RADIUS Authorization (Optional)
 You must configure the Oracle server, the Oracle client, and the RADIUS server to RADIUS users who must connect to an Oracle database.
- Step 4: Configure RADIUS Accounting
 RADIUS accounting logs information about access to the Oracle database server and stores it in a file on the RADIUS accounting server.
- Step 5: Add the RADIUS Client Name to the RADIUS Server Database
 The RADIUS server that you select must comply with RADIUS standards.
- Step 6: Configure the Authentication Server for Use with RADIUS
 After you add the RADIUS client name to the RADIUS server database, you can configure the authentication server to use the RADIUS.
- Step 8: Configure Mapping Roles
 If the RADIUS server supports vendor type attributes, then you can manage roles by storing them in the RADIUS server.

24.4.1 Step 1: Configure RADIUS Authentication

To configure RADIUS authentication, you must first configure it on the Oracle client, then the server. Afterward, you can configure additional RADIUS features.



Unless otherwise indicated, perform these configuration tasks by using Oracle Net Manager or by using any text editor to modify the sqlnet.ora file. Be aware that in a multitenant environment, the settings in the sqlnet.ora file apply to all pluggable databases (PDBs).

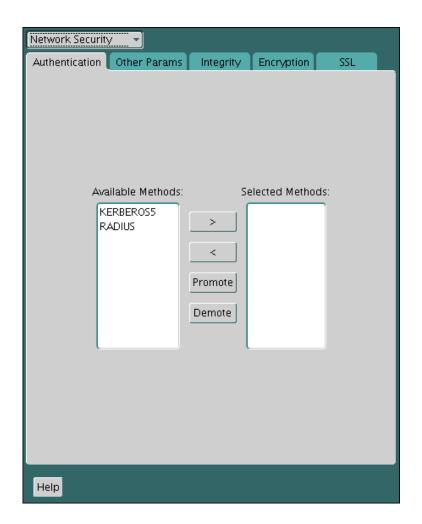
Step 1A: Configure RADIUS on the Oracle Client
 You can use Oracle Net Manager to configure RADIUS on the Oracle client.

- Step 1B: Configure RADIUS on the Oracle Database Server
 You must create a file to hold the RADIUS key and store this file on the Oracle database
 server. Then you must configure the appropriate parameters in the sqlnet.ora file.
- Step 1C: Configure Additional RADIUS Features
 You can change the default settings, configure the challenge-response mode, and set parameters for an alternate RADIUS server.

24.4.1.1 Step 1A: Configure RADIUS on the Oracle Client

You can use Oracle Net Manager to configure RADIUS on the Oracle client.

- 1. Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.The Network Security tabbed window appears.
- 4. Select the **Authentication** tab. (It should be selected by default.)



- From the Available Methods list, select RADIUS.
- Select the right-arrow (>) to move RADIUS to the Selected Methods list.

Move any other methods you want to use in the same way.

Arrange the selected methods in order of required usage by selecting a method in the Selected Methods list, and clicking **Promote** or **Demote** to position it in the list.

For example, put RADIUS at the top of the list for it to be the first service used.

8. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entry:

SQLNET.AUTHENTICATION SERVICES=(RADIUS)

24.4.1.2 Step 1B: Configure RADIUS on the Oracle Database Server

You must create a file to hold the RADIUS key and store this file on the Oracle database server. Then you must configure the appropriate parameters in the sqlnet.ora file.

- Step 1B (1): Create the RADIUS Secret Key File on the Oracle Database Server First, you must create the RADIUS secret key file.
- Step 1B (2): Configure RADIUS Parameters on the Server (sqlnet.ora file)
 After you create RADIUS secret key file, you are ready to configure the appropriate parameters in the sqlnet.ora file.
- Step 1B (3): Set Oracle Database Server Initialization Parameters
 After you configure the sqlnet.ora file, you must configure the init.ora initialization file.

24.4.1.2.1 Step 1B (1): Create the RADIUS Secret Key File on the Oracle Database Server

First, you must create the RADIUS secret key file.

Obtain the RADIUS secret key from the RADIUS server.

For each RADIUS client, the administrator of the RADIUS server creates a shared secret key, which must be less than or equal to 16 characters.

- On the Oracle database server, create a directory:
 - (UNIX) \$ORACLE HOME/network/security
 - (Windows) ORACLE BASE\ORACLE HOME\network\security
- Create the file radius.key to hold the shared secret copied from the RADIUS server. Place the file in the directory you created in Step 2.
- 4. Copy the shared secret key and paste it (and nothing else) into the radius. key file created on the Oracle database server.
- For security purposes, change the file permission of radius.key to read only, accessible only by the Oracle owner.

Oracle relies on the file system to keep this file secret.



The RADIUS server administration documentation, for information about obtaining the secret key

24.4.1.2.2 Step 1B (2): Configure RADIUS Parameters on the Server (sqlnet.ora file)

After you create RADIUS secret key file, you are ready to configure the appropriate parameters in the sqlnet.ora file.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- 3. From the Naming list, select Network Security.

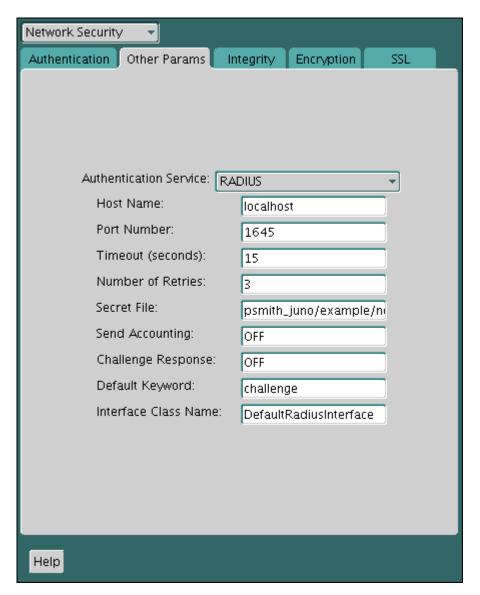
The Network Security tabbed window appears.

- 4. Select the Authentication tab.
- 5. From the Available Methods list, select **RADIUS**.
- **6.** Move RADIUS to the **Selected Methods** list by choosing the right-arrow (>).
- 7. To arrange the selected methods in order of desired use, select a method in the Selected Methods list, and select **Promote** or **Demote** to position it in the list.

For example, if you want RADIUS to be the first service used, then put it at the top of the list.

- 8. Select the **Other Params** tab.
- 9. From the Authentication Service list, select RADIUS.
- **10.** In the **Host Name** field, accept the **localhost** as the default primary RADIUS server, or enter another host name.





- 11. Ensure that the default value of the Secret File field is valid.
- **12.** From the **File** menu, select **Save Network Configuration**.

The sqlnet.ora file is updated with the following entries:

```
SQLNET.AUTHENTICATION_SERVICES=RADIUS
SQLNET.RADIUS AUTHENTICATION=RADIUS server {hostname|IP address}
```



The $IP_address$ can either be an Internet Protocol Version 4 (IPv4) or Internet Protocol Version 6 (IPv6) address. The RADIUS adapter supports both IPv4 and IPv6 based servers.

24.4.1.2.3 Step 1B (3): Set Oracle Database Server Initialization Parameters

After you configure the sqlnet.ora file, you must configure the init.ora initialization file.

Add the following setting to the init.ora file.

```
OS AUTHENT PREFIX=""
```

By default, the <code>init.ora</code> file is located in the <code>ORACLE_HOME/dbs</code> directory (or the same location of the data files) on Linux and UNIX systems, and in the <code>ORACLE_HOME\database</code> directory on Windows.

2. Restart the database.

For example:

SQL> SHUTDOWN SQL> STARTUP



Oracle Database Reference for information about setting initialization parameters

24.4.1.3 Step 1C: Configure Additional RADIUS Features

You can change the default settings, configure the challenge-response mode, and set parameters for an alternate RADIUS server.

- Step 1C(1): Change Default Settings
 You can use Oracle Net Manager to change the default RADIUS settings.
- Step 1C(2): Configure Challenge-Response Mode
 To configure challenge-response mode, you must specify information such as a dynamic password that you obtain from a token card.
- Step 1C(3): Set Parameters for an Alternate RADIUS Server
 If you are using an alternate RADIUS server, then you must set additional parameters.

24.4.1.3.1 Step 1C(1): Change Default Settings

You can use Oracle Net Manager to change the default RADIUS settings.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.

The Network Security tabbed window appears.

- 4. Click the Other Params tab.
- 5. From the Authentication Service list, select RADIUS.
- 6. Change the default setting for any of the following fields:
 - Port Number: Specifies the listening port of the primary RADIUS server. The default value is 1645.

- Timeout (seconds): Specifies the time the Oracle database server waits for a response from the primary RADIUS server. The default is 15 seconds.
- Number of Retries: Specifies the number of times the Oracle database server resends messages to the primary RADIUS server. The default is three retries. For instructions on configuring RADIUS accounting, see Step 4: Configure RADIUS Accounting.
- Secret File: Specifies the location of the secret key on the Oracle database server.
 The field specifies the location of the secret key file, not the secret key itself. For information about specifying the secret key, see Step 1B (1): Create the RADIUS Secret Key File on the Oracle Database Server.
- 7. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entries:

```
SQLNET.RADIUS_AUTHENTICATION_PORT=(PORT)
SQLNET.RADIUS_AUTHENTICATION_TIMEOUT=(NUMBER OF SECONDS TO WAIT FOR response)
SQLNET.RADIUS_AUTHENTICATION_RETRIES=(NUMBER OF TIMES TO RE-SEND TO RADIUS server)
SQLNET.RADIUS_SECRET=(path/radius.key)
```

24.4.1.3.2 Step 1C(2): Configure Challenge-Response Mode

To configure challenge-response mode, you must specify information such as a dynamic password that you obtain from a token card.

With the RADIUS adapter, this interface is Java-based to provide optimal platform independence.



Third party vendors of authentication devices must customize this graphical user interface to fit their particular device. For example, a smart card vendor would customize the Java interface so that the Oracle client reads data, such as a dynamic password, from the smart card. When the smart card receives a challenge, it responds by prompting the user for more information, such as a PIN.

To configure challenge-response mode:

- 1. If you are using JDK 1.1.7 or JRE 1.1.7, then set the <code>JAVA_HOME</code> environment variable to the JRE or JDK location on the system where the Oracle client is run:
 - On UNIX, enter this command at the prompt:

```
% setenv JAVA HOME /usr/local/packages/jre1.1.7B
```

 On Windows, select Start, Settings, Control Panel, System, Environment, and set the JAVA HOME variable as follows:

```
c:\java\jre1.1.7B
```

This step is not required for any other JDK/JRE version.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:

netmgr



- (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 3. Expand Oracle Net Configuration, and from Local, select Profile.
- 4. From the Naming list, select Network Security.

The Network Security tabbed window appears.

- 5. From the Authentication Service list, select **RADIUS**.
- 6. In the **Challenge Response** field, enter **ON** to enable challenge-response.
- In the Default Keyword field, accept the default value of the challenge or enter a keyword for requesting a challenge from the RADIUS server.

The keyword feature is provided by Oracle and supported by some, but not all, RADIUS servers. You can use this feature only if your RADIUS server supports it.

By setting a keyword, you let the user avoid using a password to verify identity. If the user does *not* enter a password, the keyword you set here is passed to the RADIUS server which responds with a challenge requesting, for example, a driver's license number or birth date. If the user *does* enter a password, the RADIUS server may or may not respond with a challenge, depending upon the configuration of the RADIUS server.

 In the Interface Class Name field, accept the default value of DefaultRadiusInterface or enter the name of the class you have created to handle the challenge-response conversation.

If other than the default RADIUS interface is used, then you also must edit the sqlnet.ora file to enter SQLNET.RADIUS_CLASSPATH=(location), where location is the complete path name of the jar file. It defaults to \$ORACLE_HOME/network/jlib/netradius.jar: \$ORACLE_HOME/JRE/lib/vt.jar

9. From the File menu, select Save Network Configuration.

The sglnet.ora file is updated with the following entries:

```
SQLNET.RADIUS_CHALLENGE_RESPONSE=([ON | OFF])
SQLNET.RADIUS_CHALLENGE_KEYWORD=(KEYWORD)
SQLNET.RADIUS_AUTHENTICATION_INTERFACE=(name of interface including the package name delimited by "/" for ".")
```

See Also:

Integrating Authentication Devices Using RADIUS for information about how to customize the challenge-response user interface

24.4.1.3.3 Step 1C(3): Set Parameters for an Alternate RADIUS Server

If you are using an alternate RADIUS server, then you must set additional parameters.

Set the following parameters in the sqlnet.ora file:

```
SQLNET.RADIUS_ALTERNATE=(hostname or ip address of alternate radius server)
SQLNET.RADIUS_ALTERNATE_PORT=(1812)
SQLNET.RADIUS_ALTERNATE_TIMEOUT=(number of seconds to wait for response)
SQLNET.RADIUS_ALTERNATE_RETRIES=(number of times to re-send to radius server)
```



24.4.2 Step 2: Create a User and Grant Access

After you complete the RADIUS authentication, you must create an Oracle Database user who is responsible for the RADIUS configuration.

1. Connect to the CDB root or to the PDB in which RADIUS is implemented.

For example:

```
CONNECT system@pdb_name;
Enter password: password
```

Create the user as a common user if you connected to the CDB root, or as a local user if you connected to a PDB..

```
CREATE USER username IDENTIFIED EXTERNALLY; GRANT CREATE SESSION TO USER user name;
```

3. Enter the user username in the RADIUS server's users file.



Administration documentation for the RADIUS server

24.4.3 Step 3: Configure External RADIUS Authorization (Optional)

You must configure the Oracle server, the Oracle client, and the RADIUS server to RADIUS users who must connect to an Oracle database.

- Step 3A: Configure the Oracle Server (RADIUS Client)
 You can edit the init.ora file to configure an Oracle server for a RADIUS client.
- Step 3B: Configure the Oracle Client Where Users Log In Next, you must configure the Oracle client where users log in.
- Step 3C: Configure the RADIUS Server
 To configure the RADIUS server, you must modify the RADIUS server attribute configuration file.

24.4.3.1 Step 3A: Configure the Oracle Server (RADIUS Client)

You can edit the init.ora file to configure an Oracle server for a RADIUS client.

To do so, you must modify the init.ora file, restart the database, and the set the RADIUS challenge-response mode.

1. Add the os_{ROLES} parameter to the init.ora file and set this parameter to TRUE as follows:

```
OS_ROLES=TRUE
```

By default, the init.ora file is located in the <code>ORACLE_HOME/dbs</code> directory (or the same location of the data files) on Linux and UNIX systems, and in the <code>ORACLE_HOME/database</code> directory on Windows.

2. Restart the database so that the system can read the change to the init.ora file.

For example:



SQL> SHUTDOWN SQL> STARTUP

- 3. Set the RADIUS challenge-response mode to ON for the server if you have not already done so by following the steps listed in Step 1C(2): Configure Challenge-Response Mode.
- 4. Add externally identified users and roles.

24.4.3.2 Step 3B: Configure the Oracle Client Where Users Log In

Next, you must configure the Oracle client where users log in.

• Set the RADIUS challenge-response mode to ON for the client if you have not already done so by following the steps listed in Step 1C(2): Configure Challenge-Response Mode.

24.4.3.3 Step 3C: Configure the RADIUS Server

To configure the RADIUS server, you must modify the RADIUS server attribute configuration file.

1. Add the following attributes to the RADIUS server attribute configuration file:

ATTRIBUTE NAME	CODE	TYPE
VENDOR_SPECIFIC	26	Integer
ORACLE_ROLE	1	String

2. Assign a Vendor ID for Oracle in the RADIUS server attribute configuration file that includes the SMI Network Management Private Enterprise Code of 111.

For example, enter the following in the RADIUS server attribute configuration file:

```
VALUE VENDOR SPECIFIC ORACLE 111
```

3. Using the following syntax, add the <code>ORACLE_ROLE</code> attribute to the user profile of the users who will use external RADIUS authorization:

```
ORA databaseSID rolename[ [A]|[D]]
```

In this specification.:

- ORA designates that this role is used for Oracle purposes
- databaseSID is the Oracle system identifier that is configured in the database init.ora file.

By default, the <code>init.ora</code> file is located in the <code>ORACLE_HOME/dbs</code> directory (or the same location of the data files) on Linux and UNIX systems, and in the <code>ORACLE_HOME\database</code> directory on Windows.

- rolename is the name of role as it is defined in the data dictionary.
- A is an optional character that indicates the user has administrator's privileges for this role.
- D is an optional character that indicates this role is to be enabled by default.

Ensure that RADIUS groups that map to Oracle roles adhere to the ORACLE ROLE syntax.

For example:

```
USERNAME USERPASSWD="user_password", SERVICE TYPE=login user,
```

VENDOR_SPECIFIC=ORACLE,
ORACLE_ROLE=ORA_ora920_sysdba



The RADIUS server administration documentation for information about configuring the server.

24.4.4 Step 4: Configure RADIUS Accounting

RADIUS accounting logs information about access to the Oracle database server and stores it in a file on the RADIUS accounting server.

Use this feature only if both the RADIUS server and authentication server support it.

- Step 4A: Set RADIUS Accounting on the Oracle Database Server
 To set RADIUS accounting on the server, you can use Oracle Net Manager.
- Step 4B: Configure the RADIUS Accounting Server
 RADIUS Accounting Server resides on the same host as the RADIUS authentication
 server or on a separate host.

24.4.4.1 Step 4A: Set RADIUS Accounting on the Oracle Database Server

To set RADIUS accounting on the server, you can use Oracle Net Manager.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:

 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.

The Network Security tabbed window appears.

- 4. Select the Other Params tab.
- From the Authentication Service list, select RADIUS.
- In the Send Accounting field, enter ON to enable accounting or OFF to disable accounting.
- 7. From the **File** menu, select **Save Network Configuration**.

The sqlnet.ora file is updated with the following entry:

SQLNET.RADIUS SEND ACCOUNTING= ON

24.4.4.2 Step 4B: Configure the RADIUS Accounting Server

RADIUS Accounting Server resides on the same host as the RADIUS authentication server or on a separate host.

See the administration documentation for the RADIUS server, for information about configuring RADIUS accounting.

24.4.5 Step 5: Add the RADIUS Client Name to the RADIUS Server Database

The RADIUS server that you select must comply with RADIUS standards.

You can use any RADIUS server that complies with the Internet Engineering Task Force (IETF) RFC #2138, Remote Authentication Dial In User Service (RADIUS), and RFC #2139 RADIUS Accounting standards. Because RADIUS servers vary, consult the documentation for your particular RADIUS server for any unique interoperability requirements.

To add the RADIUS client name to a Livingston RADIUS server:

Open the clients file, which is located in /etc/raddb/clients.

The following text and table appear:

```
\ensuremath{\mathbb{Q}} (#) clients 1.1 2/21/96 Copyright 1991 Livingston Enterprises Inc This file contains a list of clients which are allowed to make authentication requests and their encryption key. The first field is a valid hostname. The second field (separated by blanks or tabs) is the encryption key. Client Name
```

2. In the CLIENT NAME column, enter the host name or IP address of the host on which the Oracle database server is running.

In the KEY column, type the shared secret.

The value you enter in the <code>CLIENT</code> Name column, whether it is the client's name or IP address, depends on the RADIUS server.

3. Save and close the clients file.



Administration documentation for the RADIUS server

24.4.6 Step 6: Configure the Authentication Server for Use with RADIUS

After you add the RADIUS client name to the RADIUS server database, you can configure the authentication server to use the RADIUS.

 Refer to the authentication server documentation for instructions about configuring the authentication servers.

24.4.7 Step 7: Configure the RADIUS Server for Use with the Authentication Server

After you configure the authentication server for use with RADIUS, you can configure the RADIUS server to use the authentication server.

 Refer to the RADIUS server documentation for instructions about configuring the RADIUS server for use with the authentication server.

24.4.8 Step 8: Configure Mapping Roles

If the RADIUS server supports vendor type attributes, then you can manage roles by storing them in the RADIUS server.

The Oracle database server downloads the roles when there is a CONNECT request using RADIUS. To use this feature, you must configure roles on both the Oracle database server and the RADIUS server.

 Use a text editor to set the OS_ROLES parameter in the initialization parameters file on the Oracle database server.

By default, the init.ora file is located in the <code>ORACLE_HOME/dbs</code> directory (or the same location of the data files) on Linux and UNIX systems, and in the <code>ORACLE_HOME\database</code> directory on Windows.

2. Stop and restart the Oracle database server.

For example:

```
SHUTDOWN
STARTUP
```

3. Create each role that the RADIUS server will manage on the Oracle database server with the value IDENTIFIED EXTERNALLY.

To configure roles on the RADIUS server, use the following syntax:

```
ORA_DatabaseName.DatabaseDomainName_RoleName
```

In this specification:

- DatabaseName is the name of the Oracle database server for which the role is being created. This is the same as the value of the DB NAME initialization parameter.
- DatabaseDomainName is the name of the domain to which the Oracle database server belongs. The value is the same as the value of the DB DOMAIN initialization parameter.
- RoleName is name of the role created in the Oracle database server.

For example:

```
ORA USERDB.US.EXAMPLE.COM MANAGER
```

Configure RADIUS challenge-response mode.

Related Topics

- Challenge-Response (Asynchronous) Authentication Mode
 When the system uses the asynchronous mode, the user does not need to enter a user name and password at the SQL*Plus CONNECT string.
- Step 1C(2): Configure Challenge-Response Mode
 To configure challenge-response mode, you must specify information such as a dynamic password that you obtain from a token card.

24.5 Using RADIUS to Log in to a Database

You can use RADIUS to log into a database by using either synchronous authentication mode or challenge-response mode.

Start SQL*Plus and use one of the following ways to log in to the database:

If you are using the synchronous authentication mode, first ensure that challenge-response mode is not turned to ON, and then enter the following command:

```
CONNECT username@database_alias
Enter password: password
```

 If you are using the challenge-response mode, ensure that challenge-response mode is set to ON and then enter the following command:

CONNECT /@database alias



The challenge-response mode can be configured for all login cases.

24.6 RSA ACE/Server Configuration Checklist

If you are using an RSA ACE/Server RADIUS server, check the host agent and SecurID tokens for this server before making the initial connection.

- Ensure that the host agent in the RSA ACE/Server is set up to send a node secret. In version 5.0, this is done by leaving the SENT Node secret box unchecked. If the RSA ACE/Server fails to send a node secret to the agent, then a node verification failure message will be written to the RSA ACE/Server log.
- If you are using RSA SecurID tokens, then ensure that the token is synchronized with the RSA ACE/Server.

See Also:

RSA ACE/Server documentation for specific information about troubleshooting.



Customizing the Use of Strong Authentication

You can configure multiple authentication methods under Oracle Database native network encryption and strong authentication.

- Connecting to a Database Using Strong Authentication
 You can use password authentication to connect to a database that is configured to use strong authentication.
- Disabling Strong Authentication and Native Network Encryption
 You can use Oracle Net Manager to disable strong authentication and native network encryption.
- Configuring Multiple Authentication Methods
 Many networks use more than one authentication method on a single security server.
- Configuring Oracle Database for External Authentication
 You can use parameters to configure Oracle Database for network authentication.

25.1 Connecting to a Database Using Strong Authentication

You can use password authentication to connect to a database that is configured to use strong authentication.

1. To connect to an Oracle database server using a user name and password when an Oracle network and strong authentication method has been configured, disable the external authentication.

You must first follow the instructions in Disabling Strong Authentication and Native Network Encryption to disable the external authentication before you can connect to an Oracle Database server using a user name and password when an Oracle network and strong authentication method has been configured.

With the external authentication disabled, connect to the database using the following format:

```
% sqlplus username@net_service_name
Enter password: password
```

For example:

```
% sqlplus hr@emp
Enter password: password
```



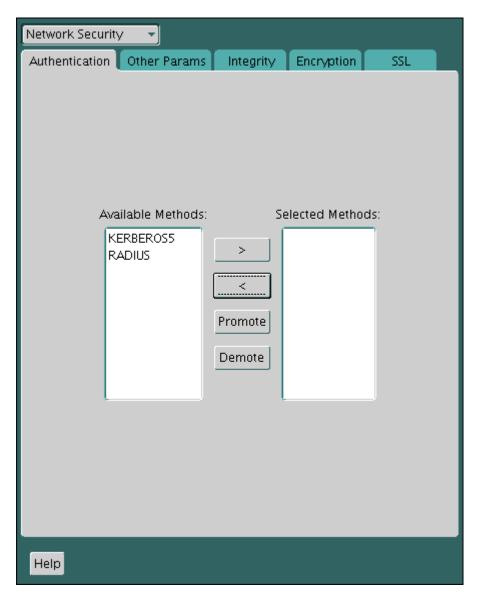
You can configure multiple authentication methods, including both externally authenticated users and password authenticated users, on a single database.

25.2 Disabling Strong Authentication and Native Network Encryption

You can use Oracle Net Manager to disable strong authentication and native network encryption.

- 1. Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.The Network Security tabbed window appears.
- 4. Select the **Authentication** tab (which is selected by default).
- **5.** Sequentially move all authentication methods from the Selected Method list to the Available Methods list by selecting a method and choosing the left arrow [<].





- 6. Select the Encryption tab.
- 7. Do the following:
 - From the Encryption menu, select SERVER.
 - Set Encryption Type to rejected.
 - In the Encryption Seed field, enter a valid encryption seed if an encryption seed was used.
 - Under Select Methods, move any methods to the Available Methods field.
- 8. Repeat these steps disable native network encryption for the client, by selecting **CLIENT** from the **Encryption** menu.
- 9. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entries to indicate that strong authentication and native network encryption are disabled:

Strong authentication:

 $SQLNET.AUTHENTICATION_SERVICES = (NONE)$

If you are using local database password authentication, then you can also set ${\tt SQLNET.AUTHENTICATION_SERVICES=(NONE)}$ in the client. This setting improves client performance.

For native network encryption, you can set it individually, for the server side and for the client side. The following examples show native network encryption being disabled for both the server and the client:

```
SQLNET.ENCRYPTION_SERVER = REJECTED
SQLNET.ENCRYPTION CLIENT = REJECTED
```

Be aware that in a multitenant environment, the settings in the sqlnet.ora file apply to all pluggable databases (PDBs).

Related Topics

About the Values for Negotiating Encryption and Integrity
 Oracle Net Manager can be used to specify four possible values for the encryption and integrity configuration parameters.

25.3 Configuring Multiple Authentication Methods

Many networks use more than one authentication method on a single security server.

Accordingly, Oracle Database lets you configure your network so that Oracle clients can use a specific authentication method, and Oracle database servers can accept any method specified.

You can set up multiple authentication methods on both client and server systems either by using Oracle Net Manager, or by using any text editor to modify the sqlnet.ora file. Use Oracle Net Manager to add authentication methods to both clients and servers.

- Start Oracle Net Manager.
 - (UNIX) From \$ORACLE_HOME/bin, enter the following command at the command line:
 netmgr
 - (Windows) Select Start, Programs, Oracle HOME_NAME, Configuration and Migration Tools, then Net Manager.
- 2. Expand Oracle Net Configuration, and from Local, select Profile.
- From the Naming list, select Network Security.

The Network Security tabbed window appears.

- 4. Select the **Authentication** tab.
- 5. Select a method listed in the Available Methods list.
- 6. Sequentially move selected methods to the Selected Methods list by clicking the right arrow (>).
- 7. Arrange the selected methods in order of desired use.

To do this, select a method in the Selected Methods list, and select **Promote** or **Demote** to position it in the list.

8. From the File menu, select Save Network Configuration.

The sqlnet.ora file is updated with the following entry, listing the selected authentication methods:

```
SQLNET.AUTHENTICATION SERVICES = (KERBEROS5, RADIUS)
```



SecurID functionality is available through RADIUS; RADIUS support is built into the RSA ACE/Server.

Related Topics

Configuring RADIUS Authentication
 RADIUS is a client/server security protocol widely used to enable remote authentication
 and access.

25.4 Configuring Oracle Database for External Authentication

You can use parameters to configure Oracle Database for network authentication.

- Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora
 The SQLNET.AUTHENTICATION_SERVICES parameter defines the authentication method and version to be used.
- Setting OS_AUTHENT_PREFIX to a Null Value
 The OS_AUTHENT_PREFIX parameter specifies a prefix that Oracle Database uses to authenticate users who attempt to connect to the server.

25.4.1 Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora

The SQLNET.AUTHENTICATION_SERVICES parameter defines the authentication method and version to be used.

You must set the SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file for all clients and servers to enable each to use a supported authentication method.

Set the SQLNET.AUTHENTICATION SERVICES parameter using the following syntax:

```
SQLNET.AUTHENTICATION SERVICES=(oracle authentication method)
```

For example, for all clients and servers using Kerberos authentication:

```
SQLNET.AUTHENTICATION SERVICES=(KERBEROS5)
```

By default, the sqlnet.ora file is located in the <code>ORACLE_HOME/network/admin</code> directory or in the location set by the <code>TNS_ADMIN</code> environment variable. Ensure that you have properly set the <code>TNS_ADMIN</code> variable to point to the correct <code>sqlnet.ora</code> file.

If you are only using local database password authentication, then set the SQLNET.AUTHENTICATION SERVICES as follows for better client performance:

SQLNET.AUTHENTICATION_SERVICES=(NONE)

Related Topics

SQL*Plus User's Guide and Reference



25.4.2 Setting OS_AUTHENT_PREFIX to a Null Value

The OS_AUTHENT_PREFIX parameter specifies a prefix that Oracle Database uses to authenticate users who attempt to connect to the server.

Authentication service-based user names can be long, and Oracle user names are limited to 128 bytes. Oracle strongly recommends that you set the <code>OS_AUTHENT_PREFIX</code> parameter to a null value.

In the initialization file for the database instance, set OS AUTHENT PREFIX as follows:

```
OS AUTHENT PREFIX=""
```

Note the following:

- The default value for OS AUTHENT PREFIX is OPS\$; however, you can set it to any string.
- If a database already has the OS_AUTHENT_PREFIX set to a value other than NULL (" "), then do not change it, because it can inhibit previously created, externally identified users from connecting to the Oracle server.

After you have set OS_AUTHENT_PREFIX to null, then you can create external users by using the following syntax:

```
CREATE USER os authent prefix username IDENTIFIED EXTERNALLY;
```

For example, to create the user king:

```
CREATE USER king IDENTIFIED EXTERNALLY;
```

The advantage of creating a user in this way is that you no longer need to maintain different user names for externally identified users. This is true for all supported authentication methods.



Part VI

Monitoring Database Activity with Auditing

Part VI describes how to monitor database activity with auditing.

- Introduction to Auditing
 Privileged users can create policies that track the changes that all users, including other privileged users, make in the database.
- Configuring Audit Policies
 Unified auditing supports custom unified audit policies, predefined unified auditing policies, and fine-grained auditing.
- Administering the Audit Trail Users who have been granted the AUDIT_ADMIN role can manage the audit trail, archive the audit trail, and purge audit trail records.



Introduction to Auditing

Privileged users can create policies that track the changes that all users, including other privileged users, make in the database.



Except where noted, this part describes how to use pure unified auditing, in which all audit records are centralized in one place.

What Is Auditing?

Auditing is the monitoring and recording of database activity, from both database users and nondatabase users.

Why Is Auditing Used?

You typically use auditing to monitor user activity.

Best Practices for Auditing

You should follow best practices guidelines for auditing.

What Is Unified Auditing?

In unified auditing, the unified audit trail captures audit information from a variety of sources.

Benefits of the Unified Audit Trail

The benefits of a unified audit trail are many.

Checking if Your Database Has Migrated to Pure Unified Auditing

The V\$OPTION dynamic view indicates if your database has been migrated to using pure unified auditing (that is, traditional auditing is turned off).

Mixed Mode Auditing

Mixed mode auditing is the default auditing in a newly installed database.

Who Can Perform Auditing?

Oracle provides two roles for users who perform auditing: AUDIT_ADMIN and AUDIT_VIEWER.

Unified Auditing in a Multitenant Environment

You can apply audit settings to individual PDBs or to the CDB, depending on the type of policy.

Auditing in a Distributed Database

Auditing is site autonomous in that a database instance audits only the statements issued by directly connected users.

Related Topics

Guidelines for Auditing

Oracle provides guidelines for auditing.

26.1 What Is Auditing?

Auditing is the monitoring and recording of database activity, from both database users and nondatabase users.

"Nondatabase users" refers to application users who are recognized in the database using the CLIENT_IDENTIFIER attribute. To audit this type of user, you can use a unified audit policy condition, a fine-grained audit policy, or Oracle Database Real Application Security.

This guide describes how to use unified auditing to create policies that consolidate audit trails from different Oracle Database components, such as fine-grained auditing or Oracle Database Vault, into one consolidated audit trail. This audit trail is viewable in the UNIFIED_AUDIT_TRAIL data dictionary view. (Other unified audit trail views, such as AUDIT_UNIFIED_POLICIES, are available.) A consolidated audit data trail enables you to run analysis reports on an entire set of audit data in one operation, rather than having to first gather them into one location before performing the analysis. Audit mining tools such as Oracle Audit Vault can look at one location rather than several in order to gather audit records. A unified audit trail ensures that the audit information is consistently formatted and contains consistent fields.

Alternatively, you can use traditional auditing, which is described in the Oracle Database release 11.2 *Oracle Database Security Guide*.

You can base auditing on individual actions, such as the type of SQL statement executed, or on combinations of session metadata that can include the user name, application, time, and so on.

You can configure auditing for both successful and failed activities, and include or exclude specific users from the audit. In a multitenant environment, you can audit individual actions of the pluggable database (PDB) or individual actions in the entire multitenant container database (CDB). In addition to auditing the standard activities the database provides, auditing can include activities from Oracle Database Real Application Security, Oracle Recovery Manager, Oracle Data Pump, Oracle Data Mining, Oracle Database Vault, Oracle Label Security, and Oracle SQL*Loader direct path events.

Auditing is enabled by default. All audit records are written to the unified audit trail in a uniform format and are made available through the <code>UNIFIED_AUDIT_TRAIL</code> view. These records reside in the <code>AUDSYS</code> schema. The audit records are stored in the <code>SYSAUX</code> tablespace by default. Oracle recommends that you configure a different tablespace for the unified audit trail, which you can do by using the <code>DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_LOCATION</code> procedure. Be aware that for Oracle Database Standard Edition and Express Edition, but not for Enterprise Edition, you can only associate the tablespace for unified auditing once. You should perform this association before you generate any audit records for the unified audit trail. After you have associated the tablespace, you cannot modify it because partitioning is only supported on Enterprise Edition.

You can configure auditing by using any of the following methods:

- Group audit settings into one unified audit policy. You can create one or more unified
 audit policies that define all the audit settings that your database needs. Auditing Activities
 with Unified Audit Policies and the AUDIT Statement describes how to accomplish this.
- Use one of the default unified audit policies. Oracle Database provides predefined unified audit policies that encompass the standard audit settings that most regulatory agencies require. See Auditing Activities with the Predefined Unified Audit Policies.
- Create fine-grained audit policies. You can create fine-grained audit policies that capture
 data such as the time an action occurred. See Auditing Specific Activities with FineGrained Auditing.



Oracle recommends that you audit your databases. Auditing is an effective method of enforcing strong internal controls so that your site can meet its regulatory compliance requirements, as defined in the Sarbanes-Oxley Act. This enables you to monitor business operations, and find any activities that may deviate from company policy. Doing so translates into tightly controlled access to your database and the application software, ensuring that patches are applied on schedule and preventing ad hoc changes. By creating effective audit policies, you can generate an audit record for audit and compliance personnel. Be selective with auditing and ensure that it meets your business compliance needs.

26.2 Why Is Auditing Used?

You typically use auditing to monitor user activity.

Auditing can be used to accomplish the following:

- Enable accountability for actions. These include actions taken in a particular schema, table, or row, or affecting specific content.
- Deter users (or others, such as intruders) from inappropriate actions based on their accountability.
- Investigate suspicious activity. For example, if a user is deleting data from tables, then a
 security administrator can audit all connections to the database and all successful and
 unsuccessful deletions of rows from all tables in the database.
- Notify an auditor of the actions of an unauthorized user. For example, an unauthorized user could be changing or deleting data, or the user has more privileges than expected, which can lead to reassessing user authorizations.
- Support post-incident investigations.
- Monitor and gather data about specific database activities. For example, the database administrator can gather statistics about which tables are being updated, how many logical I/Os are performed, or how many concurrent users connect at peak times.
- Detect problems with an authorization or access control implementation. For example, you can create audit policies that you expect will never generate an audit record because the data is protected in other ways. However, if these policies generate audit records, then you will know the other security controls are not properly implemented.
- Address auditing requirements for compliance. Regulations such as the following have common auditing-related requirements:
 - 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
 - European Union Directive on Privacy and Electronic Communications

26.3 Best Practices for Auditing

You should follow best practices guidelines for auditing.

 As a general rule, design your auditing strategy to collect the amount of information that you need to meet compliance requirements, but focus on activities that cause the greatest security concerns. For example, auditing every table in the database is not practical, but auditing tables with columns that contain sensitive data, such as salaries, is.



With both unified and fine-grained auditing, there are mechanisms you can use to design audit policies that focus on specific activities to audit.

Periodically archive and purge the audit trail data. You can use the DBMS_AUDIT_MGMT package to purge audit records in several different ways. You should regularly review the collected audit records and establish a system for collecting and retaining audit records based on your site's retention policies. In addition to DBMS_AUDIT_MGMT, Oracle Data Safe and Oracle Audit Vault and Database Firewall provide features that enable you manage the archiving and purging of audit trail data.

Related Topics

- Guidelines for Auditing
 Oracle provides guidelines for auditing.
- Purging Audit Trail Records
 The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

26.4 What Is Unified Auditing?

In unified auditing, the unified audit trail captures audit information from a variety of sources.

Unified auditing enables you to capture audit records from the following sources:

- Audit records (including SYS audit records) from unified audit policies and AUDIT settings
- Fine-grained audit records from the DBMS FGA PL/SQL package
- Oracle Database Real Application Security audit records
- Oracle Recovery Manager audit records
- Oracle Database Vault audit records
- Oracle Label Security audit records
- Oracle Data Mining records
- Oracle Data Pump
- Oracle SQL*Loader Direct Load

The unified audit trail, which resides in a read-only table in the AUDSYS schema in the SYSAUX tablespace, makes this information available in a uniform format in the UNIFIED_AUDIT_TRAIL data dictionary view, and is available in both single-instance and Oracle Database Real Application Clusters environments. In addition to the user SYS, users who have been granted the AUDIT_ADMIN and AUDIT_VIEWER roles can query these views. If your users only need to query the views but not create audit policies, then grant them the AUDIT_VIEWER role.

When the database is writeable, audit records are written to the unified audit trail. If the database is not writable, then audit records are written to new format operating system files in the <code>\$ORACLE BASE/audit/\$ORACLE SID</code> directory.



Oracle Database Reference for detailed information about the UNIFIED_AUDIT_TRAIL data dictionary view



26.5 Benefits of the Unified Audit Trail

The benefits of a unified audit trail are many.

For example:

- After unified auditing is enabled, it does not depend on the initialization parameters that
 were used in previous releases. See Table G-1 for a list of these initialization parameters.
- The audit records, including records from the SYS audit trail, for all the audited components of your Oracle Database installation are placed in one location and in one format, rather than your having to look in different places to find audit trails in varying formats. This consolidated view enables auditors to co-relate audit information from different components. For example, if an error occurred during an INSERT statement, standard auditing can indicate the error number and the SQL that was executed. Oracle Database Vault-specific information can indicate whether this error happened because of a command rule violation or realm violation. Note that there will be two audit records with a distinct AUDIT_TYPE. With this unification in place, SYS audit records appear with AUDIT_TYPE set to Standard Audit.
- The management and security of the audit trail is also improved by having it in single audit trail.
- Overall auditing performance is greatly improved. By default, the audit records are automatically written to an internal relational table in the AUDSYS schema.
- You can create named audit policies that enable you to audit the supported components listed at the beginning of this section, as well as SYS administrative users. Furthermore, you can build conditions and exclusions into your policies.
- If you are using an Oracle Audit Vault and Database Firewall environment, then the unified audit trail greatly facilitates the collection of audit data, because all of this data will come from one location.



In previous releases, users were allowed to add and remove audit configuration to objects in their own schemas without any additional privileges. This ability is no longer allowed.

26.6 Checking if Your Database Has Migrated to Pure Unified Auditing

The V\$OPTION dynamic view indicates if your database has been migrated to using pure unified auditing (that is, traditional auditing is turned off).

 Query the VALUE column of the V\$OPTION dynamic view as follows, entering Unified Auditing in the case shown:



This output shows that unified auditing is enabled and traditional auditing is disabled. If pure unified auditing has not been enabled, then the output is FALSE, which implies that your database is using mixed mode auditing. Mixed mode auditing means that both traditional auditing and unified auditing are present.

Related Topics

Oracle Database Upgrade Guide

26.7 Mixed Mode Auditing

Mixed mode auditing is the default auditing in a newly installed database.

- About Mixed Mode Auditing
 Mixed mode auditing enables both traditional (that is, the audit facility from releases earlier
 than release 12c) and the new audit facilities (unified auditing).
- Enablement of Unified Auditing
 By default and depending on the edition of Oracle Database, Oracle Database uses mixed mode auditing, supporting both unified audit and traditional audit.
- How Database Creation Determines the Type of Auditing You Have Enabled
 Unified auditing uses the \$ORACLE_BASE/audit directory as the location for the operating
 system files.
- Capabilities of Mixed Mode Auditing
 Mixed mode auditing provides several capabilities.

26.7.1 About Mixed Mode Auditing

Mixed mode auditing enables both traditional (that is, the audit facility from releases earlier than release 12c) and the new audit facilities (unified auditing).

When you create a new database, by default the database uses mixed mode auditing.

You can enable the database in either of these two modes: the mixed mode auditing or pure unified auditing mode. Even though the features of unified auditing are enabled in both these modes, there are differences between them. In mixed mode, you can use the new unified audit facility alongside the traditional auditing facility. In pure unified auditing, you only use the unified audit facility.

Table 26-1 summarizes the features of these two modes and how you enable them.

Table 26-1 Differences Between Mixed Mode Audting and Pure Unified Auditing

Mode	Features	How to Enable
Mixed mode auditing	Has both traditional and unified auditing	Ensure that at least one unified audit policy is enabled in your database. Typically you will see the ORA_SECURECONFIG and ORA_LOGON_FAILURES policies enabled by default.



Table 26-1 (Cont.) Differences Between Mixed Mode Audting and Pure Unified Auditing

Mode	Features	How to Enable
Pure unified auditing	Has only unified auditing; traditional auditing is turned off	Link the oracle binary with uniaud_on, and then restart the database. Relinking the binary turns off traditional audit. After you have decided to use pure unified auditing, you can relink the oracle binary with the unified audit option turned on. This turns off traditional auditing.
		Oracle Database Upgrade Guide describes how to enable pure unified auditing.

Mixed mode is intended to introduce unified auditing, so that you can have a feel of how it works and what its nuances and benefits are. Mixed mode enables you to migrate your existing applications and scripts to use unified auditing. Once you have decided to use pure unified auditing, you can relink the oracle binary with the unified audit option turned on and thereby enable it as the one and only audit facility the Oracle database runs. If you decide to revert back to mixed mode, you can.

As in previous releases, the traditional audit facility is driven by the AUDIT_TRAIL initialization parameter. Only for mixed mode auditing, you should set this parameter to the appropriate traditional audit trail. This traditional audit trail will then be populated with audit records, along with the unified audit trail.

When you upgrade your database to the current release, traditional auditing is preserved, and the new audit records are written to the traditional audit trail. After you complete the migration, the audit records from the previous release are still available in those audit trails. You then can archive and purge these older audit trails by using the DBMS_AUDIT_MGMT PL/SQL procedures, based on your enterprise retention policies.

See Also:

- How the Unified Auditing Migration Affects Individual Audit Features, for a comparison of the features available in the pre-migrated and post-migrated auditing environments
- Checking if Your Database Has Migrated to Pure Unified Auditing
- Oracle Database Upgrade Guide for information about migrating your databases to unified auditing, and for references to the documentation you should use if you choose not to migrate

26.7.2 Enablement of Unified Auditing

By default and depending on the edition of Oracle Database, Oracle Database uses mixed mode auditing, supporting both unified audit and traditional audit.

When you are ready to migrate to pure unified audit mode (which turns off traditional audit and improves audit performance), link the oracle binary with uniaud_on, and then restart the database, as described in *Oracle Database Upgrade Guide*.

Related Topics

• Checking if Your Database Has Migrated to Pure Unified Auditing
The V\$OPTION dynamic view indicates if your database has been migrated to using pure unified auditing (that is, traditional auditing is turned off).

26.7.3 How Database Creation Determines the Type of Auditing You Have Enabled

Unified auditing uses the <code>\$ORACLE_BASE/audit</code> directory as the location for the operating system files.

For newly created databases, mixed mode auditing is enabled by default through the predefined policies ORA SECURECONFIG and ORA LOGON FAILURES.

Ensure that at least one unified audit policy is enabled in your database. Enable the <code>ORA_SECURECONFIG</code> and <code>ORA_LOGON_FAILURES</code> policies if they are not yet enabled. Note that Oracle Database has mandatory audits that cannot be turned off, so auditing of the most common security relevant events will continue to happen even if none of the unified audit policies are enabled.

Related Topics

Secure Options Predefined Unified Audit Policy
 The ORA_SECURECONFIG unified audit policy provides all the secure configuration audit options.

26.7.4 Capabilities of Mixed Mode Auditing

Mixed mode auditing provides several capabilities.

These capabilities are as follows:

- It enables the use of all existing auditing initialization parameters: AUDIT_TRAIL, AUDIT FILE DEST, AUDIT SYS OPERATIONS, and AUDIT SYSLOG LEVEL.
- It writes mandatory audit records only to the traditional audit trails.
- It bases standard audit records on the standard audit configuration, and writes these records to the audit trail designated by the AUDIT TRAIL initialization parameter.
 - However, be aware that standard audit trail records are also generated based on unified audit policies and only these audit records are written to the unified audit trail. The standard audit records generated as a result of unified audit policies follow the semantics of unified audit policy enablement.
- Administrative user sessions generate SYS audit records. These records are written if the
 AUDIT_SYS_OPERATIONS initialization parameter is set to TRUE. This process writes the
 records only to the traditional audit trails. However, when unified audit policies are enabled
 for administrative users, these unified audit records are also written to unified audit trail.
- The format of the audit records that are written to traditional audit trails remains the same as in Oracle Database 11g Release 2.
- Oracle Database immediately writes unified audit records to an internal relational table in the AUDSYS schema.



- The performance cost of writing an audit record is equivalent to the sum of the times required for generating and writing an audit record to the traditional audit trail and the unified audit trail.
- Mixed mode auditing provides a glance of the unified audit mode features. Oracle
 recommends that you migrate to unified audit mode once you are comfortable with the new
 style of audit policies and audit trail.

Related Topics

- Writing the Unified Audit Trail Records to the AUDSYS Schema
 Oracle Database automatically writes audit records to an internal relational table in the AUDSYS schema.
- Oracle Database Upgrade Guide

26.8 Who Can Perform Auditing?

Oracle provides two roles for users who perform auditing: AUDIT ADMIN and AUDIT VIEWER.

The privileges that these roles provide are as follows:

- **AUDIT_ADMIN role.** This 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. Grant this role only to trusted users.
- AUDIT_VIEWER role. This 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.

To change audit policies or modify the audit trail (including purging old audit data), you must be granted the AUDIT_ADMIN role. An auditor can view audit data after being granted the AUDIT VIEWER role.

26.9 Unified Auditing in a Multitenant Environment

You can apply audit settings to individual PDBs or to the CDB, depending on the type of policy. Each PDB, including the root, has its own unified audit trail.

- Unified audit policies created with the CREATE AUDIT POLICY and AUDIT statements: You can create policies for both the root and individual PDBs.
- Audit records written to the syslog: On UNIX platforms, you can set the
 UNIFIED_AUDIT_COMMON_SYSTEMLOG initialization parameter in the CDB root to enable
 certain unified audit trail columns to be written to SYSLOG. On both Windows and UNIX,
 you can set the UNIFIED_AUDIT_SYSTEMLOG parameter in both the root and PDB level.
- Fine-grained audit policies: You can create policies for individual PDBs only, not the root.
- Purging the audit trail: You can perform purge operations for both the root and individual PDBs.

Related Topics

- Unified Audit Policies or AUDIT Settings in a Multitenant Environment
 In a multitenant environment, you can create unified audit policies for individual PDBs and in the root.
- Creating a Fine-Grained Audit Policy
 The DBMS FGA.ADD POLICY procedure creates a fine-grained audit policy.



Purging Audit Trail Records

The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

26.10 Auditing in a Distributed Database

Auditing is site autonomous in that a database instance audits only the statements issued by directly connected users.

A local Oracle Database node cannot audit actions that take place in a remote database.



Configuring Audit Policies

Unified auditing supports custom unified audit policies, predefined unified auditing policies, and fine-grained auditing.

- Selecting an Auditing Type
 - You can audit general activities (such as SQL statement actions), commonly used auditing activities, or fine-grained audit scenarios.
- Auditing Activities with Unified Audit Policies and the AUDIT Statement
 You can use the CREATE AUDIT POLICY and AUDIT statements to use unified auditing policies.
- Auditing Activities with the Predefined Unified Audit Policies
 Oracle Database provides predefined unified audit policies that cover commonly used security-relevant audit settings.
- Auditing Specific Activities with Fine-Grained Auditing
 Fine-grained auditing enables you to create audit policies at the granular level.
- Audit Policy Data Dictionary Views
 Data dictionary and dynamic views can be used to find detailed auditing information.

27.1 Selecting an Auditing Type

You can audit general activities (such as SQL statement actions), commonly used auditing activities, or fine-grained audit scenarios.

- Auditing SQL Statements, Privileges, and Other General Activities
 You can audit many types of objects, from SQL statements to other Oracle Database components, such as Oracle Database Vault..
- Auditing Commonly Used Security-Relevant Activities
 Oracle Database provides a set default unified audit policies that you can choose from for
 commonly used security-relevant audits.
- Auditing Specific, Fine-Grained Activities
 Use fine-grained auditing if you want to audit individual columns and use event handlers.

27.1.1 Auditing SQL Statements, Privileges, and Other General Activities

You can audit many types of objects, from SQL statements to other Oracle Database components, such as Oracle Database Vault..

In addition, you can create policies that use conditions. However, if you want to audit specific columns or use event handlers, you must use fine-grained auditing.

The general steps for performing this type of auditing are as follows:

1. In most cases, use the CREATE AUDIT POLICY statement to create an audit policy. If you must audit application context values, then use the AUDIT statement.

See the relevant categories under Auditing Activities with Unified Audit Policies and the AUDIT Statement.

2. If you are creating an audit policy, then use the AUDIT statement to enable it and optionally apply (or exclude) the audit settings to one or more users, including administrative users who log in with the SYSDBA administrative privilege (for example, the SYS user).

AUDIT also enables you to create an audit record upon an action's success, failure, or both.

See Enabling and Applying Unified Audit Policies to Users and Roles.

3. Query the UNIFIED AUDIT TRAIL view to find the generated audit records.

See also Audit Policy Data Dictionary Views for additional views.

4. Periodically archive and purge the contents of the audit trail.

See Purging Audit Trail Records.

27.1.2 Auditing Commonly Used Security-Relevant Activities

Oracle Database provides a set default unified audit policies that you can choose from for commonly used security-relevant audits.

The general steps for performing this type of auditing are as follows:

- See Auditing Activities with the Predefined Unified Audit Policies to learn about the default audit policies.
- 2. Use the AUDIT statement enable the policy and optionally apply (or exclude) the audit settings to one or more users.

See Enabling and Applying Unified Audit Policies to Users and Roles.

3. Query the UNIFIED_AUDIT_TRAIL view to find the generated audit records.

See also Audit Policy Data Dictionary Views for additional views.

4. Periodically archive and purge the contents of the audit trail.

See Purging Audit Trail Records.

27.1.3 Auditing Specific, Fine-Grained Activities

Use fine-grained auditing if you want to audit individual columns and use event handlers.

This type of auditing provides all the features available in unified audit policies.

The general steps for fine-grained auditing are as follows:

- 1. See Auditing Specific Activities with Fine-Grained Auditing to understand more about auditing specific activities.
- 2. Use the DBMS_FGA PL/SQL package to configure fine-grained auditing policies. See Using the DBMS_FGA PL/SQL Package to Manage Fine-Grained Audit Policies.
- 3. Query the UNIFIED AUDIT TRAIL view to find the generated audit records.

See also Audit Policy Data Dictionary Views for additional views.

4. Periodically archive and purge the contents of the audit trail.

See Purging Audit Trail Records.



27.2 Auditing Activities with Unified Audit Policies and the AUDIT Statement

You can use the CREATE AUDIT POLICY and AUDIT statements to use unified auditing policies.

About Auditing Activities with Unified Audit Policies and AUDIT
 You can audit the several types of activities, using unified audit policies and the AUDIT SQL
 statement.

Best Practices for Creating Custom Unified Audit Policies

You can enable multiple policies at a time in the database, but ideally, limit the number of enabled policies.

Syntax for Creating a Unified Audit Policy

To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

Auditing Roles

You can use the CREATE AUDIT POLICY statement to audit database roles.

Auditing System Privileges

You can use the CREATE AUDIT POLICY statement to audit system privileges.

Auditing Administrative Users

You can create unified audit policies to capture the actions of administrative user accounts, such as sys.

Auditing Object Actions

You can use the CREATE AUDIT POLICY statement to audit object actions.

Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges

The CREATE AUDIT POLICY statement can audit the READ ANY TABLE and SELECT ANY TABLE privileges.

Auditing SQL Statements and Privileges in a Multitier Environment

You can create a unified audit policy to audit the activities of a client in a multitier environment.

Creating a Condition for a Unified Audit Policy

You can use the CREATE AUDIT POLICY statement to create conditions for a unified audit policy.

Auditing Application Context Values

You can use the AUDIT statement to audit application context values.

Auditing Oracle Database Real Application Security Events

You can use CREATE AUDIT POLICY statement to audit Oracle Database Real Application Security events.

Auditing Oracle Recovery Manager Events

You can use the CREATE AUDIT POLICY statement to audit Oracle Recovery Manager events.

Auditing Oracle Database Vault Events

In an Oracle Database Vault environment, the CREATE AUDIT POLICY statement can audit Database Vault activities.

Auditing Oracle Label Security Events

In an Oracle Label Security environment, the CREATE AUDIT POLICY statement can audit Oracle Label Security activities.



Auditing Oracle Data Mining Events

You can use the CREATE AUDIT POLICY statement to audit Oracle Data Mining events.

Auditing Oracle Data Pump Events

You can use the CREATE AUDIT POLICY statement to audit Oracle Data Pump.

Auditing Oracle SQL*Loader Direct Load Path Events

You can use the CREATE AUDIT POLICY statement to audit Oracle SQL*Loader direct load path events.

Auditing Only Top-Level Statements

You can audit top-level SQL or PL/SQL statements to limit the volume of audit records.

Unified Audit Policies or AUDIT Settings in a Multitenant Environment In a multitenant environment, you can create unified audit policies for individual PDBs and in the root.

Altering Unified Audit Policies

You can use the ALTER AUDIT POLICY statement to modify a unified audit policy.

Enabling and Applying Unified Audit Policies to Users and Roles

You can use the AUDIT POLICY statement to enable and apply unified audit policies to users and roles.

Disabling Unified Audit Policies

You can use the NOAUDIT POLICY statement to disable a unified audit policy.

Dropping Unified Audit Policies

You can use the DROP AUDIT POLICY statement to drop a unified audit policy.

Tutorial: Auditing Nondatabase Users

This tutorial shows how to create a unified audit policy that uses a client identifier to audit a nondatabase user's actions.

Related Topics

Auditing SQL Statements, Privileges, and Other General Activities
 You can audit many types of objects, from SQL statements to other Oracle Database components, such as Oracle Database Vault..

27.2.1 About Auditing Activities with Unified Audit Policies and AUDIT

You can audit the several types of activities, using unified audit policies and the ${\tt AUDIT}$ SQL statement.

The kinds of activities that you can audit are as follows:

- User accounts (including administrative users who log in with the SYSDBA administrative privilege), roles, and privileges
- Object actions, such as dropping a table or a running a procedure
- Application context values
- Activities from Oracle Database Real Application Security, Oracle Recovery Manager,
 Oracle Data Mining, Oracle Data Pump, Oracle SQL*Loader direct path events, Oracle
 Database Vault, and Oracle Label Security

To find system actions to audit, you can guery the AUDITABLE SYSTEM ACTIONS system table.

To accomplish this, depending on what you want to audit, use the following:

• Unified audit policies. A unified audit policy is a named group of audit settings that enable you to audit a particular aspect of user behavior in the database. To create the

policy, you use the CREATE AUDIT POLICY statement. The policy can be as simple as auditing the activities of a single user or you can create complex audit policies that use conditions. You can have more than one audit policy in effect at a time in a database. An audit policy can contain both system-wide and object-specific audit options. Most of the auditing that you will do for general activities (including standard auditing) requires the use of audit policies.

- AUDIT and NOAUDIT SQL statements. The AUDIT and NOAUDIT SQL statements enable you to, respectively, enable and disable an audit policy. The AUDIT statement also lets you include or exclude specific users for the policy. The AUDIT and NOAUDIT statements also enable you to audit application context values.
- For Oracle Recovery Manager, you do not create unified audit policies. The UNIFIED_AUDIT_TRAIL view automatically captures commonly audited Recovery Manager events.

27.2.2 Best Practices for Creating Custom Unified Audit Policies

You can enable multiple policies at a time in the database, but ideally, limit the number of enabled policies.

The unified audit policy syntax is designed so that you can write one policy that covers all the audit settings that your database needs. A good practice is to group related options into a single policy instead of creating multiple small policies. This enables you to manage the policies much easier. As an example, each predefined audit policies contains multiple audit settings within one unified audit policy.

Limiting the number of enabled audit policies for a user session has the following benefits:

- It reduces the logon overhead that is associated with loading the audit policy's details into the session's UGA memory. If the enabled policy count is less, then less time is spent in loading the policy information.
- It reduces the session's UGA memory consumption, because a fewer number of policies are required to be cached in UGA memory.
- It makes the internal audit check functionality more efficient, which determines whether to generate an audit record for its associated event.
- If you have configured a unified audit policy for LOGON statements, then audit records for both direct logins as well as ALTER SESSION and SET CONTAINER statements are generated.

Related Topics

Auditing Activities with the Predefined Unified Audit Policies
 Oracle Database provides predefined unified audit policies that cover commonly used security-relevant audit settings.

27.2.3 Syntax for Creating a Unified Audit Policy

To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

When you create a unified audit policy, Oracle Database stores it in a first class object that is owned by the SYS schema, not in the schema of the user who created the policy.

Example 27-1 shows the syntax for the CREATE AUDIT POLICY statement.



Example 27-1 Syntax for the CREATE AUDIT POLICY Statement

In this specification:

privilege_audit_clause describes privilege-related audit options. See Auditing System
 Privileges for details. The detailed syntax for configuring privilege audit options is as
 follows:

```
privilege audit clause := PRIVILEGES privilege1 [, privilege2]
```

action_audit_clause and standard_actions describe object action-related audit options.
 See Auditing Object Actions. The syntax is as follows:

component_actions enables you to create an audit policy for Oracle Label Security, Oracle
Database Real Application Security, Oracle Database Vault, Oracle Data Pump, or Oracle
SQL*Loader. See the appropriate section under Auditing Activities with Unified Audit
Policies and the AUDIT Statement for more information. The syntax is:

```
component_actions :=
   ACTIONS COMPONENT=[OLS|XS] action1 [,action2 ] |
   ACTIONS COMPONENT=DV DV_action ON DV_object_name |
   ACTIONS COMPONENT=DATAPUMP [ EXPORT | IMPORT | ALL ] |
   ACTIONS COMPONENT=DIRECT_LOAD [ LOAD | ALL ]
```

role audit clause enables you to audit roles. See Auditing Roles. The syntax is:

```
role audit clause := ROLES role1 [, role2]
```

WHEN audit_condition EVALUATE PER enables you to specify a function to create a
condition for the audit policy and the evaluation frequency. You must include the EVALUATE
PER clause with the WHEN condition. See Creating a Condition for a Unified Audit Policy. The
syntax is:

```
WHEN 'audit_condition := function operation value_list'
EVALUATE PER {STATEMENT|SESSION|INSTANCE}
```

 ONLY TOPLEVEL allows users to audit only the top-level operations that are performed for the actions that were configured as part of this audit policy. See Auditing Only Top-Level Statements. CONTAINER, allows users to audit only the top-level operations that were performed for the
actions that were configured as part of this audit policy. See Unified Audit Policies or
AUDIT Settings in a Multitenant Environment.

This syntax is designed to audit any of the components listed in the policy. For example, suppose you create the following policy:

```
CREATE AUDIT POLICY table_pol
PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE
ROLES emp admin, sales admin;
```

The audit trail will capture SQL statements that require the CREATE ANY TABLE system privilege or the DROP ANY TABLE system privilege or any system privilege directly granted to the role emp_admin or any system privilege directly granted to the role sales_admin. (Be aware that it audits privileges that are *directly* granted, not privileges that are granted recursively through a role.)

After you create the policy, you must enable it by using the AUDIT statement. Optionally, you can apply the policy to one or more users, exclude one or more users from the policy, and designate whether an audit record is written when the audited action succeeds, fails, or both succeeds or fails. See Enabling and Applying Unified Audit Policies to Users and Roles.

27.2.4 Auditing Roles

You can use the CREATE AUDIT POLICY statement to audit database roles.

- About Role Auditing
 When you audit a role, Oracle Database audits all system privileges that are directly granted to the role.
- Configuring Role Unified Audit Policies
 To create a unified audit policy to capture role use, you must include the ROLES clause in the CREATE AUDIT POLICY statement.
- Example: Auditing the DBA Role in a Multitenant Environment
 The CREATE AUDIT POLICY statement can audit roles in a multitenant environment.

27.2.4.1 About Role Auditing

When you audit a role, Oracle Database audits all system privileges that are directly granted to the role.

You can audit any role, including user-defined roles. If you create a common unified audit policy for roles with the ROLES audit option, then you must specify only common roles in the role list. When such a policy is enabled, Oracle Database audits all system privileges that are commonly and directly granted to the common role. The system privileges that are locally granted to the common role will not be audited. To find if a role was commonly granted, query the DBA_ROLES data dictionary view. To find if the privileges granted to the role were commonly granted, query the ROLE SYS PRIVS view.

Related Topics

Predefined Roles in an Oracle Database Installation
 Oracle Database provides a set of predefined roles to help in database administration.



27.2.4.2 Configuring Role Unified Audit Policies

To create a unified audit policy to capture role use, you must include the ROLES clause in the CREATE AUDIT POLICY statement.

Use the following syntax to create a unified audit policy that audits roles:

```
CREATE AUDIT POLICY policy_name
ROLES role1 [, role2];
```

For example:

```
CREATE AUDIT POLICY audit_roles_pol ROLES IMP FULL DATABASE; EXP FULL DATABASE;
```

You can build more complex role unified audit policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.4.3 Example: Auditing the DBA Role in a Multitenant Environment

The CREATE AUDIT POLICY statement can audit roles in a multitenant environment.

The following example shows how to audit a predefined common role DBA in a multitenant environment.

Example 27-2 Auditing the DBA Role in a Multitenant Environment

```
CREATE AUDIT POLICY role_dba_audit_pol
ROLES DBA
CONTAINER = ALL;
AUDIT POLICY role dba audit pol;
```

27.2.5 Auditing System Privileges

You can use the CREATE AUDIT POLICY statement to audit system privileges.

- About System Privilege Auditing
 System privilege auditing audits activities that successfully use a system privilege, such as
 READ ANY TABLE.
- System Privileges That Can Be Audited
 You can audit the use of almost any system privilege.
- System Privileges That Cannot Be Audited Several system privileges cannot be audited.
- Configuring a Unified Audit Policy to Capture System Privilege Use
 The PRIVILEGES clause in the CREATE AUDIT POLICY statement audits system privilege use.
- Example: Auditing a User Who Has ANY Privileges
 The CREATE AUDIT POLICY statement can audit users for ANY privileges.

- Example: Using a Condition to Audit a System Privilege
 The CREATE AUDIT POLICY statement can create an audit policy that uses a condition to audit a system privilege.
- How System Privilege Unified Audit Policies Appear in the Audit Trail
 The UNIFIED AUDIT TRAIL data dictionary view lists system privilege audit events.

27.2.5.1 About System Privilege Auditing

System privilege auditing audits activities that successfully use a system privilege, such as READ ANY TABLE.

In this kind of auditing, SQL statements that require the audited privilege to succeed are recorded.

A single unified audit policy can contain both privilege and action audit options. Do not audit the privilege use of administrative users such as SYS. Instead, audit their object actions.

Be aware that a single query can generate multiple audit records, one for each object that is being accessed in the query and if auditing is enabled on all these objects. For example, while you query the view, multiple audit records can be generated for each of the underlying objects that is referenced by the view itself.



You can audit system privileges, objects, database events, and so on. However, if you must find database privilege usage (for example, which privileges that have been granted to a given role are used), and generate a report of the used and unused privileges, then you can create a privilege capture.

Related Topics

- Auditing Object Actions
 You can use the CREATE AUDIT POLICY statement to audit object actions.
- Performing Privilege Analysis to Identify Privilege Use
 Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.

27.2.5.2 System Privileges That Can Be Audited

You can audit the use of almost any system privilege.

To find a list of auditable system privileges, you can query the SYSTEM PRIVILEGE MAP table.

For example:



Similar to action audit options, privilege auditing audits the use of system privileges that have been granted to database users. If you set similar audit options for both SQL statement and privilege auditing, then only a single audit record is generated. For example, if two policies exist, with one auditing EXECUTE PROCEDURE specifically on the HR.PROC procedure and the second auditing EXECUTE PROCEDURE in general (all procedures), then only one audit record is written.

Privilege auditing does not occur if the action is already permitted by the existing owner and object privileges. Privilege auditing is triggered only if the privileges are insufficient, that is, only if what makes the action possible is a system privilege. For example, suppose that user SCOTT has been granted the SELECT ANY TABLE privilege and SELECT ANY TABLE is being audited. If SCOTT selects his own table (for example, SCOTT.EMP), then the SELECT ANY TABLE privilege is not used. Because he performed the SELECT statement within his own schema, no audit record is generated. On the other hand, if SCOTT selects from another schema (for example, the HR.EMPLOYEES table), then an audit record is generated. Because SCOTT selected a table outside his own schema, he needed to use the SELECT ANY TABLE privilege.

27.2.5.3 System Privileges That Cannot Be Audited

Several system privileges cannot be audited.

These privileges are:

- INHERIT ANY PRIVILEGE
- INHERIT PRIVILEGE
- TRANSLATE ANY SQL
- TRANSLATE SOL

27.2.5.4 Configuring a Unified Audit Policy to Capture System Privilege Use

The PRIVILEGES clause in the CREATE AUDIT POLICY statement audits system privilege use.

Use the following syntax to create a unified audit policy that audits privileges:

```
CREATE AUDIT POLICY policy_name
  PRIVILEGES privilege1 [, privilege2];
```

For example:

```
CREATE AUDIT POLICY my_simple_priv_policy PRIVILEGES SELECT ANY TABLE, CREATE LIBRARY;
```

You can build more complex privilege unified audit policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.5.5 Example: Auditing a User Who Has ANY Privileges

The CREATE AUDIT POLICY statement can audit users for any privileges.

Example 27-3 shows how to audit several ANY privileges of the user HR MGR.



Example 27-3 Auditing a User Who Has ANY Privileges

```
CREATE AUDIT POLICY hr_mgr_audit_pol
PRIVILEGES DROP ANY TABLE, DROP ANY CONTEXT, DROP ANY INDEX, DROP ANY LIBRARY;
AUDIT POLICY hr mgr audit pol BY HR MGR;
```

27.2.5.6 Example: Using a Condition to Audit a System Privilege

The CREATE AUDIT POLICY statement can create an audit policy that uses a condition to audit a system privilege.

Example 27-4 shows how to use a condition to audit privileges that are used by two operating system users, psmith and jrawlins.

Example 27-4 Using a Condition to Audit a System Privilege

```
CREATE AUDIT POLICY os_users_priv_pol
PRIVILEGES SELECT ANY TABLE, CREATE LIBRARY
WHEN 'SYS_CONTEXT (''USERENV'', ''OS_USER'') IN (''psmith'', ''jrawlins'')'
EVALUATE PER SESSION;

AUDIT POLICY os_users_priv_pol;
```

27.2.5.7 How System Privilege Unified Audit Policies Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists system privilege audit events.

The following example, based on the unified audit policy os_users_priv_pol that was created in Example 27-4, shows a list of privileges used by the operating system user psmith.

```
SELECT SYSTEM_PRIVILEGE_USED FROM UNIFIED_AUDIT_TRAIL

WHERE OS_USERNAME = 'PSMITH' AND UNIFIED_AUDIT_POLICIES = 'OS_USERS_PRIV_POL';

SYSTEM_PRIVILEGE_USED

SELECT ANY TABLE

DROP ANY TABLE
```



If you have created an audit policy for the SELECT ANY TABLE system privilege, whether the user has exercised the READ object privilege or the SELECT object privilege will affect the actions that the audit trail captures.

Related Topics

Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges
 The CREATE AUDIT POLICY statement can audit the READ ANY TABLE and SELECT ANY TABLE privileges.

27.2.6 Auditing Administrative Users

You can create unified audit policies to capture the actions of administrative user accounts, such as SYS.

- Administrative User Accounts That Can Be Audited
 Oracle Database provides administrative user accounts that are associated with administrative privileges.
- Configuring a Unified Audit Policy to Capture Administrator Activities
 The CREATE AUDIT POLICY statement can audit administrative users.
- Example: Auditing the SYS User
 The CREATE AUDIT POLICY statement can audit the SYS user.

27.2.6.1 Administrative User Accounts That Can Be Audited

Oracle Database provides administrative user accounts that are associated with administrative privileges.

Table 27-1 lists default administrative user accounts and the administrative privileges with which they are typically associated.

Table 27-1 Administrative Users and Administrative Privileges

Administrative User Account	Administrative Privilege
SYS	SYSDBA
PUBLIC ¹	SYSOPER
SYSASM	SYSASM
SYSBACKUP	SYSBACKUP
SYSDG	SYSDG
SYSKM	SYSKM

PUBLIC refers to the user PUBLIC, which is the effective user when you log in with the SYSOPER administrative privilege. It does not refer to the PUBLIC role.

Related Topics

Activities That Are Mandatorily Audited
 Certain security sensitive database activities are always audited and such audit configuration cannot be disabled.

27.2.6.2 Configuring a Unified Audit Policy to Capture Administrator Activities

The CREATE AUDIT POLICY statement can audit administrative users.

To audit administrative users, create a unified audit policy and then apply this policy to the
user, the same as you would for non-administrative users. Note that top-level statements
by administrative users are mandatorily audited until the database opens.

27.2.6.3 Example: Auditing the SYS User

The CREATE AUDIT POLICY statement can audit the SYS user.

Example 27-5 shows how to audit grants of the DBMS FGA PL/SQL package by user SYS.

Example 27-5 Auditing the SYS User

CREATE AUDIT POLICY dbms_fga_grants ACTIONS GRANT



ON DBMS FGA;

AUDIT POLICY dbms_fga_grants BY SYS;

27.2.7 Auditing Object Actions

You can use the CREATE AUDIT POLICY statement to audit object actions.

About Auditing Object Actions

You can audit actions performed on specific objects, such as $\tt UPDATE$ statements on the $\tt HR.EMPLOYEES$ table.

Object Actions That Can Be Audited

Auditing object actions can be broad or focused (for example, auditing all user actions or only a select list of user actions).

Configuring an Object Action Unified Audit Policy

The ACTIONS clause in the CREATE AUDIT POLICY statement creates a policy that captures object actions.

Example: Auditing Actions on SYS Objects

The CREATE AUDIT POLICY statement can audit actions on SYS objects.

Example: Auditing Multiple Actions on One Object

The CREATE AUDIT POLICY statement can audit multiple actions on one object.

• Example: Auditing GRANT and REVOKE Operations on an Object

The CREATE AUDIT POLICY statement can audit GRANT and REVOKE operations on objects, such as tables.

Example: Auditing Both Actions and Privileges on an Object

The CREATE AUDIT POLICY statement can audit both actions and privileges on an object, using a single policy.

Example: Auditing All Actions on a Table

The CREATE AUDIT POLICY statement can audit all actions on a table.

• Example: Auditing All Actions in the Database

The CREATE AUDIT POLICY statement can audit all actions in the database.

How Object Action Unified Audit Policies Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists object action audit events.

· Auditing Functions, Procedures, Packages, and Triggers

You can audit functions, procedures, PL/SQL packages, and triggers.

Auditing of Oracle Virtual Private Database Predicates

The unified audit trail automatically captures the predicates that are used in Oracle Virtual Private Database (VPD) policies.

Audit Policies for Oracle Virtual Private Database Policy Functions

Auditing can affect dynamic VPD policies, static VPD policies, and context-sensitive VPD policies.

Unified Auditing with Editioned Objects

When an editioned object has a unified audit policy, it applies in all editions in which the object is visible.

27.2.7.1 About Auditing Object Actions

You can audit actions performed on specific objects, such as UPDATE statements on the HR.EMPLOYEES table.



The audit can include both DDL and DML statements that were used on the object. A single unified audit policy can contain both privilege and action audit options, as well as audit options set for multiple objects.

27.2.7.2 Object Actions That Can Be Audited

Auditing object actions can be broad or focused (for example, auditing all user actions or only a select list of user actions).

Table 27-2 lists the object-level standard database action options. Audit policies for the SELECT SQL statement will capture READ actions as well as SELECT actions.

Table 27-2 Object-Level Standard Database Action Audit Option

Object	SQL Action That Can Be Audited
Table	ALTER, AUDIT, COMMENT, DELETE, FLASHBACK, GRANT, INDEX, INSERT, LOCK, RENAME, SELECT, UPDATE
View	AUDIT, COMMENT, DELETE, FLASHBACK, GRANT, INSERT, LOCK, RENAME, SELECT, UPDATE
Sequence	ALTER, AUDIT, GRANT, SELECT
Procedure (including triggers)	AUDIT, EXECUTE, GRANT
Function	AUDIT, EXECUTE, GRANT
Package	AUDIT, EXECUTE, GRANT
Materialized views	ALTER, AUDIT, COMMENT, DELETE, INDEX, INSERT, LOCK, SELECT, UPDATE
Mining Model	AUDIT, COMMENT, GRANT, RENAME, SELECT
Directory	AUDIT, GRANT, READ
Library	EXECUTE, GRANT
Object type	ALTER, AUDIT, GRANT
Java schema objects (source, class, resource)	AUDIT, EXECUTE, GRANT

Related Topics

- Auditing Functions, Procedures, Packages, and Triggers
 You can audit functions, procedures, PL/SQL packages, and triggers.
- Audit Policies for Oracle Virtual Private Database Policy Functions
 Auditing can affect dynamic VPD policies, static VPD policies, and context-sensitive VPD policies.

27.2.7.3 Configuring an Object Action Unified Audit Policy

The ACTIONS clause in the CREATE AUDIT POLICY statement creates a policy that captures object actions.

Use the following syntax to create a unified audit policy that audits object actions:

```
CREATE AUDIT POLICY policy_name
ACTIONS action1 [, action2 ON object1] [, action3 ON object2];
```

For example:



```
CREATE AUDIT POLICY my_simple_obj_policy
ACTIONS SELECT ON OE.ORDERS, UPDATE ON HR.EMPLOYEES;
```

Note that you can audit multiple actions on multiple objects, as shown in this example.

You can build complex object action unified audit policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.7.4 Example: Auditing Actions on SYS Objects

The CREATE AUDIT POLICY statement can audit actions on SYS objects.

Example 27-6 shows how to create an audit policy that audits SELECT statements on the SYS.USER\$ system table. The audit policy applies to all users, including SYS and SYSTEM.

Example 27-6 Auditing Actions on SYS Objects

```
CREATE AUDIT POLICY select_user_dictionary_table_pol ACTIONS SELECT ON SYS.USER$;

AUDIT POLICY select_user_dictionary_table_pol;
```

27.2.7.5 Example: Auditing Multiple Actions on One Object

The CREATE AUDIT POLICY statement can audit multiple actions on one object.

Example 27-7 shows how to audit multiple SQL statements performed by users <code>jrandolph</code> and <code>phawkins</code> on the <code>app_lib</code> library.

Example 27-7 Auditing Multiple Actions on One Object

```
CREATE AUDIT POLICY actions_on_hr_emp_pol1
ACTIONS EXECUTE, GRANT
ON app_lib;

AUDIT POLICY actions_on_hr_emp_pol1 BY jrandolph, phawkins;
```

27.2.7.6 Example: Auditing GRANT and REVOKE Operations on an Object

The CREATE AUDIT POLICY statement can audit GRANT and REVOKE operations on objects, such as tables.

Enabling auditing on GRANT operations on an object automatically enables the audit of REVOKE operations on the object as well.

Example 27-8 Auditing GRANT and REVOKE Operations

```
CREATE AUDIT POLICY grant_revoke_pol
ACTIONS GRANT ON HR.EMPLOYEES;

AUDIT POLICY grant_revoke_pol;
```

To select from the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view for this type of policy, you can perform a query similar to the following. The grantee name (to whom the privilege is granted) is recorded in the <code>TARGET_USER</code> column.

```
SELECT DBUSERNAME, OBJECT_PRIVILEGES, ACTION_NAME, OBJECT_SCHEMA, OBJECT_NAME, TARGET_USER
FROM UNIFIED_AUDIT_TRAIL
WHERE ACTION NAME IN ('GRANT', 'REVOKE');
```

27.2.7.7 Example: Auditing Both Actions and Privileges on an Object

The CREATE AUDIT POLICY statement can audit both actions and privileges on an object, using a single policy.

Example 27-9 shows a variation of Example 27-7, in which all EXECUTE and GRANT statements on the app lib library using the CREATE LIBRARY privilege are audited.

Example 27-9 Auditing Both Actions and Privileges on an Object

```
CREATE AUDIT POLICY actions_on_hr_emp_pol2
PRIVILEGES CREATE LIBRARY
ACTIONS EXECUTE, GRANT
ON app_lib;

AUDIT POLICY actions on hr emp pol2 BY jrandolph, phawkins;
```

You can audit directory objects. For example, suppose you create a directory object that contains a preprocessor program that the <code>ORACLE_LOADER</code> access driver will use. You can audit anyone who runs this program within this directory object.

27.2.7.8 Example: Auditing All Actions on a Table

The CREATE AUDIT POLICY statement can audit all actions on a table.

You can use the ALL keyword to audit all actions. Oracle recommends that you audit all actions only on sensitive objects. ALL is useful in that it captures indirect SELECT operations. Example 27-10 shows how to audit all actions on the HR.EMPLOYEES table, except actions by user pmulligan.

Example 27-10 Auditing All Actions on a Table

```
CREATE AUDIT POLICY all_actions_on_hr_emp_pol
ACTIONS ALL ON HR.EMPLOYEES;

AUDIT POLICY all_actions_on_hr_emp_pol EXCEPT pmulligan;
```

Related Topics

Example: Auditing All Actions in the Database

The CREATE AUDIT POLICY statement can audit all actions in the database.

27.2.7.9 Example: Auditing All Actions in the Database

The CREATE AUDIT POLICY statement can audit all actions in the database.

To prevent the scenario of a large number of audit records being generated and quickly filling up the audit trail, even for all the recursive actions for this audit policy configuration, include the ONLY TOPLEVEL clause in the CREATE AUDIT POLICY statement. As an alternative to ONLY TOPLEVEL, you can create the ACTIONS ALL policy using a condition to so that only a subset of records are captured.



Use ACTIONS ALL auditing with caution. Do not enable it for users who must perform online transaction processing (OLTP) workloads. This will avoid generating a large number of audit records.

Example 27-11 shows how to audit all actions in the entire database.

Example 27-11 Auditing All Actions in the Database

```
CREATE AUDIT POLICY all_actions_pol ACTIONS ALL ONLY TOPLEVEL;

AUDIT POLICY all actions pol;
```

Related Topics

Creating a Condition for a Unified Audit Policy
 You can use the CREATE AUDIT POLICY statement to create conditions for a unified audit policy.

27.2.7.10 How Object Action Unified Audit Policies Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists object action audit events.

For example:

27.2.7.11 Auditing Functions, Procedures, Packages, and Triggers

You can audit functions, procedures, PL/SQL packages, and triggers.

The areas that you can audit are as follows:

- You can individually audit standalone functions, standalone procedures, and PL/SQL packages.
- If you audit a PL/SQL package, Oracle Database audits all functions and procedures within the package.
- If you enable auditing for all executions, Oracle Database audits all triggers in the database, as well as all the functions and procedures within PL/SQL packages.
- You cannot audit individual functions or procedures within a PL/SQL package.
- When you audit the EXECUTE operation on a PL/SQL stored procedure or stored function, the database considers only its ability to find the procedure or function and authorize its execution when determining the success or failure of the operation for the purposes of auditing. Therefore, if you specify the WHENEVER NOT SUCCESSFUL clause, then only invalid object errors, non-existent object errors, and authorization failures are audited; errors encountered during the execution of the procedure or function are not audited. If you specify the WHENEVER SUCCESSFUL clause, then all executions that are not blocked by invalid object errors, non-existent object errors, or authorization failures are audited, regardless of whether errors are encountered during execution.

27.2.7.12 Auditing of Oracle Virtual Private Database Predicates

The unified audit trail automatically captures the predicates that are used in Oracle Virtual Private Database (VPD) policies.

You do not need to create a unified audit policy to capture the VPD predicate audit information.

This type of audit enables you to identify the predicate expression that was run as part of a DML operation and thereby help you to identify other actions that may have occurred as part of the DML operation. For example, if a malicious attack on your database is performed using a VPD predicate, then you can track the attack by using the unified audit trail. In addition to predicates from user-created VPD policies, the internal predicates from Oracle Label Security and Oracle Real Application Security policies are captured as well. For example, Oracle Label Security internally creates a VPD policy while applying an OLS policy to a table. Oracle Real Application Security generates a VPD policy while enabling an Oracle RAS policy.

The unified audit trail writes this predicate information to the RLS_INFO column of the UNIFIED_AUDIT_TRAIL data dictionary view. If you have fine-grained audit policies, then the RLS INFO column of these views captures VPD predicate information as well.

The audit trail can capture the predicates and their corresponding policy names if multiple VPD policies are enforced on the object. The audit trail captures the policy schema and policy name to enable you to differentiate predicates that are generated from different policies. By default, this information is concatenated in the RLS_INFO column, but Oracle Database provides a function in the DBMS_AUDIT_UTIL PL/SQL package that enables you to reformat the results in an easy-to-read format.

The following example shows how you can audit the predicates of a VPD policy:

1. Create the following VPD policy function:

```
CREATE OR REPLACE FUNCTION auth_orders(
   schema_var IN VARCHAR2,
   table_var IN VARCHAR2
)
RETURN VARCHAR2
IS
   return_val VARCHAR2 (400);
BEGIN
   return_val := 'SALES_REP_ID = 159';
   RETURN return_val;
END auth_orders;
//
```

2. Create the following VPD policy:

3. Create and enable the following the unified audit policy:

```
CREATE AUDIT POLICY oe_pol
ACTIONS SELECT ON OE.ORDERS;
AUDIT POLICY oe pol;
```

4. Connect as user OE and query the OE.ORDERS table.

```
CONNECT OE
Enter password: password
SELECT COUNT(*) FROM ORDERS;
```

5. Connect as a user who has been granted the AUDIT_ADMIN role, and then query the UNIFIED AUDIT TRAIL data dictionary view.

```
CONNECT sec_admin
Enter password: password
SELECT RLS INFO FROM UNIFIED AUDIT TRAIL;
```

Output similar to the following should appear:

```
((POLICY_TYPE=[3]'VPD'), (POLICY_SCHEMA=[9]'SEC_ADMIN'),
(POLICY_NAME=[13]'ORDERS_POLICY'), (PREDICATE=[16]'SALES_REP_ID=159'));
```

6. To extract these details and add them to their own columns, run the appropriate function from the DBMS AUDIT UTIL PL/SQL package.

For unified auditing, you must run the <code>DBMS_AUDIT_UTIL.DECODE_RLS_INFO_ATRAIL_UNI</code> function.

For example:

```
SELECT DBUSERNAME, ACTION_NAME, OBJECT_NAME, SQL_TEXT,

RLS_PREDICATE, RLS_POLICY_TYPE, RLS_POLICY_OWNER, RLS_POLICY_NAME
FROM TABLE (DBMS_AUDIT_UTIL.DECODE_RLS_INFO_ATRAIL_UNI
(CURSOR (SELECT * FROM UNIFIED AUDIT TRAIL)));
```

The reformatted audit trail output appears similar to the following:

```
DBUSERNAME ACTION_NAME OBJECT_NAME SQL_TEXT

RLS_PREDICATE RLS_POLICY_TYPE RLS_POLICY_OWNER RLS_POLICY_NAME

OE SELECT ORDERS SELECT COUNT(*) FROM ORDERS

SALES_REP_ID = 159 VPD SEC_ADMIN ORDERS_POLICY
```

See Also:

- Using Oracle Virtual Private Database to Control Data Access for more information about Oracle Virtual Private Database
- Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS AUDIT UTIL PL/SQL package

27.2.7.13 Audit Policies for Oracle Virtual Private Database Policy Functions

Auditing can affect dynamic VPD policies, static VPD policies, and context-sensitive VPD policies.

- Dynamic policies: Oracle Database evaluates the policy function twice, once during SQL statement parsing and again during execution. As a result, two audit records are generated for each evaluation.
- Static policies: Oracle Database evaluates the policy function once and then caches it in the SGA. As a result, only one audit record is generated.
- Context-sensitive policies: Oracle Database executes the policy function once, during statement parsing. As a result, only one audit record is generated.

27.2.7.14 Unified Auditing with Editioned Objects

When an editioned object has a unified audit policy, it applies in all editions in which the object is visible.

When an editioned object is actualized, any unified audit policies that are attached to it are newly attached to the new actual occurrence. When you newly apply a unified audit policy to an inherited editioned object, this action will actualize it.

You can find the editions in which audited objects appear by querying the <code>OBJECT_NAME</code> and <code>OBJ_EDITION_NAME</code> columns in the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view.

See Also:

Oracle Database Development Guide for detailed information about editions

27.2.8 Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges

The CREATE AUDIT POLICY statement can audit the READ ANY TABLE and SELECT ANY TABLE privileges.

- About Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges
 You can create unified audit policies that capture the use of the READ ANY TABLE and
 SELECT ANY TABLE system privileges.
- Creating a Unified Audit Policy to Capture READ Object Privilege Operations
 You can create unified audit policies that capture READ object privilege operations.

27.2.8.1 About Auditing the READ ANY TABLE and SELECT ANY TABLE Privileges

You can create unified audit policies that capture the use of the READ ANY TABLE and SELECT ANY TABLE system privileges.

Based on the action that the user tried to perform and the privilege that was granted to the user, the <code>SYSTEM_PRIVILEGE_USED</code> column of the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view will record either the <code>READ ANY TABLE</code> system privilege or the <code>SELECT ANY TABLE</code> system privilege. For example, suppose the user has been granted the <code>SELECT ANY TABLE</code> privilege and then performs a query on a table. The audit trail will record that the user used the <code>SELECT ANY TABLE</code> system privilege. If the user was granted <code>READ ANY TABLE</code> and performed the same query, then the <code>READ ANY TABLE</code> privilege is recorded.



27.2.8.2 Creating a Unified Audit Policy to Capture READ Object Privilege Operations

You can create unified audit policies that capture READ object privilege operations.

• To create a unified audit policy to capture any READ object operations, create the policy for the SELECT statement, not for the READ statement.

For example:

CREATE AUDIT POLICY read_hr_employees ACTIONS SELECT ON HR.EMPLOYEES;

For any SELECT object operations, also create the policy on the SELECT statement, as with other object actions that you can audit.

Related Topics

Auditing Object Actions
 You can use the CREATE AUDIT POLICY statement to audit object actions.

27.2.8.3 How the Unified Audit Trail Captures READ ANY TABLE and SELECT ANY TABLE

The unified audit trail captures SELECT behavior based on whether a user has the READ ANY TABLE or the SELECT ANY TABLE privilege.

Table 27-3 describes how the unified audit trail captures these actions.

Table 27-3 Auditing Behavior for READ ANY TABLE and SELECT ANY TABLE

Statement User Issues	Privilege Granted to User	System Privilege Being Audited	Expected UNIFIED_AUDIT_TRAIL Behavior
SELECT	SELECT ANY TABLE	SELECT ANY TABLE	Record inserted into SYSTEM_PRIVILEGE_USED:
			SELECT ANY TABLE
SELECT	SELECT ANY TABLE	READ ANY TABLE	No record
SELECT	SELECT ANY TABLE	Both SELECT ANY TABLE and READ ANY TABLE	Record inserted into SYSTEM_PRIVILEGE_USED:
			SELECT ANY TABLE
SELECT	SELECT ANY TABLE	Neither SELECT ANY TABLE nor READ ANY TABLE	No record
SELECT	READ ANY TABLE	SELECT ANY TABLE	No record
SELECT	READ ANY TABLE	READ ANY TABLE	Record inserted into SYSTEM_PRIVILEGE_USED:
			READ ANY TABLE
SELECT	READ ANY TABLE	Both SELECT ANY TABLE and READ ANY TABLE	Record inserted into SYSTEM_PRIVILEGE_USED:
			READ ANY TABLE
SELECT	READ ANY TABLE	Neither SELECT ANY TABLE nor READ ANY TABLE	No record



Table 27-3 (Cont.) Auditing Behavior for READ ANY TABLE and SELECT ANY TABLE

SELECT Both SELECT ANY TABLE No record, because READ A was used for access SELECT Both SELECT ANY TABLE SELECT Both SELECT ANY Neither SELECT ANY TABLE No record TABLE and READ ANY TABLE SELECT Neither SELECT ANY TABLE SELECT Neither SELECT ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE No record	_TRAIL
TABLE and READ ANY TABLE SELECT Both SELECT ANY Both SELECT ANY TABLE READ ANY TABLE SELECT Both SELECT ANY Both SELECT ANY TABLE ANY TABLE SELECT Both SELECT ANY TABLE ANY TABLE SELECT Both SELECT ANY Neither SELECT ANY TABLE TABLE and READ ANY TABLE SELECT Neither SELECT ANY TABLE SELECT Neither SELECT ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE SELECT Neither SELECT ANY TABLE No record TABLE SELECT Neither SELECT ANY TABLE No record	NY TABLE
TABLE and READ ANY TABLE SYSTEM_PRIVILEGE_USED: ANY TABLE SELECT Both SELECT ANY Neither SELECT ANY TABLE TABLE and READ ANY TABLE No record No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY READ ANY TABLE No record TABLE nor READ ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY Both SELECT ANY TABLE No record TABLE nor READ ANY TABLE No record TABLE nor READ ANY TABLE No record TABLE No record TABLE nor READ ANY TABLE No record TABLE No record TABLE NO READ ANY TABLE No record	
SELECT Both SELECT ANY Neither SELECT ANY TABLE ANY TABLE No record TABLE and READ nor READ ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY READ ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY Both SELECT ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY Both SELECT ANY TABLE No record TABLE nor READ ANY TABLE No record	
TABLE and READ ANY TABLE Neither SELECT ANY SELECT ANY TABLE No record TABLE nor READ ANY TABLE No record No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY TABLE No record TABLE No record TABLE nor READ ANY TABLE No record TABLE No record TABLE nor READ ANY TABLE No record TABLE nor READ ANY TABLE No record TABLE nor READ ANY TABLE No record	
TABLE nor READ ANY TABLE SELECT Neither SELECT ANY READ ANY TABLE No record TABLE nor READ ANY TABLE SELECT Neither SELECT ANY Both SELECT ANY TABLE No record TABLE nor READ ANY TABLE No record TABLE SELECT Neither SELECT ANY Neither SELECT ANY TABLE No record TABLE nor READ ANY TABLE No record TABLE nor READ ANY Neither SELECT ANY TABLE No record	
TABLE nor READ ANY TABLE SELECT Neither SELECT ANY Both SELECT ANY TABLE No record TABLE nor READ ANY and READ ANY TABLE TABLE SELECT Neither SELECT ANY Neither SELECT ANY TABLE No record TABLE nor READ ANY nor READ ANY TABLE	
TABLE nor READ ANY and READ ANY TABLE TABLE Neither SELECT ANY Neither SELECT ANY TABLE No record TABLE nor READ ANY nor READ ANY TABLE	
TABLE nor READ ANY nor READ ANY TABLE	
SELECT FOR UPDATE SELECT ANY TABLE SELECT ANY TABLE Record inserted into SYSTEM_PRIVILEGE_USED: SELECT ANY TABLE	
SELECT FOR UPDATE SELECT ANY TABLE READ ANY TABLE No record	
SELECT FOR UPDATE SELECT ANY TABLE Both SELECT ANY TABLE Record inserted into and READ ANY TABLE SYSTEM_PRIVILEGE_USED:	
SELECT ANY TABLE	
SELECT FOR UPDATE SELECT ANY TABLE Neither SELECT ANY TABLE No record nor READ ANY TABLE	
SELECT FOR UPDATE READ ANY TABLE SELECT ANY TABLE No record	
SELECT FOR UPDATE READ ANY TABLE READ ANY TABLE No record	
SELECT FOR UPDATE READ ANY TABLE Both SELECT ANY TABLE No record and READ ANY TABLE	
SELECT FOR UPDATE READ ANY TABLE Neither SELECT ANY TABLE No record nor READ ANY TABLE	
SELECT FOR UPDATE Both SELECT ANY SELECT ANY TABLE Record inserted into SYSTEM_PRIVILEGE_USED:	
ANY TABLE SELECT ANY TABLE	



Table 27-3 (Cont.) Auditing Behavior for READ ANY TABLE and SELECT ANY TABLE	Table 27-3 ((Cont.) Auditing	Behavior for READ	ANY TABLE and SELECT ANY TABLE
--	--------------	------------------	-------------------	---------------------------------------

Statement User Issues	Privilege Granted to User	System Privilege Being Audited	Expected UNIFIED_AUDIT_TRAIL Behavior
SELECT FOR UPDATE	Both SELECT ANY TABLE and READ ANY TABLE	READ ANY TABLE	No record, because READ ANY TABLE was used for access
SELECT FOR UPDATE	Both SELECT ANY TABLE and READ	Both SELECT ANY TABLE and READ ANY TABLE	Record inserted into SYSTEM_PRIVILEGE_USED:
	ANY TABLE		SELECT ANY TABLE
SELECT FOR UPDATE	Both SELECT ANY TABLE and READ ANY TABLE	Neither SELECT ANY TABLE nor READ ANY TABLE	No record
SELECT FOR UPDATE	Neither SELECT ANY TABLE nor READ ANY TABLE	SELECT ANY TABLE	No record
SELECT FOR UPDATE	Neither SELECT ANY TABLE nor READ ANY TABLE	READ ANY TABLE	No record
SELECT FOR UPDATE		Both SELECT ANY TABLE and READ ANY TABLE	No record
SELECT FOR UPDATE	Neither SELECT ANY TABLE nor READ ANY TABLE	Neither SELECT ANY TABLE or READ ANY TABLE	No record

27.2.9 Auditing SQL Statements and Privileges in a Multitier Environment

You can create a unified audit policy to audit the activities of a client in a multitier environment.

In a multitier environment, Oracle Database preserves the identity of a client through all tiers. Thus, you can audit actions taken on behalf of the client by a middle-tier application, by using the BY user clause in the AUDIT statement for your policy. The audit applies to all user sessions, including proxy sessions.

The middle tier can also set the user client identity in a database session, enabling the auditing of end-user actions through the middle-tier application. The end-user client identity then shows up in the audit trail.

For example, suppose the proxy user apphr can connect as user jackson. The policy and enablement can be as follows:

CREATE AUDIT POLICY prox_pol ACTIONS LOGON; AUDIT POLICY prox_pol BY jackson;

You can audit user activity in a multitier environment. Once audited, you can verify these activities by querying the UNIFIED AUDIT TRAIL data dictionary view. For example:

SELECT DBUSERNAME, DB_PROXY_USERNAME, PROXY_SESSIONID, ACTION_NAME FROM UNIFIED_AUDIT_TRAIL WHERE DBPROXY USERNAME IS NOT NULL;

Output similar to the following appears:

DBUSERNAME	DBPROXY_USERNAME	PROXY_SESSIONID	ACTION_NAME
JACKSON	APPHR	1214623540	LOGON

Figure 27-1 illustrates how you can audit proxy users by querying the PROXY_SESSIONID, ACTION_NAME, and SESSION_ID columns of the UNIFIED_AUDIT_TRAIL view. In this scenario, both the database user and proxy user accounts are known to the database. Session pooling can be used.

Figure 27-1 Auditing Proxy Users

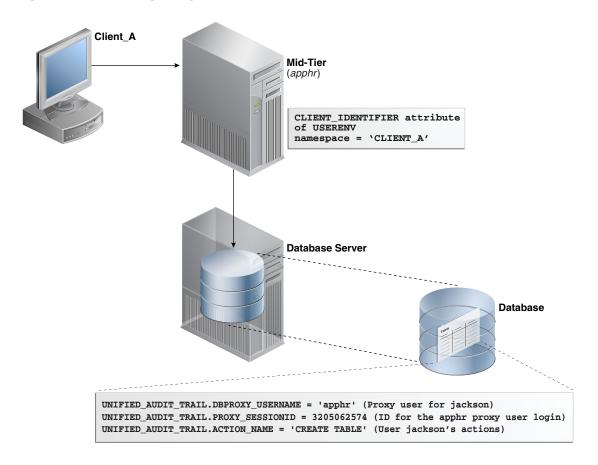


Figure 27-2 illustrates how you can audit client identifier information across multiple database sessions by querying the CLIENT_ID column of the DBA_AUDIT_TRAIL data dictionary view. In this scenario, the client identifier has been set to CLIENT_A. As with the proxy user-database user scenario described in Figure 27-1, session pooling can be used.

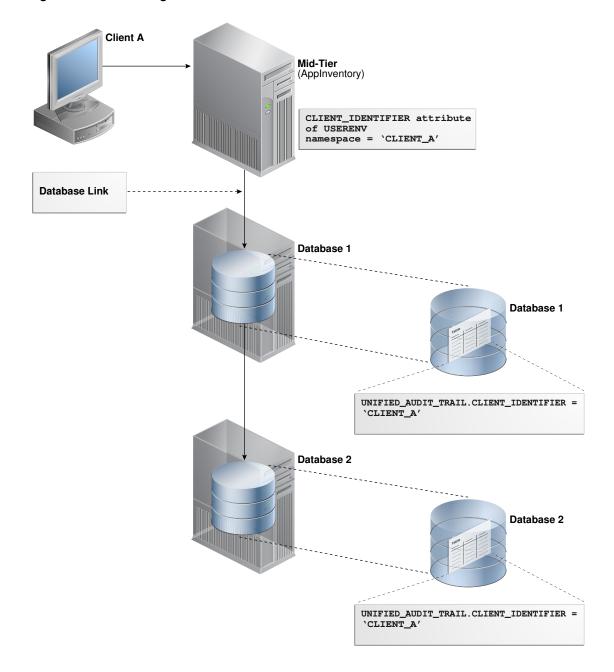


Figure 27-2 Auditing Client Identifier Information Across Sessions

Related Topics

• Preserving User Identity in Multitiered Environments
You can use middle tier servers for proxy authentication and client identifiers to identify application users who are not known to the database.

27.2.10 Creating a Condition for a Unified Audit Policy

You can use the CREATE AUDIT POLICY statement to create conditions for a unified audit policy.

About Conditions in Unified Audit Policies
 You can create a unified audit policy that uses a SYS_CONTEXT namespace-attribute pair to specify a condition.

Configuring a Unified Audit Policy with a Condition

The WHEN clause in the CREATE AUDIT POLICY statement defines the condition in the audit policy.

Example: Auditing Access to SQL*Plus

The CREATE AUDIT POLICY statement can audit access to SQL*Plus.

• Example: Auditing Actions Not in Specific Hosts

The CREATE AUDIT POLICY statement can audit actions that are not in specific hosts.

• Example: Auditing Both a System-Wide and a Schema-Specific Action
The CREATE AUDIT POLICY statement can audit both system-wide and schema-specific actions.

• Example: Auditing a Condition Per Statement Occurrence

The CREATE AUDIT POLICY statement can audit conditions.

- Example: Unified Audit Session ID of a Current Administrative User Session
 The SYS CONTEXT function can be used to find session IDs.
- Example: Unified Audit Session ID of a Current Non-Administrative User Session
 The SYS_CONTEXT function can find the session ID of a current non-administrative user session.
- How Audit Records from Conditions Appear in the Audit Trail
 The audit record conditions from a unified audit policy do not appear in the audit trail.

27.2.10.1 About Conditions in Unified Audit Policies

You can create a unified audit policy that uses a SYS_CONTEXT namespace-attribute pair to specify a condition.

For example, this audit condition can apply to a specific user who may fulfil the audit condition, or a computer host where the audit condition is fulfilled.

If the audit condition is satisfied, then Oracle Database creates an audit record for the event. As part of the condition definition, you must specify whether the audited condition is evaluated per statement occurrence, session, or database instance.



Audit conditions can use both secure and insecure application contexts.

27.2.10.2 Configuring a Unified Audit Policy with a Condition

The WHEN clause in the CREATE AUDIT POLICY statement defines the condition in the audit policy.

Use the following syntax to create a unified audit policy that uses a condition:

```
CREATE AUDIT POLICY policy_name
  action_privilege_role_audit_option
[WHEN function_operation_value_list_1 [[AND | OR] function_operation_value_list_n]
  EVALUATE PER STATEMENT | SESSION | INSTANCE];
```

In this specification:



- action_privilege_role_audit_option refers to audit options for system actions, object actions, privileges, and roles.
- WHEN defines the condition. It has the following components:
 - function uses the following types of functions:

Numeric functions, such as BITAND, CEIL, FLOOR, and LN POWER

Character functions that return character values, such as CONCAT, LOWER, and UPPER

Character functions that return numeric values, such as LENGTH or INSTR

Environment and identifier functions, such as SYS_CONTEXT and UID. For SYS_CONTEXT, in most cases, you may want to use the USERENV namespace, which is described in *Oracle Database SQL Language Reference*.

- operation can be any the following operators: AND, OR, IN, NOT IN, =, <, >,<>
- value list refers to the condition for which you are testing.

You can include additional conditions for each function_operation_value_list set, separated by AND or OR.

When you write the WHEN clause, follow these guidelines:

- Enclose the entire function operation value setting in single quotation marks.
 Within the clause, enclose each quoted component within two pairs of single quotation marks. Do not use double quotation marks.
- Do not exceed 4000 bytes for the WHEN condition.
- EVALUATE PER refers to the following options:
 - STATEMENT evaluates the condition for each relevant auditable statement that occurs.
 - SESSION evaluates the condition only once during the session, and then caches and reuses the result during the remainder of the session. Oracle Database evaluates the condition the first time the policy is used, and then stores the result in UGA memory afterward.
 - INSTANCE evaluates the condition only once during the database instance lifetime. After Oracle Database evaluates the condition, it caches and re-uses the result for the remainder of the instance lifetime. As with the SESSION evaluation, the evaluation takes place the first time it is needed, and then the results are stored in UGA memory afterward.

For example:

```
CREATE AUDIT POLICY oe_orders_pol
ACTIONS UPDATE ON OE.ORDERS
WHEN 'SYS_CONTEXT(''USERENV'', ''IDENTIFICATION_TYPE'') = ''EXTERNAL'''
EVALUATE PER STATEMENT;
```

Remember that after you create the policy, you must use the AUDIT statement to enable it.



Oracle Database SQL Language Reference for more information about functions that you can use in conditions

27.2.10.3 Example: Auditing Access to SQL*Plus

The CREATE AUDIT POLICY statement can audit access to SQL*Plus.

Example 27-12 shows how to audit access to the database with SQL*Plus by users who have been directly granted the roles emp admin and sales admin.

Example 27-12 Auditing Access to SQL*Plus

```
CREATE AUDIT POLICY logon_pol

ACTIONS LOGON

WHEN 'INSTR(UPPER(SYS_CONTEXT(''USERENV'', ''CLIENT_PROGRAM_NAME'')), ''SQLPLUS'') > 0'

EVALUATE PER SESSION;

AUDIT POLICY logon pol BY USERS WITH GRANTED ROLES emp admin, sales admin;
```

27.2.10.4 Example: Auditing Actions Not in Specific Hosts

The CREATE AUDIT POLICY statement can audit actions that are not in specific hosts.

Example 27-13 shows how to audit two actions (UPDATE and DELETE statements) on the OE.ORDERS table, but excludes the host names sales_24 and sales_12 from the audit. It performs the audit on a per session basis and writes audit records for failed attempts only.

Example 27-13 Auditing Actions Not in Specific Hosts

```
CREATE AUDIT POLICY oe_table_audit1

ACTIONS UPDATE ON OE.ORDERS, DELETE ON OE.ORDERS

WHEN 'SYS_CONTEXT (''USERENV'', ''HOST'') NOT IN (''sales_24'',''sales_12'')'

EVALUATE PER SESSION;

AUDIT POLICY oe table audit1 WHENEVER NOT SUCCESSFUL;
```

27.2.10.5 Example: Auditing Both a System-Wide and a Schema-Specific Action

The CREATE AUDIT POLICY statement can audit both system-wide and schema-specific actions.

Example 27-14 shows a variation of Example 27-13 in which the UPDATE statement is audited system wide. The DELETE statement audit is still specific to the OE.ORDERS table.

Example 27-14 Auditing Both a System-Wide and a Schema-Specific Action

```
CREATE AUDIT POLICY oe_table_audit2

ACTIONS UPDATE, DELETE ON OE.ORDERS

WHEN 'SYS_CONTEXT (''USERENV'', ''HOST'') NOT IN (''sales_24'',''sales_12'')'

EVALUATE PER SESSION;

AUDIT POLICY oe table audit2;
```

27.2.10.6 Example: Auditing a Condition Per Statement Occurrence

The CREATE AUDIT POLICY statement can audit conditions.

Example 27-15 shows how to audit a condition based on each occurrence of the DELETE statement on the OE.ORDERS table and exclude user jmartin from the audit.

Example 27-15 Auditing a Condition Per Statement Occurrence

```
CREATE AUDIT POLICY sales_clerk_pol

ACTIONS DELETE ON OE.ORDERS

WHEN 'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'') = ''sales_clerk'''

EVALUATE PER STATEMENT;

AUDIT POLICY sales_clerk_pol EXCEPT jmartin;
```

27.2.10.7 Example: Unified Audit Session ID of a Current Administrative User Session

The SYS CONTEXT function can be used to find session IDs.

Example 27-16 shows how to find the unified audit session ID of current user session for an administrative user.

Example 27-16 Unified Audit Session ID of a Current Administrative User Session

```
CONNECT SYS AS SYSDBA
Enter password: password

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

Output similar to the following appears:

Note that in mixed mode auditing, the <code>UNIFIED_AUDIT_SESSIONID</code> value in the <code>USERENV</code> namespace is different from the value that is recorded by the <code>SESSIONID</code> parameter. Hence, if you are using mixed mode auditing and want to find the correct audit session ID, you should use the <code>USERENV UNIFIED_AUDIT_SESSIONID</code> parameter, not the <code>SESSIONID</code> parameter. In pure unified auditing, the <code>SESSIONID</code> and <code>UNIFIED_AUDIT_SESSIONID</code> values are the same.

27.2.10.8 Example: Unified Audit Session ID of a Current Non-Administrative User Session

The SYS CONTEXT function can find the session ID of a current non-administrative user session.

Example 27-17 shows how to find the unified audit session ID of a current user session for a non-administrative user.

Example 27-17 Unified Audit Session ID of a Current Non-Administrative User Session

```
CONNECT mblake -- Or, CONNECT mblake@hrpdb for a PDB
Enter password: password
SELECT SYS_CONTEXT('USERENV', 'UNIFIED_AUDIT_SESSIONID') FROM DUAL;
```

Output similar to the following appears:



27.2.10.9 How Audit Records from Conditions Appear in the Audit Trail

The audit record conditions from a unified audit policy do not appear in the audit trail.

If the condition evaluates to true and the record is written, then the record appears in the audit trail. You can check the audit trail by querying the UNIFIED AUDIT TRAIL data dictionary view.

Related Topics

Audit Policy Data Dictionary Views
 Data dictionary and dynamic views can be used to find detailed auditing information.

27.2.11 Auditing Application Context Values

You can use the AUDIT statement to audit application context values.

- About Auditing Application Context Values
 You can capture application context values in the unified audit trail.
- Configuring Application Context Audit Settings
 The AUDIT statement with the CONTEXT keyword configures auditing for application context values.
- Disabling Application Context Audit Settings
 The NOAUDIT statement disables application context audit settings.
- Example: Auditing Application Context Values in a Default Database

 The AUDIT CONTEXT NAMESPACE statement can audit application context values.
- Example: Auditing Application Context Values from Oracle Label Security
 The AUDIT CONTEXT NAMESPACE statement can audit application context values from Oracle Label Security.
- How Audited Application Contexts Appear in the Audit Trail
 The UNIFIED_AUDIT_POLICIES data dictionary view lists application context audit events.

27.2.11.1 About Auditing Application Context Values

You can capture application context values in the unified audit trail.

This feature enables you to capture any application context values set by the database applications, while executing the audited statement.

If you plan to audit Oracle Label Security, then this feature captures session label activity for the database audit trail. The audit trail records all the values retrieved for the specified contextattribute value pairs.

The application context audit setting or the audit policy have session static semantics. In other words, if a new policy is enabled for a user, then the subsequent user sessions will see an effect of this command. After the session is established, then the policies and contexts settings are loaded and the subsequent AUDIT statements have no effect on that session.

For multitenant environments, the application context audit policy applies only to the current PDB.



See Also:

- Using Application Contexts to Retrieve User Information, for detailed information about application contexts
- Unified Audit Policies or AUDIT Settings in a Multitenant Environment
- Oracle Label Security Administrator's Guide for detailed information about Oracle Label Security

27.2.11.2 Configuring Application Context Audit Settings

The AUDIT statement with the CONTEXT keyword configures auditing for application context values.

You do not create an unified audit policy for this type of auditing.

Use the following syntax to configure auditing for application context values:

```
AUDIT CONTEXT NAMESPACE context_name1 ATTRIBUTES attribute1 [, attribute2] [, CONTEXT NAMESPACE context_name2 ATTRIBUTES attribute1 [, attribute2]] [BY user list];
```

In this specification:

- context_name1: Optionally, you can include one additional CONTEXT name-attribute value pair.
- user_list is an optional list of database user accounts. Separate multiple names with a
 comma. If you omit this setting, then Oracle Database configures the application context
 policy for all users. When each user logs in, a list of all pertinent application contexts and
 their attributes is cached for the user session.

For example:

```
AUDIT CONTEXT NAMESPACE clientcontext3 ATTRIBUTES module, action, CONTEXT NAMESPACE ols_session_labels ATTRIBUTES ols_pol1, ols_pol3 BY appuser1, appuser2;
```

To find a list of currently configured application context audit settings, query the AUDIT_UNIFIED_CONTEXTS data dictionary view.

27.2.11.3 Disabling Application Context Audit Settings

The NOAUDIT statement disables application context audit settings.

 To disable an application context audit setting, specify the namespace and attribute settings in the NOAUDIT statement. You can enter the attributes in any order (that is, they do not need to match the order used in the corresponding AUDIT CONTEXT statement.)

For example:

```
NOAUDIT CONTEXT NAMESPACE client_context ATTRIBUTES module,
CONTEXT NAMESPACE ols_session_labels ATTRIBUTES ols_pol1, ols_pol3
BY USERS WITH GRANTED ROLES emp_admin;
```

To find the currently audited application contexts, query the AUDIT_UNIFIED_CONTEXTS data dictionary view.

27.2.11.4 Example: Auditing Application Context Values in a Default Database

The AUDIT CONTEXT NAMESPACE statement can audit application context values.

Example 27-18 shows how to audit the clientcontext application values for the module and action attributes, by the user appuser1.

Example 27-18 Auditing Application Context Values in a Default Database

AUDIT CONTEXT NAMESPACE clientcontext ATTRIBUTES module, action BY appuser1;

27.2.11.5 Example: Auditing Application Context Values from Oracle Label Security

The AUDIT CONTEXT NAMESPACE statement can audit application context values from Oracle Label Security.

Example 27-19 shows how to audit an application context for Oracle Label Security called ols session labels, for the attributes ols poll and ols poll.

Example 27-19 Auditing Application Context Values from Oracle Label Security

AUDIT CONTEXT NAMESPACE ols session labels ATTRIBUTES ols pol1, ols pol2;

27.2.11.6 How Audited Application Contexts Appear in the Audit Trail

The UNIFIED AUDIT POLICIES data dictionary view lists application context audit events.

The <code>APPLICATION_CONTEXTS</code> column of the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view shows application context audit data. The application contexts appear as a list of semi-colon separated values.

For example:

27.2.12 Auditing Oracle Database Real Application Security Events

You can use CREATE AUDIT POLICY statement to audit Oracle Database Real Application Security events.

- About Auditing Oracle Database Real Application Security Events
 You must have the AUDIT_ADMIN role to audit Oracle Database Real Application Security
 events.
- Oracle Database Real Application Security Auditable Events
 Oracle Database provides Real Application Security events that you can audit, such CREATE USER, UPDATE USER.
- Oracle Database Real Application Security User, Privilege, and Role Audit Events
 The unified audit trail can capture Oracle Database Real Application Security events for
 users, privileges, and roles.

- Oracle Database Real Application Security Security Class and ACL Audit Events
 The unified audit trail can capture Oracle Database Real Application Security security class and ACL audit events.
- Oracle Database Real Application Security Session Audit Events
 The unified audit trail can capture Oracle Database Real Application Security session audit events.
- Oracle Database Real Application Security ALL Events
 The unified audit trail can capture Oracle Database Real Application Security ALL events.
- Example: Auditing Real Application Security User Account Modifications
 The CREATE AUDIT POLICY statement can audit Real Application Security user account modifications.
- Example: Using a Condition in a Real Application Security Unified Audit Policy
 The CREATE AUDIT POLICY statement can set a condition for a Real Application Security
 unified audit policy.
- How Oracle Database Real Application Security Events Appear in the Audit Trail
 The DBA_XS_AUDIT_TRAIL data dictionary view lists Oracle Real Application Security audit

27.2.12.1 About Auditing Oracle Database Real Application Security Events

You must have the ${\tt AUDIT_ADMIN}$ role to audit Oracle Database Real Application Security events.

To access the audit trail, you can query the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view, whose Real Application Security-specific columns begin with <code>XS_</code>. If you want to find audit information about the internally generated VPD predicate that is created while an Oracle Real Application Security policy is being enabled, then you can query the <code>RLS_INFO</code> column.

Real Application Security-specific views are as follows:

- DBA_XS_AUDIT_TRAIL provides detailed information about Real Application Security events that were audited.
- DBA_XS_AUDIT_POLICY_OPTIONS describes the auditing options that were defined for Real Application Security unified audit policies.
- DBA_XS_ENB_AUDIT_POLICIES lists users for whom Real Application Security unified audit
 polices are enabled.



See Also:

- Auditing Application Context Values
 Oracle Database Real Application Security Predefined Audit Policies
- Auditing of Oracle Virtual Private Database Predicates for information about how to format the output of the RLS INFO column
- Oracle Database Real Application Security Administrator's and Developer's Guide for detailed information about Oracle Database Real Application Security

27.2.12.2 Oracle Database Real Application Security Auditable Events

Oracle Database provides Real Application Security events that you can audit, such CREATE USER, UPDATE USER.

To find a list of auditable Real Application Security events that you can audit, you can query the COMPONENT and NAME columns of the AUDITABLE_SYSTEM_ACTIONS data dictionary view, as follows:

```
SELECT NAME FROM AUDITABLE_SYSTEM_ACTIONS WHERE COMPONENT = 'XS';

NAME
------
CREATE USER
UPDATE USER
DELETE USER
```

Related Topics

- Oracle Database Real Application Security User, Privilege, and Role Audit Events
 The unified audit trail can capture Oracle Database Real Application Security events for
 users, privileges, and roles.
- Oracle Database Real Application Security Security Class and ACL Audit Events
 The unified audit trail can capture Oracle Database Real Application Security security class and ACL audit events.
- Oracle Database Real Application Security Session Audit Events
 The unified audit trail can capture Oracle Database Real Application Security session audit events.
- Oracle Database Real Application Security ALL Events
 The unified audit trail can capture Oracle Database Real Application Security ALL events.

27.2.12.3 Oracle Database Real Application Security User, Privilege, and Role Audit Events

The unified audit trail can capture Oracle Database Real Application Security events for users, privileges, and roles.

Table 27-4 describes these events.



Table 27-4 Oracle Database Real Application Security User, Privilege, and Role Audit Events

Audit Event	Description
CREATE USER	Creates an Oracle Database Real Application Security user account through the XS_PRINCIPAL.CREATE_USER procedure
UPDATE USER	Updates an Oracle Database Real Application Security user account through the following procedures: • XS_PRINCIPAL.SET_EFFECTIVE_DATES • XS_PRINCIPAL.SET_USER_DEFAULT_ROLES_ALL • XS_PRINCIPAL.SET_USER_SCHEMA • XS_PRINCIPAL.SET_GUID • XS_PRINCIPAL.SET_USER_STATUS • XS_PRINCIPAL.SET_USER_STATUS
DELETE USER	Deletes an Oracle Database Real Application Security user account through the through the XS_PRINCIPAL.DELETE_PRINCIPAL procedure
AUDIT_GRANT_PRIVILEGE	Audits the GRANT_SYSTEM_PRIVILEGE privilege
AUDIT_REVOKE_PRIVILEG E	Audits the REVOKE_SYSTEM_PRIVILEGE privilege
CREATE ROLE	Creates an Oracle Database Real Application Security role through the XS_PRINCIPAL.CREATE_ROLE procedure
UPDATE ROLE	Updates an Oracle Database Real Application Security role through the following procedures: • XS_PRINCIPAL.SET_DYNAMIC_ROLE_SCOPE • XS_PRINCIPAL.SET_DYNAMIC_ROLE_DURATION • XS_PRINCIPAL.SET_EFFECTIVE_DATES • XS_PRINCIPAL.SET_ROLE_DEFAULT
DELETE ROLE	Deletes an Oracle Database Real Application Security role through the XS PRINCIPAL. DELETE ROLE procedure
GRANT ROLE	Grants Oracle Database Real Application Security roles through the XS_PRINCIPAL.GRANT_ROLES procedure
REVOKE ROLE	Revokes Oracle Database Real Application Security roles through the XS_PRINCIPAL.REVOKE_ROLES procedure and revokes all granted roles through the XS_PRINCIPAL.REVOKE_ALL_GRANTED_ROLES procedure
ADD PROXY	Adds Oracle Database Real Application Security proxy user account through the XS_PRINCIPAL.ADD_PROXY_USER procedure, and adds proxies to database users through the XS_PRINCIPAL.ADD_PROXY_TO_SCHEMA procedure
REMOVE PROXY	Removes an Oracle Database Real Application Security proxy user account through the XS_PRINCIPAL.REMOVE_PROXY_USER, XS_PRINCIPAL.REMOVE_ALL_PROXY_USERS, and XS_PRINCIPAL.REMOVE_PROXY_FROM_SCHEMA PROCEDURES
SET USER PASSWORD	Sets the Oracle Database Real Application Security user account password through the <code>XS_PRINCIPAL.SET_PASSWORD</code> procedure
SET USER VERIFIER	Sets the Oracle Database Real Application Security proxy user account verifier through the <code>XS_PRINCIPAL.SET_VERIFIER</code> procedure



27.2.12.4 Oracle Database Real Application Security Security Class and ACL Audit Events

The unified audit trail can capture Oracle Database Real Application Security security class and ACL audit events.

Table 27-5 describes these events.

Table 27-5 Oracle Database Real Application Security Security Class and ACL Audit Events

Audit Event	Description
CREATE SECURITY CLASS	Creates a security class through the XS_SECURITY_CLASS.CREATE_SECURITY_CLASS procedure
UPDATE SECURITY CLASS	Creates a security class through the following procedures: XS_SECURITY_CLASS.SET_DEFAULT_ACL XS_SECURITY_CLASS.ADD_PARENTS XS_SECURITY_CLASS.REMOVE_ALL_PARENTS XS_SECURITY_CLASS.REMOVE_PARENTS XS_SECURITY_CLASS.ADD_PRIVILEGES XS_SECURITY_CLASS.REMOVE_ALL_PRIVILEGES XS_SECURITY_CLASS.ADD_IMPLIED_PRIVILEGES XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES XS_SECURITY_CLASS.REMOVE_IMPLIED_PRIVILEGES XS_SECURITY_CLASS.REMOVE_ALL_IMPLIED_PRIVILEGES XS_SECURITY_CLASS.REMOVE_ALL_IMPLIED_PRIVILEGES XS_SECURITY_CLASS.SET_DESCRIPTION
DELETE SECURITY CLASS	Deletes a security class through the XS_SECURITY_CLASS.DELETE_SECURITY_CLASS procedure
CREATE ACL	Creates an Access Control List (ACL) through the XS_ACL.CREATE_ACL procedure
UPDATE ACL	Updates an ACL through the following procedures: XS_ACL.APPEND_ACES XS_ACL.REMOVE_ALL_ACES XS_ACL.SET_SECURITY_CLASS XS_ACL.SET_PARENT_ACL XS_ACL.ADD_ACL_PARAMETER XS_ACL.REMOVE_ALL_ACL_PARAMETERS XS_ACL.REMOVE_ACL_PARAMETER XS_ACL.SET_DESCRIPTION
DELETE ACL	Deletes an ACL through the ${\tt XS_ACL.DELETE_ACL}$ procedure
CREATE DATA SECURITY-	Creates a data security policy through the XS_DATA_SECURITY.CREATE_DATA_SECURITY procedure
UPDATE DATA SECURITY	 Updates a data security policy through the following procedures: XS_DATA_SECURITY.CREATE_ACL_PARAMETER XS_DATA_SECURITY.DELETE_ACL_PARAMETER XS_DATA_SECURITY.SET_DESCRIPTION
DELETE DATA SECURITY	Deletes a data security policy through the XS_DATA_SECURITY.DELETE_DATA_SECURITY procedure



Table 27-5 (Cont.) Oracle Database Real Application Security Security Class and ACL Audit Events

Audit Event	Description
ENABLE DATA SECURITY	Enables extensible data security for a database table or view through the XS_DATA_SECURITY.ENABLE_OBJECT_POLICY procedure
DISABLE DATA SECURITY	Disables extensible data security for a database table or view through the XS_DATA_SECURITY.DISABLE_XDS procedure

27.2.12.5 Oracle Database Real Application Security Session Audit Events

The unified audit trail can capture Oracle Database Real Application Security session audit events.

Table 27-4 describes these events.

Table 27-6 Oracle Database Real Application Security Session Audit Events

udit Event	Description
REATE SESSION	Creates a session through the DBMS_XS_SESSIONS.CREATE_SESSION procedure
ESTROY SESSION	Destroys a session through the DBMS_XS_SESSIONS.DESTROY_SESSION procedure
REATE SESSION NAMESPACE	Creates a namespace through the DBMS_XS_SESSIONS.CREATE_NAMESPACE procedure
ELETE SESSION NAMESPACE	Deletes a namespace through the DBMS_XS_SESSIONS.DELETE_NAMESPACE procedure
REATE NAMESPACE ATTRIBUTE	Creates a namespace attribute through the DBMS_XS_SESSIONS.CREATE_ATTRIBUTE procedure
ET NAMESPACE ATTRIBUTE	Sets a namespace attribute through the DBMS_XS_SESSIONS.SET_ATTRIBUTE procedure
ET NAMESPACE ATTRIBUTE	Gets a namespace attribute through the DBMS_XS_SESSIONS.GET_ATTRIBUTE procedure
ELETE NAMESPACE ATTRIBUTE	Deletes a namespace attribute through the DBMS_XS_SESSIONS.DELETE_ATTRIBUTE procedure
REATE NAMESPACE TEMPLATE	Creates a namespace attribute through the XS_NS_TEMPLATE.CREATE_NS_TEMPLATE procedure
PDATE NAMESPACE TEMPLATE	Updates a namespace attribute through the following procedures: XS_NS_TEMPLATE.SET_HANDLER XS_NS_TEMPLATE.ADD_ATTRIBUTES XS_NS_TEMPLATE.REMOVE_ALL_ATTRIBUTES XS_NS_TEMPLATE.REMOVE_ATTRIBUTES XS_NS_TEMPLATE.SET_DESCRIPTION
ELETE NAMESPACE TEMPLATE	Deletes a namespace through the XS_NS_TEMPLATE.DELETE_NS_TEMPLATE procedure
DD GLOBAL CALLBACK	Adds a global callback through the DBMS XS SESSIONS.ADD GLOBAL CALLBACK procedure



Table 27-6 (Cont.) Oracle Database Real Application Security Session Audit Events

Audit Event	Description
DELETE GLOBAL CALLBACK	Deletes a global callback through the DBMS_XS_SESSIONS.DELETE_GLOBAL_CALLBACK procedure
ENABLE GLOBAL CALLBACK	Enables a global callback through the DBMS_XS_SESSIONS.ENABLE_GLOBAL_CALLBACK procedure
SET COOKIE	Sets a session cookie through the DBMS_XS_SESSIONS.SET_SESSION_COOKIE procedure
SET INACTIVE TIMEOUT	Sets the time-out time for inactive sessions through the DBMS_XS_SESSIONS.SET_INACTIVITY_TIMEOUT procedure
SWITCH USER	Sets the security context of the current lightweight user session to a newly initialized security context for a specified user through the DBMS_XS_SESSIONS.SWITCH_USER procedure
ASSIGN USER	Assigns or removes one or more dynamic roles for the specified user through the <code>DBMS_XS_SESSIONS.ASSIGN_USER</code> procedure
ENABLE ROLE	Enable a role for a lightweight user session through the DBMS_XS_SESSIONS.ENABLE_ROLE procedure
DISABLE ROLE	Disables a role for a lightweight user session through the DBMS_XS_SESSIONS.DISABLE_ROLE procedure

27.2.12.6 Oracle Database Real Application Security ALL Events

The unified audit trail can capture Oracle Database Real Application Security ALL events.

Table 27-7 describes these events.

Table 27-7 Oracle Database Real Application Security ALL Events

Audit Event	Description
	<u> </u>
ALL	Captures all Real Application Security actions

27.2.12.7 Configuring a Unified Audit Policy for Oracle Database Real Application Security

The CREATE AUDIT POLICY statement can create a unified audit policy for Oracle Real Application Security.

 Use the following syntax to create a unified audit policy for Oracle Database Real Application Security:

```
CREATE AUDIT POLICY policy_name

ACTIONS COMPONENT=XS component_action1 [, action2];
```

For example:

CREATE AUDIT POLICY audit_ras_pol ACTIONS COMPONENT=XS SWITCH USER, DISABLE ROLE;

You can build more complex policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.12.8 Example: Auditing Real Application Security User Account Modifications

The CREATE AUDIT POLICY statement can audit Real Application Security user account modifications.

Example 27-20 shows how to audit user bhurst's attempts to switch users and disable roles.

Example 27-20 Auditing Real Application Security User Account Modifications

```
CREATE AUDIT POLICY ras_users_pol
ACTIONS COMPONENT=XS SWITCH USER, DISABLE ROLE;
AUDIT POLICY ras users pol BY bhurst;
```

27.2.12.9 Example: Using a Condition in a Real Application Security Unified Audit Policy

The CREATE AUDIT POLICY statement can set a condition for a Real Application Security unified audit policy.

Example 27-21 shows how to create Real Application Security unified audit policy that applies the audit only to actions from the nemosity computer host.

Example 27-21 Using a Condition in a Real Application Security Unified Audit Policy

```
CREATE AUDIT POLICY ras_acl_pol
ACTIONS DELETE ON OE.CUSTOMERS
ACTIONS COMPONENT=XS CREATE ACL, UPDATE ACL, DELETE ACL
WHEN 'SYS_CONTEXT(''USERENV'', ''HOST'') = ''nemosity'''
EVALUATE PER INSTANCE;

AUDIT POLICY ras_acl_pol BY pfitch;
```

27.2.12.10 How Oracle Database Real Application Security Events Appear in the Audit Trail

The DBA_XS_AUDIT_TRAIL data dictionary view lists Oracle Real Application Security audit events.

The following example queries the Real Application Security-specific view,

27.2.13 Auditing Oracle Recovery Manager Events

You can use the CREATE AUDIT POLICY statement to audit Oracle Recovery Manager events.

- About Auditing Oracle Recovery Manager Events
 - The UNIFIED_AUDIT_TRAIL data dictionary view automatically stores Oracle Recovery Manager audit events in the RMAN column.
- Oracle Recovery Manager Unified Audit Trail Events
 The unified audit trail can capture Oracle Recovery Manager events.
- How Oracle Recovery Manager Audited Events Appear in the Audit Trail
 The UNIFIED_AUDIT_TRAIL data dictionary view lists Oracle Recovery Manager audit events.

27.2.13.1 About Auditing Oracle Recovery Manager Events

The UNIFIED_AUDIT_TRAIL data dictionary view automatically stores Oracle Recovery Manager audit events in the RMAN column.

Unlike other Oracle Database components, you do not create a unified audit policy for Oracle Recovery Manager events.

However, you must have the AUDIT_ADMIN or AUDIT_VIEWER role in order to query the UNIFIED_AUDIT_TRAIL view to see these events. If you have the SYSBACKUP or the SYSDBA administrative privilege, then you can find additional information about Recovery Manager jobs by querying views such as V\$RMAN STATUS or V\$RMAN BACKUP JOB DETAILS.



Oracle Database Backup and Recovery User's Guide

27.2.13.2 Oracle Recovery Manager Unified Audit Trail Events

The unified audit trail can capture Oracle Recovery Manager events.

Table 27-8 describes these events.

Table 27-8 Oracle Recovery Manager Columns in UNIFIED_AUDIT_TRAIL View

Recovery Manager Column	Description
RMAN_SESSION_RECID	Recovery Manager session identifier. Together with the RMAN_SESSION_STAMP column, this column uniquely identifies the Recovery Manager job. The Recovery Manager session ID is a a RECID value in the control file that identifies the Recovery Manager job. (Note that the Recovery Manager session ID is not the same as a user session ID.)
RMAN_SESSION_STAMP	Timestamp for the session. Together with the RMAN_SESSION_RECID column, this column identifies Recovery Manager jobs.
RMAN_OPERATION	The Recovery Manager operation executed by the job. One row is added for each distinct operation within a Recovery Manager session. For example, a backup job contains BACKUP as the RMAN_OPERATION value.



Table 27-8 (Cont.) Oracle Recovery Manager Columns in UNIFIED_AUDIT_TRAIL View

Recovery Manager Column	Description
RMAN_OBJECT_TYPE	Type of objects involved in a Recovery Manager session. It contains one of the following values. If the Recovery Manager session does not satisfy more than one of them, then preference is given in the following order, from top to bottom of the list.
	1. DB FULL (Database Full) refers to a full backup of the database
	2. RECVR AREA refers to the Fast Recovery area
	3. DB INCR (Database Incremental) refers to incremental backups of the database
	4. DATAFILE FULL refers to a full backup of the data files
	5. DATAFILE INCR refers to incremental backups of the data files
	6. ARCHIVELOG refers to archived redo log files
	7. CONTROLFILE refers to control files
	8. SPFILE refers to the server parameter file
	9. BACKUPSET refers to backup files
RMAN_DEVICE_TYPE	Device associated with a Recovery Manager session. This column can be DISK, SBT (system backup tape), or * (asterisk). An asterisk indicates more than one device. In most cases, the value will be DISK and SBT.

27.2.13.3 How Oracle Recovery Manager Audited Events Appear in the Audit Trail

The UNIFIED_AUDIT_TRAIL data dictionary view lists Oracle Recovery Manager audit events.

Table 27-8 lists the columns in the UNIFIED_AUDIT_TRAIL data dictionary view that you can query to find Oracle Recovery Manager-specific audit data.

For example:

```
SELECT RMAN_OPERATION FROM UNIFIED_AUDIT_TRAIL
WHERE RMAN_OBJECT_TYPE = 'DB FULL';

RMAN_OPERATION
-----
BACKUP
```

27.2.14 Auditing Oracle Database Vault Events

In an Oracle Database Vault environment, the CREATE AUDIT POLICY statement can audit Database Vault activities.

- About Auditing Oracle Database Vault Events
 As with all unified auditing, you must have the AUDIT_ADMIN role before you can audit Oracle Database Vault events.
- Who Is Audited in Oracle Database Vault?
 Audited Oracle Database Vault users include administrators and users whose activities affect Database Vault enforcement policies.

About Oracle Database Vault Unified Audit Trail Events

The audit trail in an Oracle Database Vault environment captures all configuration changes or attempts at changes to Database Vault policies.

Oracle Database Vault Realm Audit Events

The unified audit trail captures Oracle Database Vault realm events.

Oracle Database Vault Rule Set and Rule Audit Events

The unified audit trail can capture Oracle Database Vault rule set and rule audit events.

Oracle Database Vault Command Rule Audit Events

The unified audit trail can capture Oracle Database Vault command rule audit events.

Oracle Database Vault Factor Audit Events

The unified audit trail can capture Oracle Database Vault factor events.

Oracle Database Vault Secure Application Role Audit Events

The unified audit trail can capture Oracle Database Vault secure application role audit events.

Oracle Database Vault Oracle Label Security Audit Events

The unified audit trail can capture Oracle Database Vault Oracle Label Security audit events.

Oracle Database Vault Oracle Data Pump Audit Events

The unified audit trail can capture Oracle Database Vault Oracle Data Pump audit events.

Oracle Database Vault Enable and Disable Audit Events

The unified audit trail can capture Oracle Database Vault enable and disable audit events.

· Configuring a Unified Audit Policy for Oracle Database Vault

The ACTIONS and ACTIONS COMPONENT clauses in the CREATE AUDIT POLICY statement can create unified audit policies for Oracle Database Vault events.

Example: Auditing an Oracle Database Vault Realm

The CREATE AUDIT POLICY statement can audit Oracle Database Vault realms.

Example: Auditing an Oracle Database Vault Rule Set

The CREATE AUDIT POLICY statement can audit Oracle Database Vault rule sets.

Example: Auditing Two Oracle Database Vault Events

The CREATE AUDIT POLICY statement can audit multiple Oracle Database Vault events.

• Example: Auditing Oracle Database Vault Factors

The CREATE AUDIT POLICY statement can audit Oracle Database Vault factors.

How Oracle Database Vault Audited Events Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists Oracle Database Vault audited events.

27.2.14.1 About Auditing Oracle Database Vault Events

As with all unified auditing, you must have the AUDIT_ADMIN role before you can audit Oracle Database Vault events.

To create Oracle Database Vault unified audit policies, you must set the CREATE AUDIT POLICY statement's COMPONENT clause to DV, and then specify an action, such as Rule Set Failure, and an object, such as the name of a rule set.

To access the audit trail, you can query the following views:

- UNIFIED AUDIT TRAIL
- AUDSYS.DV\$CONFIGURATION AUDIT



AUDSYS.DV\$ENFORCEMENT AUDIT

In the UNIFIED_AUDIT_TRAIL view, the Oracle Database Vault-specific columns begin with DV_. You must have the AUDIT_VIEWER role before you can query the UNIFIED_AUDIT_TRAIL view.

In addition to these views, the Database Vault reports capture the results of Database Vaultspecific unified audit policies.

See Also:

- Oracle Database Vault Predefined Unified Audit Policy for DVSYS and LBACSYS Schemas
- Oracle Database Vault Administrator's Guide for detailed information about
 Oracle Database Vault audit policies

27.2.14.2 Who Is Audited in Oracle Database Vault?

Audited Oracle Database Vault users include administrators and users whose activities affect Database Vault enforcement policies.

These users are as follows:

- Database Vault administrators. All configuration changes that are made to Oracle
 Database Vault are mandatorily audited. The auditing captures activities such as creating,
 modifying, or deleting realms, factors, command rules, rule sets, rules, and so on. The
 AUDSYS.DV\$CONFIGURATION_AUDIT data dictionary view captures configuration changes
 made by Database Vault administrators.
- Users whose activities affect Oracle Database Vault enforcement policies. The AUDSYS.DV\$ENFORCEMENT_AUDIT data dictionary view captures enforcement-related audits

See Also:

Oracle Database Vault Administrator's Guide for more information about the AUDSYS.DV\$CONFIGURATION_AUDIT and AUDSYS.DV\$ENFORCEMENT_AUDIT data dictionary views

27.2.14.3 About Oracle Database Vault Unified Audit Trail Events

The audit trail in an Oracle Database Vault environment captures all configuration changes or attempts at changes to Database Vault policies.

It also captures violations by users to existing Database Vault policies.

You can audit the following kinds of Oracle Database Vault events:

All configuration changes or attempts at changes to Oracle Database Vault policies.
 It captures both Database Vault administrator changes and attempts made by unauthorized users.



Violations by users to existing Database Vault policies. For example, if you create a
policy to prevent users from accessing a specific schema table during non-work hours, the
audit trail will capture this activity.

27.2.14.4 Oracle Database Vault Realm Audit Events

The unified audit trail captures Oracle Database Vault realm events.

Table 27-9 describes these events.

Table 27-9 Oracle Database Vault Realm Audit Events

Audit Event	Description
CREATE_REALM	Creates a realm through the DVSYS.DBMS_MACADM.CREATE_REALM procedure
UPDATE_REALM	Updates a realm through the DVSYS.DBMS_MACADM.UPDATE_REALM procedure
RENAME_REALM	Renames a realm through the DVSYS.DBMS_MACADM.RENAME_REALM procedure
DELETE_REALM	Deletes a realm through the DVSYS.DBMS_MACADM.DELETE_REALM procedure
DELETE_REALM_CASCADE	Deletes a realm and its related Database Vault configuration information through the DVSYS.DBMS_MACADM.DELETE_REALM_CASCADE procedure
ADD_AUTH_TO_REALM	Adds an authorization to the realm through the DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM procedure
DELETE_AUTH_FROM_REALM	Removes an authorization from the realm through the DVSYS.DBMS_MACADM.DELETE_AUTH_FROM_REALM procedure
UPDATE_REALM_AUTH	Updates a realm authorization through the DVSYS.DBMS_MACADM.UPDATE_REALM_AUTHORIZATION procedure
ADD_OBJECT_TO_REALM	Adds an object to a realm authorization through the DVSYS.DBMS_MACADM.ADD_AUTH_TO_REALM procedure
DELETE_OBJECT_FROM_REALM	Removes an object from a realm authorization through the DVSYS.DBMS_MACADM.DELETE_OBJECT_FROM_REALM procedure

27.2.14.5 Oracle Database Vault Rule Set and Rule Audit Events

The unified audit trail can capture Oracle Database Vault rule set and rule audit events.

Table 27-10 describes these events.

Table 27-10 Oracle Database Vault Rule Set and Rule Audit Events

Audit Event	Description
CREATE_RULE_SET	Creates a rule set through the DVSYS.DBMS_MACADM.CREATE_RULE_SET procedure



Table 27-10 (Cont.) Oracle Database Vault Rule Set and Rule Audit Events

Audit Event	Description
UPDATE_RULE_SET	Updates a rule set through the DVSYS.DBMS_MACADM.UPDATE_RULE_SET procedure
RENAME_RULE_SET	Renames a rule set through the DVSYS.DBMS_MACADM.RENAME_RULE_SET procedure
DELETE_RULE_SET	Deletes a rule set through the DVSYS.DBMS_MACADM.DELETE_RULE_SET procedure
ADD_RULE_TO_RULE_SET	Adds a rule to an existing rule set through the DVSYS.DBMS_MACADM.ADD_RULE_TO_RULE_SET procedure
DELETE_RULE_FROM_RULE_SET	Removes a rule from an existing rule set through the DVSYS.DBMS_MACADM.DELETE_RULE_FROM_RULE_SET procedure
CREATE_RULE	Creates a rule through the DVSYS.DBMS_MACADM.CREATE_RULE procedure
UPDATE_RULE	Updates a rule through the DVSYS.DBMS_MACADM.UPDATE_RULE procedure
RENAME_RULE	Renames a rule through the DVSYS.DBMS_MACADM.RENAME_RULE procedure
DELETE_RULE	Deletes a rule through the DVSYS.DBMS_MACADM.DELETE_RULE procedure
SYNC_RULES	Synchronizes the rules in Oracle Database Vault and Advanced Queuing Rules engine through the DVSYS.DBMS_MACADM.SYNC_RULES procedure

27.2.14.6 Oracle Database Vault Command Rule Audit Events

The unified audit trail can capture Oracle Database Vault command rule audit events.

Table 27-11 describes these events.

Table 27-11 Oracle Database Vault Command Rule Audit Events

Audit Event	Description
CREATE_COMMAND_RULE	Creates a command rule through the DVSYS.DBMS_MACADM.CREATE_COMMAND_RULE procedure
DELETE_COMMAND_RULE	Deletes a command rule through the DVSYS.DBMS_MACADM.DELETE_COMMAND_RULE procedure
UPDATE_COMMAND_RULE	Updates a command rule through the DVSYS.DBMS_MACADM.UPDATE_COMMAND_RULE procedure



27.2.14.7 Oracle Database Vault Factor Audit Events

The unified audit trail can capture Oracle Database Vault factor events.

Table 27-12 describes these events.

Table 27-12 Oracle Database Vault Factor Audit Events

Audit Event	Description
CREATE_FACTOR_TYPE	Creates a factor type through the DVSYS.DBMS_MACADM.CREATE_FACTOR_TYPE procedure
DELETE_FACTOR_TYPE	Deletes a factor type through the DVSYS.DBMS_MACADM.DELETE_FACTOR_TYPE procedure
UPDATE_FACTOR_TYPE	Updates a factor type through the DVSYS.DBMS_MACADM.UPDATE_FACTOR_TYPE procedure
RENAME_FACTOR_TYPE	Renames a factor type through the DVSYS.DBMS_MACADM.RENAME_FACTOR_TYPE procedure
CREATE_FACTOR	Creates a factor through the DVSYS.DBMS_MACADM.CREATE_FACTOR procedure
UPDATE_FACTOR	Updates a factor through the DVSYS.DBMS_MACADM.UPDATE_FACTOR procedure
DELETE_FACTOR	Deletes a factor through the DVSYS.DBMS_MACADM.DELETE_FACTOR procedure
RENAME_FACTOR	Renames a factor through the DVSYS.DBMS_MACADM.RENAME_FACTOR procedure
ADD_FACTOR_LINK	Specifies a parent-child relationship between two factors through the <code>DVSYS.DBMS_MACADM.ADD_FACTOR_LINK</code> procedure
DELETE_FACTOR_LINK	Removes the parent-child relationship between two factors through the DVSYS.DBMS_MACADM.DELETE_FACTOR_LINK procedure
ADD_POLICY_FACTOR	Specifies that the label for a factor contributes to the Oracle Label Security label for a policy, through the DVSYS.DBMS_MACADM.ADD_POLICY_FACTOR procedure
DELETE_POLICY_FACTOR	Removes factor label from being associated with an Oracle Label Security label for a policy, through the DBMS_MACADM.DELETE_POLICY_FACTOR procedure
CREATE_IDENTITY	Creates a factor identity through the DVSYS.DBMS_MACADM.CREATE_IDENTITY procedure
UPDATE_IDENTITY	Updates a factor identity through the DVSYS.DBMS_MACADM.UPDATE_IDENTITY procedure
CHANGE_IDENTITY_FACTOR	Associates an identity with a different factor through the DVSYS.DBMS_MACADM.CHANGE_IDENTITY_FACTOR procedure
CHANGE_IDENTITY_VALUE	Updates the value of an identity through the DVSYS.DBMS_MACADM.CHANGE_IDENTITY_VALUE procedure
DELETE_IDENTITY	Deletes an existing factor identity through the DVSYS.DBMS_MACADM.DELETE_IDENTITY procedure



Table 27-12 (Cont.) Oracle Database Vault Factor Audit Events

Audit Event	Description
CREATE_IDENTITY_MAP	Creates a factor identity map through the DVSYS.DBMS_MACADM.CREATE_IDENTITY_MAP procedure
DELETE_IDENTITY_MAP	Deletes a factor identity map through the DVSYS.DBMS_MACADM.DELETE_IDENTITY_MAP procedure
CREATE_DOMAIN_IDENTITY	Adds an Oracle Database Real Application Clusters database node to the domain factor identities and labels it according to the Oracle Label Security policy, through the DVSYS.DBMS_MACADM.CREATE_DOMAIN_IDENTITY procedure
DROP_DOMAIN_IDENTITY	Drops an Oracle RAC node from the domain factor identities through the DVSYS.DBMS_MACADM.DROP_DOMAIN_IDENTITY procedure

27.2.14.8 Oracle Database Vault Secure Application Role Audit Events

The unified audit trail can capture Oracle Database Vault secure application role audit events.

Table 27-13 describes these events.

Table 27-13 Oracle Database Vault Secure Application Role Audit Events

Audit Event	Description
CREATE_ROLE	Creates an Oracle Database Vault secure application role through the DVSYS.DBMS_MACADM.CREATE_ROLE procedure
DELETE_ROLE	Deletes an Oracle Database Vault secure application role through the <code>DVSYS.DBMS_MACADM.DELETE_ROLE</code> procedure
UPDATE_ROLE	Updates an Oracle Database Vault secure application role through the DVSYS.DBMS_MACADM.UPDATE_ROLE procedure
RENAME_ROLE	Renames an Oracle Database Vault secure application role through the DVSYS.DBMS_MACADM.RENAME_ROLE procedure

27.2.14.9 Oracle Database Vault Oracle Label Security Audit Events

The unified audit trail can capture Oracle Database Vault Oracle Label Security audit events.

Table 27-14 describes these events.

Table 27-14 Oracle Database Vault Oracle Label Security Audit Events

Audit Event	Description
CREATE_POLICY_LABEL	Creates an Oracle Label Security policy label through the DVSYS.DBMS_MACADM.CREATE_POLICY_LABEL procedure
DELETE_POLICY_LABEL	Deletes an Oracle Label Security policy label through the DVSYS.DBMS_MACADM.DELETE_POLICY_LABEL procedure
CREATE_MAC_POLICY	Specifies the algorithm that is used to merge labels when computing the label for a factor, or the Oracle Label Security Session label, through the DVSYS.DBMS_MACADM.CREATE_MAC_POLICY procedure
UPDATE_MAC_POLICY	Changes the Oracle Label Security merge label algorithm through the <code>DVSYS.DBMS_MACADM.UPDATE_MAC_POLICY</code> procedure
DELETE_MAC_POLICY_CASCADE	Deletes all Oracle Database Vault objects related to an Oracle Label Security policy, through the DVSYS.DBMS_MACADM.DELETE_MAC_POLICY_CASCADE procedure

27.2.14.10 Oracle Database Vault Oracle Data Pump Audit Events

The unified audit trail can capture Oracle Database Vault Oracle Data Pump audit events.

Table 27-15 describes these events.

Table 27-15 Oracle Database Vault Oracle Data Pump Audit Events

Audit Event	Description
AUTHORIZE_DATAPUMP_USER	Authorizes an Oracle Data Pump user through the DVSYS.DBMS_MACADM.AUTHORIZE_DATAPUMP_USER procedure
UNAUTHORIZE_DATAPUMP_USER	Removes from authorization an Oracle Data Pump user through the DVSYS.DBMS_MACADM.UNAUTHORIZE_DATAPUMP_USER procedure

27.2.14.11 Oracle Database Vault Enable and Disable Audit Events

The unified audit trail can capture Oracle Database Vault enable and disable audit events.

Table 27-16 describes these events.

Table 27-16 Oracle Database Vault Enable and Disable Audit Events

Event	Description
ENABLE_EVENT	DBMS_MACADM.ENABLE_EVENT
DISABLE_EVENT	DBMS_MACADM.DISABLE_EVENT



27.2.14.12 Configuring a Unified Audit Policy for Oracle Database Vault

The ACTIONS and ACTIONS COMPONENT clauses in the CREATE AUDIT POLICY statement can create unified audit policies for Oracle Database Vault events.

Use the following syntax to create an Oracle Database Vault unified audit policy:

```
CREATE AUDIT POLICY policy_name

ACTIONS COMPONENT= DV DV action ON DV object [,DV action2 ON DV object2]
```

In this specification:

- DV action is one of the following:
 - Realm-related actions:

Realm Violation audits realm violations (for example, when an unauthorized user attempts to access a realm-protected object).

Realm Success audits when a realm-protected object is successfully accessed by an authorized user.

Realm Access audits both realm violation and realm success cases, that is, audits whenever the realm access attempt has been made, whether the access succeeded or failed.

- Rule set-related actions: Rule Set Failure, Rule Set Success, Rule Set Eval
- Factor-related actions: Factor Error, Factor Null, Factor Validate Error, Factor
 Validate False, Factor Trust Level Null, Factor Trust Level Neg, Factor All
- DV objects is one of the following:
 - Realm Name
 - Rule Set Name
 - Factor Name

If the object was created in lower or mixed case, then you must enclose <code>DV_objects</code> in double quotation marks. If you had created the object in all capital letters, then you can omit the quotation marks.

For example, to audit realm violations on the Database Vault Account Management realm:

```
CREATE AUDIT POLICY audit_dv

ACTIONS COMPONENT=DV Realm Violation ON "Database Vault Account Management";
```

Remember that after you create the policy, you must use the AUDIT statement to enable it.

27.2.14.13 Example: Auditing an Oracle Database Vault Realm

The CREATE AUDIT POLICY statement can audit Oracle Database Vault realms.

Example 27-22 shows how to audit a realm violation on the HR schema.

Example 27-22 Auditing a Realm Violation

```
CREATE AUDIT POLICY dv_realm_hr

ACTIONS COMPONENT=DV Realm Violation ON "HR Schema Realm";

AUDIT POLICY dv_realm_hr;
```



27.2.14.14 Example: Auditing an Oracle Database Vault Rule Set

The CREATE AUDIT POLICY statement can audit Oracle Database Vault rule sets.

Example: Auditing an Oracle Database Vault Rule Set shows how to audit the Can Maintain Accounts/Profile rule set.

Example 27-23 Auditing a Rule Set

```
CREATE AUDIT POLICY dv_rule_set_accts

ACTIONS COMPONENT=DV RULE SET FAILURE ON "Can Maintain Accounts/Profile";

AUDIT POLICY dv rule set accts;
```

27.2.14.15 Example: Auditing Two Oracle Database Vault Events

The CREATE AUDIT POLICY statement can audit multiple Oracle Database Vault events.

Example 27-24 shows how to audit a realm violation and a rule set failure.

Example 27-24 Auditing Two Oracle Database Vault Events

```
CREATE AUDIT POLICY audit_dv

ACTIONS COMPONENT=DV REALM VIOLATION ON "Oracle Enterprise Manager", Rule Set Failure ON "Allow Sessions";

AUDIT POLICY audit_dv;
```

27.2.14.16 Example: Auditing Oracle Database Vault Factors

The CREATE AUDIT POLICY statement can audit Oracle Database Vault factors.

Example 27-25 shows how to audit two types of errors for one factor.

Example 27-25 Auditing Oracle Database Vault Factor Settings

```
CREATE AUDIT POLICY audit_dv_factor

ACTIONS COMPONENT=DV FACTOR ERROR ON "Database_Domain", Factor Validate Error ON
"Client_IP";

AUDIT POLICY audit dv factor;
```

27.2.14.17 How Oracle Database Vault Audited Events Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists Oracle Database Vault audited events.

The DV_* columns of the UNIFIED_AUDIT_TRAIL view show Oracle Database Vault-specific audit data.

For example:

```
SELECT DBUSERNAME, SQL_TEXT, UNIFIED_AUDIT_POLICIES, DV_ACTION_NAME,

DV_ACTION_OBJECT_NAME, DV_RULE_SET_NAME

FROM UNIFIED_AUDIT_TRAIL

WHERE AUDIT_TYPE = 'DATABASE VAULT'

ORDER BY EVENT_TIMESTAMP;

DBUSERNAME SQL_TEXT UNIFIED_AUDIT_POLICIES DV_ACTION_NAME

DV_ACTION_OBJECT_NAME DV_RULE_SET_NAME
```



PFITCH SELECT * FROM HR.EMPLOYEES DV_AUDIT_POLICY Command Failure Audit SELECT HR_data_protection

27.2.15 Auditing Oracle Label Security Events

In an Oracle Label Security environment, the CREATE AUDIT POLICY statement can audit Oracle Label Security activities.

- About Auditing Oracle Label Security Events
 - As with all unified auditing, you must have the AUDIT_ADMIN role before you can audit Oracle Label Security (OLS) events.
- Oracle Label Security Unified Audit Trail Events
 The unified audit trail can capture Oracle Label Security audit events.
- Oracle Label Security Auditable User Session Labels
 The ORA_OLS_SESSION_LABELS application context can capture user session label usage for each Oracle Database event.
- Configuring a Unified Audit Policy for Oracle Label Security
 The ACTIONS and ACTIONS COMPONENT clauses in the CREATE AUDIT POLICY statement can be used to create Oracle Label Security event audit policies.
- Example: Auditing Oracle Label Security Session Label Attributes
 The AUDIT CONTEXT NAMESPACE statement can audit Oracle Label Security session label attributes.
- Example: Excluding a User from an Oracle Label Security Policy
 The CREATE AUDIT POLICY statement can exclude users from policies.
- Example: Auditing Oracle Label Security Policy Actions
 The CREATE AUDIT POLICY statement can audit Oracle Label Security policy actions.
- Example: Querying for Audited OLS Session Labels
 The LBACSYS.ORA_GET_AUDITED_LABEL function can be used in a UNIFIED_AUDIT_TRAIL
 query to find audited Oracle Label Security session labels.
- How Oracle Label Security Audit Events Appear in the Audit Trail
 The UNIFIED AUDIT TRAIL data dictionary view lists Oracle Label Security audit events.

27.2.15.1 About Auditing Oracle Label Security Events

As with all unified auditing, you must have the AUDIT_ADMIN role before you can audit Oracle Label Security (OLS) events.

To create Oracle Label Security unified audit policies, you must set the CREATE AUDIT POLICY statement COMPONENT clause to OLS.

To audit user session label information, you use the AUDIT statement to audit application context values.

To access the audit trail, you can query the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view. This view contains Oracle Label Security-specific columns whose names begin with <code>OLS_</code>. If you want to find audit information about the internally generated VPD predicate that is created when you apply an Oracle Label Security policy to a table, then you can query the <code>RLS_INFO</code> column.



See Also:

- Auditing of Oracle Virtual Private Database Predicates for information about how to format the output of the RLS INFO column
- Oracle Label Security Administrator's Guide for more information about Oracle Label Security

27.2.15.2 Oracle Label Security Unified Audit Trail Events

The unified audit trail can capture Oracle Label Security audit events.

To find a list of auditable Oracle Label Security events that you can audit, you can query the COMPONENT and NAME columns of the AUDITABLE_SYSTEM_ACTIONS data dictionary view.

For example:

Table 27-17 describes the Oracle Label Security audit events.

Table 27-17 Oracle Label Security Audit Events

Audit Event	Description
CREATE POLICY	Creates an Oracle Label Security policy through the SA_SYSDBA.CREATE_POLICY procedure
ALTER POLICY	Alters an Oracle Label Security policy through the SA_SYSDBA.ALTER_POLICY procedure
DROP POLICY	Drops an Oracle Label Security policy through the SA_SYSDBA.DROP_POLICY procedure
APPLY POLICY	Applies a table policy through the SA_POLICY_ADMIN.APPLY_TABLE_POLICY procedure or a schema policy through the SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY procedure
REMOVE POLICY	Removes a table policy through the SA_POLICY_ADMIN.REMOVE_TABLE_POLICY procedure or a schema policy through the SA_POLICY_ADMIN.REMOVE_SCHEMA_POLICY procedure
SET AUTHORIZATION	Covers all Oracle Label Security authorizations, including Oracle Label Security privileges and user labels to either users or trusted stored procedures. The PL/SQL procedures that correspond to the SET AUTHORIZATION event are SA_USER_ADMIN.SET_USER_LABELS, SA_USER_ADMIN.SET_USER_PRIVS, and SA_USER_ADMIN.SET_PROG_PRIVS.



Table 27-17 (Cont.) Oracle Label Security Audit Events

Audit Event	Description
PRIVILEGED ACTION	Covers any action that requires the user of an Oracle Label Security privilege. These actions are logons, SA_SESSION.SET_ACCESS_PROFILE executions, and the invocation of trusted stored procedures.
ENABLE POLICY	Enables an Oracle Label Security policy through the following procedures:
	 SA_SYSDBA.ENABLE_POLICY: Enforces access control on the tables and schemas protected by the policy SA_POLICY_ADMIN.ENABLE_TABLE_POLICY: Enables an Oracle Label Security policy for a specified table SA_POLICY_ADMIN.ENABLE_SCHEMA_POLICY: Enables an Oracle Label Security policy for all the tables in a specified schema
DISABLE POLICY	Disables an Oracle Label Security policy through the following procedures:
	 SA_SYSDBA.DISABLE_POLICY: Disables the enforcement of an Oracle Label Security policy
	 SA_POLICY_ADMIN.DISABLE_TABLE_POLICY: Disables the enforcement an Oracle Label Security policy for a specified table
	 SA_POLICY_ADMIN.DISABLE_SCHEMA_POLICY: Disables the enforcement of an Oracle Label Security policy for all the tables in a specified schema
SUBSCRIBE OID	Subscribes to an Oracle Internet Directory-enabled Oracle Label Security policy through the SA_POLICY_ADMIN.POLICY_SUBSCRIBE procedure
UNSUBSCRIBE OID	Unsubscribes to an Oracle Internet Directory-enabled Oracle Label Security policy through the SA POLICY ADMIN. POLICY UNSUBSCRIBE procedure
CREATE DATA LABEL	Creates an Oracle Label Security data label through the SA_LABEL_ADMIN.CREATE_LABEL procedure. CREATE DATA LABEL also corresponds to the LBACSYS.TO_DATA_LABEL function.
ALTER DATA LABEL	Alters an Oracle Label Security data label through the SA_LABEL_ADMIN.ALTER_LABEL procedure
DROP DATA LABEL	Drops an Oracle Label Security data label through the SA_LABEL_ADMIN.DROP_LABEL procedure
CREATE LABEL COMPONENT	Creates an Oracle Label Security component through the following procedures:
	• Levels: SA_COMPONENTS.CREATE_LEVEL
	• Compartments: SA_COMPONENTS.CREATE_COMPARTMENT • Groups: SA_COMPONENTS_CREATE_GROUP
ALTER LABEL COMPONENTS	 Groups: SA_COMPONENTS.CREATE_GROUP Alters an Oracle Label Security component through the following procedures:
	• Levels: SA_COMPONENTS.ALTER_LEVEL
	Compartments: SA_COMPONENTS.ALTER_COMPARTMENT
	• Groups: SA_COMPONENTS.ALTER_GROUP and SA_COMPONENTS.ALTER_GROUP_PARENT



Table 27-17 (Cont.) Oracle Label Security Audit Events

Audit Event	Description
DROP LABEL COMPONENTS	Drops an Oracle Label Security component through the following procedures:
	• Levels: SA_COMPONENTS.DROP_LEVEL
	• Compartments: SA_COMPONENTS.DROP_COMPARTMENT
	• Groups: SA_COMPONENTS.DROP_GROUP
ALL	Enables auditing of all Oracle Label Security actions

27.2.15.3 Oracle Label Security Auditable User Session Labels

The ORA_OLS_SESSION_LABELS application context can capture user session label usage for each Oracle Database event.

The attributes used by this application context refer to Oracle Label Security policies. .

The syntax is the same as the syntax used for application context auditing, described in Configuring Application Context Audit Settings. For example:

```
AUDIT CONTEXT NAMESPACE ORA SESSION LABELS ATTRIBUTES policy1, policy2;
```

Because the recording of session labels is not user-session specific, the BY $user_list$ clause is not required for auditing Oracle Label Security application contexts.

To disable the auditing of user session label information, you use the NOAUDIT statement. For example, to stop auditing for policies policy1 and policy2, enter the following statement:

NOAUDIT CONTEXT NAMESPACE ORA SESSION LABELS ATTRIBUTES policy1, policy2;

27.2.15.4 Configuring a Unified Audit Policy for Oracle Label Security

The ACTIONS and ACTIONS COMPONENT clauses in the CREATE AUDIT POLICY statement can be used to create Oracle Label Security event audit policies.

Use the following syntax to create an Oracle Label Security unified audit policy:

```
CREATE AUDIT POLICY policy_name
ACTIONS action1 [,action2 ]
ACTIONS COMPONENT=OLS component action1 [, action2];
```

For example:

CREATE AUDIT POLICY audit_ols ACTIONS SELECT ON OE.ORDERS ACTIONS COMPONENT=OLS ALL;

You can build more complex policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.15.5 Example: Auditing Oracle Label Security Session Label Attributes

The AUDIT CONTEXT NAMESPACE statement can audit Oracle Label Security session label attributes.

Example 27-26 shows how to audit ORA_OLS_SESSION_LABELS application context attributes for the Oracle Label Security policies usr pol1 and usr pol2.

Example 27-26 Auditing Oracle Label Security Session Label Attributes

AUDIT CONTEXT NAMESPACE ORA SESSION LABELS ATTRIBUTES usr pol1, usr pol2;

27.2.15.6 Example: Excluding a User from an Oracle Label Security Policy

The CREATE AUDIT POLICY statement can exclude users from policies.

Example 27-27 shows how to create a unified audit policy that excludes actions from user ols mgr.

Example 27-27 Excluding a User from an Oracle Label Security Policy

```
CREATE AUDIT POLICY auth_ols_audit_pol
ACTIONS SELECT ON HR.EMPLOYEES
ACTIONS COMPONENT=OLS DROP POLICY, DISABLE POLICY;
AUDIT POLICY auth ols audit pol EXCEPT ols mgr;
```

27.2.15.7 Example: Auditing Oracle Label Security Policy Actions

The CREATE AUDIT POLICY statement can audit Oracle Label Security policy actions.

Example 27-28 shows how to audit the DROP POLICY, DISABLE POLICY, UNSUBSCRIBE OID events, and UPDATE and DELETE statements on the HR.EMPLOYEES table. Then this policy is applied to the HR and LBACSYS users, and audit records are written to the unified audit trail only when the audited actions are successful.

Example 27-28 Auditing Oracle Label Security Policy Actions

```
CREATE AUDIT POLICY generic_audit_pol

ACTIONS UPDATE ON HR.EMPLOYEES, DELETE ON HR.EMPLOYEES

ACTIONS COMPONENT=OLS DROP POLICY, DISABLE POLICY, UNSUBSCRIBE OID;

AUDIT POLICY generic_audit_pol BY HR, LBACSYS WHENEVER SUCCESSFUL;
```

27.2.15.8 Example: Querying for Audited OLS Session Labels

The LBACSYS.ORA_GET_AUDITED_LABEL function can be used in a UNIFIED_AUDIT_TRAIL query to find audited Oracle Label Security session labels.

Example 27-29 shows how to use the LBACSYS.ORA_GET_AUDITED_LABEL function in a UNIFIED AUDIT TRAIL data dictionary view query.

Example 27-29 Querying for Audited Oracle Label Security Session Labels

```
SELECT ENTRY_ID, SESSIONID,

LBACSYS.ORA_GET_AUDITED_LABEL( APPLICATION_CONTEXTS, 'GENERIC_AUDIT_POL1') AS

SESSION_LABEL1,

LBACSYS.ORA_GET_AUDITED_LABEL( APPLICATION_CONTEXTS, 'GENERIC_AUDIT_POL2') AS

SESSION_LABEL2
```

27.2.15.9 How Oracle Label Security Audit Events Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists Oracle Label Security audit events.

The OLS_* columns of the UNIFIED_AUDIT_TRAIL view show Oracle Label Security-specific audit data. For example:

The session labels that the audit trail captures are stored in the <code>APPLICATION_CONTEXTS</code> column of the <code>UNIFIED_AUDIT_TRAIL</code> view. You can use the <code>LBACSYS.ORA_GET_AUDITED_LABEL</code> function to retrieve session labels that are stored in the <code>APPLICATION_CONTEXTS</code> column. This function accepts the <code>UNIFIED_AUDIT_TRAIL.APPLICATION_CONTEXTS</code> column value, and the Oracle Label Security policy name as arguments, and then returns the session label that is stored in the column for the specified policy.

See Also:

Oracle Label Security Administrator's Guide for more information about the $ORA_GET_AUDITED_LABEL$ function

27.2.16 Auditing Oracle Data Mining Events

You can use the CREATE AUDIT POLICY statement to audit Oracle Data Mining events.

- About Auditing Oracle Data Mining Events
 You must have the AUDIT ADMIN role to audit Oracle Data Mining events.
- Oracle Data Mining Unified Audit Trail Events
 The unified audit trail can capture Oracle Data Mining audit events...
- Configuring a Unified Audit Policy for Oracle Data Mining
 The CREATE AUDIT POLICY statement ACTIONS and ON MINING MODEL clauses can be used to create Oracle Data Mining event unified audit policies.
- Example: Auditing Multiple Oracle Data Mining Operations by a User
 The CREATE AUDIT POLICY statement can audit multiple Oracle Data Mining operations.
- Example: Auditing All Failed Oracle Data Mining Operations by a User
 The CREATE AUDIT POLICY statement can audit failed Oracle Data Mining operations by a user.

How Oracle Data Mining Events Appear in the Audit Trail
 The UNIFIED AUDIT TRAIL data dictionary view lists Oracle Data Mining audit events.

27.2.16.1 About Auditing Oracle Data Mining Events

You must have the AUDIT ADMIN role to audit Oracle Data Mining events.

To access the audit trail, you can query the UNIFIED AUDIT TRAIL data dictionary view.



Oracle Data Mining Concepts for more information about Oracle Data Mining

27.2.16.2 Oracle Data Mining Unified Audit Trail Events

The unified audit trail can capture Oracle Data Mining audit events...

Table 27-18 describes these events.

Table 27-18 Oracle Data Mining Audit Events

Audit Event	Description
AUDIT	Generates an audit record for a Data Mining model
COMMENT	Adds a comment to a Data Mining model
GRANT	Gives permission to a user to access the Data Mining model
RENAME	Changes the name of the Data Mining model
SELECT	Applies the Data Mining model or view its signature

27.2.16.3 Configuring a Unified Audit Policy for Oracle Data Mining

The CREATE AUDIT POLICY statement ACTIONS and ON MINING MODEL clauses can be used to create Oracle Data Mining event unified audit policies.

Use the following syntax to create a unified audit policy for Oracle Data Mining:

```
CREATE AUDIT POLICY policy_name
ACTIONS {operation | ALL}
ON MINING MODEL schema name.model name;
```

For example:

CREATE AUDIT POLICY dm ops ACTIONS RENAME ON MINING MODEL hr.dm emp;

You can build more complex policies, such as those that include conditions. Remember that after you create the policy, you must use the \mathtt{AUDIT} statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.16.4 Example: Auditing Multiple Oracle Data Mining Operations by a User

The CREATE AUDIT POLICY statement can audit multiple Oracle Data Mining operations.

Example 27-30 shows how to audit multiple Oracle Data Mining operations by user psmith. Include the ON MINING MODEL schema_name.model_name clause for each event, and separate each with a comma. This example specifies the same schema_name.model name for both actions, but the syntax enables you to specify different schema_name.model_name settings for different schemas and data models.

Example 27-30 Auditing Multiple Oracle Data Mining Operations by a User

```
CREATE AUDIT POLICY dm_ops_pol
ACTIONS SELECT ON MINING MODEL dmuser1.nb_model, ALTER ON MINING MODEL dmuser1.nb_model;
AUDIT POLICY dm_ops_pol BY psmith;
```

27.2.16.5 Example: Auditing All Failed Oracle Data Mining Operations by a User

The CREATE AUDIT POLICY statement can audit failed Oracle Data Mining operations by a user.

Example 27-31 shows how to audit all failed Oracle Data Mining operations by user psmith.

Example 27-31 Auditing All Failed Oracle Data Mining Operations by a User

```
CREATE AUDIT POLICY dm_all_ops_pol ACTIONS ALL ON MINING MODEL dmuser1.nb_model;
AUDIT POLICY dm all ops pol BY psmith WHENEVER NOT SUCCESSFUL;
```

27.2.16.6 How Oracle Data Mining Events Appear in the Audit Trail

The UNIFIED AUDIT TRAIL data dictionary view lists Oracle Data Mining audit events.

The following example shows how to query the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view for Data Mining audit events.

```
SELECT DBUSERNAME, ACTION NAME, SYSTEM PRIVILEGE USED, RETURN CODE,
OBJECT SCHEMA, OBJECT NAME, SQL TEXT
FROM UNIFIED AUDIT TRAIL;
DBUSERNAME ACTION NAME SYSTEM PRIVILEGE USED RETURN CODE
_____
OBJECT SCHEMA OBJECT NAME
-----
______
DMUSER1 CREATE MINING MODEL CREATE MINING MODEL
 dbms_data_mining.create_model(model_name => 'nb_model',
           mining_function => dbms_data_mining.classification,
           data_table_name => 'dm_data',
            case id column name => 'case id',
            target column name => 'target');
END;
DMUSER1 SELECT MINING MODEL
                                                   0
        NB MODEL
select prediction(nb model using *) from dual
```

```
SELECT MINING MODEL
                                                           40284
DMUSER2
DMUSER1 NB MODEL
select prediction(dmuser1.nb_model using *) from dual
DMUSER1 ALTER MINING MODEL
                                                              Λ
DMUSER1
         NB MODEL
BEGIN dbms data mining.rename model('nb model', 'nb model1'); END;
DMUSER2 ALTER MINING MODEL
                                                           40284
DMUSER1
                  NB MODEL
BEGIN dbms data mining.rename model('dmuser1.nb model1', 'nb model'); END;
DMUSER2 ALTER MINING MODEL
                                                           40284
DMUSER1
         NB MODEL
BEGIN dbms_data_mining.rename_model('dmuser1.nb model1', 'nb model'); END;
```

27.2.17 Auditing Oracle Data Pump Events

You can use the CREATE AUDIT POLICY statement to audit Oracle Data Pump.

- About Auditing Oracle Data Pump Events
 The CREATE AUDIT POLICY statement COMPONENT clause must be set to DATAPUMP to create
 Oracle Data Pump unified audit policies.
- Oracle Data Pump Unified Audit Trail Events
 The unified audit trail can capture Oracle Data Pump events.
- Configuring a Unified Audit Policy for Oracle Data Pump

 The ACTIONS COMPONENT clause in the CREATE AUDIT POLICY statement can be used to create an Oracle Data Pump event unified audit policy.
- Example: Auditing Oracle Data Pump Import Operations
 The CREATE AUDIT POLICY statement can audit Oracle Data Pump import operations.
- Example: Auditing All Oracle Data Pump Operations

 The CREATE AUDIT POLICY statement can audit all Oracle Data Pump operations.
- How Oracle Data Pump Audited Events Appear in the Audit Trail
 The UNIFIED_AUDIT_TRAIL data dictionary view lists Oracle Data Pump audited events.

27.2.17.1 About Auditing Oracle Data Pump Events

The CREATE AUDIT POLICY statement COMPONENT clause must be set to DATAPUMP to create Oracle Data Pump unified audit policies.

You can audit Data Pump export (expdp) and import (impdp) operations.

As with all unified auditing, you must have the AUDIT_ADMIN role before you can audit Oracle Data Pump events.

To access the audit trail, query the ${\tt UNIFIED_AUDIT_TRAIL}$ data dictionary view. The Data Pump-specific columns in this view begin with ${\tt DP}$.

Oracle Database records the Oracle Data Pump record before the worker process has determined or dispatched the actual workload. Therefore, there is no success or failure code that is captured in the audit record. A return code of 0 is expected behavior irrespective of the success or failure of the Data Pump job. Additionally, because Data Pump is restartable, reports on the success and failure status of the export or import operations might not be feasible to obtain.

Related Topics

Oracle Database Utilities

27.2.17.2 Oracle Data Pump Unified Audit Trail Events

The unified audit trail can capture Oracle Data Pump events.

The unified audit trail captures information about both export (expdp) and import (impdp) operations.

27.2.17.3 Configuring a Unified Audit Policy for Oracle Data Pump

The ACTIONS COMPONENT clause in the CREATE AUDIT POLICY statement can be used to create an Oracle Data Pump event unified audit policy.

Use the following syntax to create a unified audit policy for Oracle Data Pump:

```
CREATE AUDIT POLICY policy_name
ACTIONS COMPONENT=DATAPUMP { EXPORT | IMPORT | ALL };
```

For example:

```
CREATE AUDIT POLICY audit_dp_export_pol ACTIONS COMPONENT=DATAPUMP EXPORT;
```

You can build more complex policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.17.4 Example: Auditing Oracle Data Pump Import Operations

The CREATE AUDIT POLICY statement can audit Oracle Data Pump import operations.

Example 27-32 shows how to audit all Oracle Data Pump import operations.

Example 27-32 Auditing Oracle Data Pump Import Operations

```
CREATE AUDIT POLICY audit_dp_import_pol
ACTIONS COMPONENT=DATAPUMP IMPORT;

AUDIT POLICY audit_dp_import_pol;
```

27.2.17.5 Example: Auditing All Oracle Data Pump Operations

The CREATE AUDIT POLICY statement can audit all Oracle Data Pump operations.

Example 27-33 shows how to audit both Oracle Database Pump export and import operations.

Example 27-33 Auditing All Oracle Data Pump Operations

```
CREATE AUDIT POLICY audit_dp_all_pol
ACTIONS COMPONENT=DATAPUMP ALL;

AUDIT POLICY audit_dp_all_pol BY SYSTEM;
```



27.2.17.6 How Oracle Data Pump Audited Events Appear in the Audit Trail

The <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view lists Oracle Data Pump audited events.

The DP_* columns of the UNIFIED_AUDIT_TRAIL view show Oracle Data Pump-specific audit data. For example:

```
SELECT DP TEXT PARAMETERS1, DP BOOLEAN PARAMETERS1 FROM UNIFIED AUDIT TRAIL
WHERE AUDIT TYPE = 'DATAPUMP';
DP TEXT PARAMETERS1
                                        DP BOOLEAN PARAMETERS1
______
MASTER TABLE: "SCOTT"."SYS EXPORT TABLE 01", MASTER_ONLY: FALSE,
JOB TYPE: EXPORT,
                                        DATA ONLY: FALSE,
                                METADATA_ONLY: FALSE,
DUMPFILE_PRESENT: TRUE,
JOB_RESTARTED: FALSE
METADATA JOB MODE: TABLE EXPORT,
JOB VERSION: 19.1.0.0,
ACCESS METHOD: DIRECT PATH,
DATA OPTIONS: 0,
DUMPER DIRECTORY: NULL
REMOTE LINK: NULL,
TABLE EXISTS: NULL,
```

(This output was reformatted for easier readability.)

PARTITION OPTIONS: NONE

Oracle Database records the Oracle Data Pump record before the worker process has determined or dispatched the actual workload. Therefore, there is no success or failure code that is captured in the audit record. A return code of 0 is expected behavior irrespective of the success or failure of the Data Pump job. Additionally, because Data Pump is restartable reports on the success and failure status of the export or import operations might not show as much data as desired.

27.2.18 Auditing Oracle SQL*Loader Direct Load Path Events

You can use the CREATE AUDIT POLICY statement to audit Oracle SQL*Loader direct load path events.

- About Auditing in Oracle SQL*Loader Direct Path Load Events
 You must have the AUDIT ADMIN role to audit Oracle SQL*Loader direct path events.
- Oracle SQL*Loader Direct Load Path Unified Audit Trail Events
 The unified audit trail can capture SQL*Loader Direct Load Path events.
- Configuring a Unified Audit Trail Policy for Oracle SQL*Loader Direct Path Events
 The CREATE AUDIT POLICY statement ACTIONS COMPONENT clause can create unified audit
 policies for Oracle SQL*Loader direct path events.
- Example: Auditing Oracle SQL*Loader Direct Path Load Operations
 The CREATE AUDIT POLICY statement can audit Oracle SQL*Loader direct path load operations.
- How SQL*Loader Direct Path Load Audited Events Appear in the Audit Trail
 The UNIFIED_AUDIT_TRAIL data dictionary view lists SQL*Loader direct path load audited events.

27.2.18.1 About Auditing in Oracle SQL*Loader Direct Path Load Events

You must have the AUDIT ADMIN role to audit Oracle SQL*Loader direct path events.



To create SQL*Loader unified audit policies, you must set the CREATE AUDIT POLICY statement's COMPONENT clause to DIRECT_LOAD. You can audit direct path load operations only, not other SQL*Loader loads, such as conventional path loads.

To access the audit trail, you can query the <code>DIRECT_PATH_NUM_COLUMNS_LOADED</code> column in the <code>UNIFIED AUDIT TRAIL</code> data dictionary view.



Oracle Database Utilities for detailed information about Oracle SQL*Loader

27.2.18.2 Oracle SQL*Loader Direct Load Path Unified Audit Trail Events

The unified audit trail can capture SQL*Loader Direct Load Path events.

The unified audit trail captures information about direct path loads that SQL*Loader performs (that is, when you set direct=true on the SQL*Loader command line or in the SQL*Loader control file).

It also audits Oracle Call Interface (OCI) programs that use the direct path API.

See Also:

Oracle Database Utilities for detailed information about direct path loads in Oracle SOL*Loader

27.2.18.3 Configuring a Unified Audit Trail Policy for Oracle SQL*Loader Direct Path Events

The CREATE AUDIT POLICY statement ACTIONS COMPONENT clause can create unified audit policies for Oracle SQL*Loader direct path events.

Use the following syntax to create an Oracle SQL*Loader unified audit policy:

```
CREATE AUDIT POLICY policy_name
ACTIONS COMPONENT=DIRECT LOAD { LOAD };
```

For example:

```
CREATE AUDIT POLICY audit_sqlldr_pol ACTIONS COMPONENT=DIRECT_LOAD LOAD;
```

You can build more complex policies, such as those that include conditions. Remember that after you create the policy, you must use the AUDIT statement to enable it.

Related Topics

Syntax for Creating a Unified Audit Policy
 To create a unified audit policy, you must use the CREATE AUDIT POLICY statement.

27.2.18.4 Example: Auditing Oracle SQL*Loader Direct Path Load Operations

The CREATE AUDIT POLICY statement can audit Oracle SQL*Loader direct path load operations.

Example 27-32 shows how to audit SQL*Loader direct path load operations.

Example 27-34 Auditing Oracle SQL*Loader Direct Path Load Operations

```
CREATE AUDIT POLICY audit_sqlldr_load_pol
ACTIONS COMPONENT=DIRECT_LOAD LOAD;

AUDIT POLICY audit sqlldr load pol;
```

27.2.18.5 How SQL*Loader Direct Path Load Audited Events Appear in the Audit Trail

The UNIFIED_AUDIT_TRAIL data dictionary view lists SQL*Loader direct path load audited events.

The DIRECT_PATH_NUM_COLUMNS_LOADED column of the UNIFIED_AUDIT_TRAIL view shows the number of columns that were loaded using the SQL*Loader direct path load method. For example:

27.2.19 Auditing Only Top-Level Statements

You can audit top-level SQL or PL/SQL statements to limit the volume of audit records.

- About Auditing Only Top-Level SQL Statements
 A top-level statement is a statement that is executed directly by a user, not a statement that is run from within a PL/SQL procedure.
- Configuring a Unified Audit Policy to Capture Only Top-Level Statements
 The ONLY TOPLEVEL clause in the CREATE AUDIT POLICY statement enables you to audit only the SQL statements that are directly issued by an end user by honoring the audit configuration in the audit policy.
- Example: Auditing Top-Level Statements
 The CREATE AUDIT POLICY statement can include or exclude top-level statement audit records in the unified audit trail for any user.
- Example: Comparison of Top-Level SQL Statement Audits
 You can generate top-level SQL statement audit records from SQL statements that are run directly in SQL or from within a PL/SQL procedure.
- How the Unified Audit Trail Captures Top-Level SQL Statements
 The ONLY TOPLEVEL clause has no impact on the output for an individual unified audit trail record.

27.2.19.1 About Auditing Only Top-Level SQL Statements

A top-level statement is a statement that is executed directly by a user, not a statement that is run from within a PL/SQL procedure.

Consider auditing top-level statements from all users, including user SYS to reduce the volume of audit. The feature audits all the user-initiated actions and ignores the recursive SQL statements. For example, auditing the <code>DBMS_STATS.GATHER_DATABASE_STATS</code> SQL statement can generate over 200,000 individual audit records and by adding top-level this reduces to a single audit record.

27.2.19.2 Configuring a Unified Audit Policy to Capture Only Top-Level Statements

The ONLY TOPLEVEL clause in the CREATE AUDIT POLICY statement enables you to audit only the SQL statements that are directly issued by an end user by honoring the audit configuration in the audit policy.

To find policies that include the <code>ONLY TOPLEVEL</code> clause, query the <code>AUDIT_ONLY_TOPLEVEL</code> column of the <code>AUDIT UNIFIED POLICIES</code> data dictionary view.

Use the following syntax to create a unified audit policy that audits only top-level SQL statements.

```
CREATE AUDIT POLICY policy_name all_existing_options
ONLY TOPLEVEL;
```

For example, to limit the audit trail to top-level instances of the SELECT statement on the HR.EMPLOYEES table:

```
CREATE AUDIT POLICY actions_on_hr_emp_pol
ACTIONS SELECT ON HR.EMPLOYEES
ONLY TOPLEVEL;
```

27.2.19.3 Example: Auditing Top-Level Statements

The CREATE AUDIT POLICY statement can include or exclude top-level statement audit records in the unified audit trail for any user.

The following example shows an audit policy that will capture all top level statements executed by user SYS.

Example 27-35 Example: Auditing Top-Level Statements Executed by User SYS

```
CREATE AUDIT POLICY actions_all_pol ACTIONS ALL ONLY TOPLEVEL;

AUDIT POLICY actions all pol BY SYS;
```

27.2.19.4 Example: Comparison of Top-Level SQL Statement Audits

You can generate top-level SQL statement audit records from SQL statements that are run directly in SQL or from within a PL/SQL procedure.

This example shows how generating audit records differs when you access a view outside a PL/SQL procedure as opposed to accessing the view inside the PL/SQL procedure. The output illustrates the difference in volume in audit records that are generated from the two different audit policies.

- 1. Log in to the database instance as user SYS with the SYSDBA administrative privilege. In a multitenant environment, log in to the PDB. To find the available PDBs in a CDB, log in to the CDB root container and then query the PDB_NAME column of the DBA_PDBS data dictionary view. To check the current container, run the show con name command.
- **2.** Create the following procedure:

```
CREATE OR REPLACE PROCEDURE proc1 AS
cnt number;
BEGIN
   SELECT COUNT(*) INTO CNT FROM SYS.DBA_USERS WHERE USER_ID=9999;
END;
//
```

3. Create the and enable following audit policy to capture top-level actions:

```
CREATE AUDIT POLICY toplevel_pol ACTIONS ALL ONLY TOPLEVEL; AUDIT POLICY toplevel pol;
```

4. Execute the following query to generate an audit record and to access the SYS.DBA_USERS view outside of the proc1 procedure that you just created:

```
SELECT /* TOPLEVEL */ COUNT(*) FROM SYS.DBA USERS WHERE USER ID=0000;
```

The output should be as follows:

```
COUNT(*)
```

5. Execute the proc1 procedure that you created earlier, to access the SYS.DBA_USERS view again, but from within a procedure.

```
EXEC proc1;
```

6. Query the UNIFIED AUDIT TRAIL data dictionary view as follows:

```
SELECT ACTION_NAME, OBJECT_SCHEMA,OBJECT_NAME,STATEMENT_ID,ENTRY_ID,
UNIFIED_AUDIT_POLICIES,SQL_TEXT
FROM UNIFIED_AUDIT_TRAIL
ORDER BY EVENT TIMESTAMP;
```

Output similar to the following appears:

1



TOPLEVEL POL COMMIT 3 2 TOPLEVEL POL COMMIT 4 3 TOPLEVEL POL SELECT SYS USER\$ TOPLEVEL POL select /* toplevel */ count(*) from sys.dba users where user id=0000 SELECT 5 5 RESOURCE_GROUP_MAPPING\$ TOPLEVEL POL select /* toplevel */ count(*) from sys.dba users where user _id=0000 SELECT SYS 5 TS\$ TOPLEVEL POL select /* toplevel */ count(*) from sys.dba_users where user id=0000 SELECT SYS 5 TS\$ 7 TOPLEVEL POL select /* toplevel */ count(*) from sys.dba users where user _id=0000 SELECT SYS TS\$ TOPLEVEL_POL select /* toplevel */ count(*) from sys.dba users where user _id=0000 SELECT SYS 5 PROFNAME\$ TOPLEVEL POL select /* toplevel */ count(*) from sys.dba users where user _id=0000 SELECT SYS 5 USER ASTATUS MAP 10 TOPLEVEL POL select /* toplevel */ count(*) from sys.dba users where user _id=0000 SELECT SYS



```
5
                                                     11
PROFILE$
TOPLEVEL POL
select /* toplevel */ count(*) from sys.dba users where user
SELECT
                     SYS
                                           5
PROFILE$
                                                     12
TOPLEVEL POL
select /* toplevel */ count(*) from sys.dba users where user
id=0000
SELECT
                     SYS
DBA USERS
                                           5
                                                     13
TOPLEVEL POL
select /* toplevel */ count(*) from sys.dba users where user
id=0000
EXECUTE
                     SYS
PROC1
                                                     14
TOPLEVEL POL
BEGIN proc1; END;
14 rows selected.
```

7. Disable and then drop the toplevel pol audit policy.

```
NOAUDIT POLICY toplevel_pol;
DROP AUDIT POLICY toplevel_pol;
```

8. Create and enable a new audit policy to capture all actions.

```
CREATE AUDIT POLICY recursive_pol ACTIONS ALL;
AUDIT POLICY recursive pol;
```

9. Clean up the audit trail.

```
DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,FALSE
);
```

10. Execute the following query to generate an audit record and to access the SYS.DBA_USERS view outside of the proc1 procedure:

```
SELECT /* TOPLEVEL */ COUNT(*) FROM SYS.DBA USERS WHERE USER ID=0000;
```

The output should be as follows:

```
COUNT (*)
```

11. Execute the proc1 procedure to access the SYS.DBA_USERS again, but from within the proc1 procedure.

```
EXEC proc1;
```

12. Query the UNIFIED_AUDIT_TRAIL data dictionary view as follows:

```
SELECT ACTION_NAME, OBJECT_SCHEMA,OBJECT_NAME,STATEMENT_ID,ENTRY_ID,
UNIFIED_AUDIT_POLICIES,SQL_TEXT
FROM UNIFIED_AUDIT_TRAIL
ORDER BY EVENT_TIMESTAMP;
```

Output similar to the following should appear:

ACTION_N	IAME OBJECT	_SCHEMA	
OBJECT_N	AME	UNIFIED_AUDIT_POLICIES	STATEMENT_ID
	ID SQL_TEXT		
LOGON			
	1	RECURSIVE_POL	1
ALTER SE	SSION		1
	2 ALTER SESSION SE	RECURSIVE_POL T TIME_ZONE='-07:00'	1
COMMIT		RECURSIVE POL	3
COMMIT	3		-
COMMIT	4	RECURSIVE_POL	4
SELECT	SYS		
USER\$	5 select /* toplev	RECURSIVE_POL el */ count(*) from sys.dba_users	5 where user
SELECT	_id=0000		
	_GROUP_MAPPING\$	<pre>RECURSIVE_POL el */ count(*) from sys.dba_users</pre>	5 where user
SELECT	SYS		
TS\$	7 select /* toplev _id=0000	RECURSIVE_POL el */ count(*) from sys.dba_users	5 where user
SELECT	SYS		
TS\$	8 select /* toplev id=0000	RECURSIVE_POL el */ count(*) from sys.dba_users	5 where user
SELECT	SYS		
TS\$		<pre>RECURSIVE_POL el */ count(*) from sys.dba_users</pre>	5 where user

SELECT			SYS		_
PROFNAME		select / _id=0000	* tople	RECURSIVE_POL vel */ count(*) from sys.dba_users where us	5 ser
SELECT USER_AST			SYS * tople	RECURSIVE_POL vel */ count(*) from sys.dba_users where us	5 ser
SELECT PROFILE\$		select / _id=0000	SYS * tople	RECURSIVE_POL vel */ count(*) from sys.dba_users where u	5 ser
SELECT PROFILE\$		select /	SYS * tople	<pre>RECURSIVE_POL vel */ count(*) from sys.dba_users where us</pre>	5 ser
SELECT DBA_USER		select / _id=0000	SYS * tople	<pre>RECURSIVE_POL vel */ count(*) from sys.dba_users where us</pre>	5 ser
SELECT USER\$	15	SELECT C	SYS	RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
	_	_		RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
SELECT TS\$	17	SELECT C	SYS	RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
SELECT TS\$	18	SELECT C	SYS	RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
SELECT TS\$	19	SELECT C	SYS	RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
SELECT PROFNAME		SELECT C	SYS	RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
		JS_MAP SELECT C		RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7
SELECT PROFILE\$		SELECT C	SYS	RECURSIVE_POL FROM SYS.DBA_USERS WHERE USER_ID=9999	7



```
SELECT
                     SYS
                                                                            7
PROFILE$
                                RECURSIVE POL
        23 SELECT COUNT(*) FROM SYS.DBA USERS WHERE USER ID=9999
SELECT
                     SYS
                                                                            7
DBA USERS
                                RECURSIVE POL
        24 SELECT COUNT(*) FROM SYS.DBA USERS WHERE USER ID=9999
EXECUTE
                     SYS
                                RECURSIVE POL
                                                                            7
PROC1
        25 BEGIN proc1; END;
25 rows selected.
```

The output in this query generates 25 records, as opposed to the 14 that were generated earlier.

13. Disable and remove the recursive pol policy.

```
NOAUDIT POLICY recursive_pol;
DROP AUDIT POLICY recursive pol;
```

27.2.19.5 How the Unified Audit Trail Captures Top-Level SQL Statements

The ONLY TOPLEVEL clause has no impact on the output for an individual unified audit trail record.

The only effect that <code>ONLY TOPLEVEL</code> has on a policy is to limit the number of records generated for the given unified audit policy.

27.2.20 Unified Audit Policies or AUDIT Settings in a Multitenant Environment

In a multitenant environment, you can create unified audit policies for individual PDBs and in the root.

- About Local, CDB Common, and Application Common Audit Policies
 An audit policy can be either a local audit policy, a CDB common audit policy, or an application common audit policy.
- Traditional Auditing in a Multitenant Environment
 In traditional auditing (not unified auditing), the AUDIT and NOAUDIT statements can audit statements and privileges in a multitenant environment.
- Configuring a Local Unified Audit Policy or Common Unified Audit Policy
 The CONTAINER clause is specific to multitenant environment use for the CREATE AUDIT
 POLICY statement.
- Example: Local Unified Audit Policy
 The CREATE AUDIT POLICY statement can create a local unified audit policy in either the
- Example: CDB Common Unified Audit Policy
 The CREATE AUDIT POLICY statement can create a CDB common unified audit policy.

root or a PDB.

- Example: Application Common Unified Audit Policy
 For application container common unified audit policies, you can audit action options and system privilege options, and refer to common objects and roles.
- How Local or Common Audit Policies or Settings Appear in the Audit Trail
 You can query unified audit policy views from either the root or the PDB in which the action
 occurred.

27.2.20.1 About Local, CDB Common, and Application Common Audit Policies

An audit policy can be either a local audit policy, a CDB common audit policy, or an application common audit policy.

This applies to both unified audit policies and policies that are created using the AUDIT SQL statement.

• Local audit policy. This type of policy can exist in either the root (CDB or application) or the PDB (CDB or application). A local audit policy that exists in the root can contain object audit options for both local and common objects. Both local and common users who have been granted the AUDIT_ADMIN role can enable local policies: local users from their PDBs and common users from the root or the PDB to which they have privileges. You can enable a local audit policy for both local and common users and roles.

You can create local audit policies for application local objects and application local roles, as well as system action options and system privilege options. You cannot enforce a local audit policy for a common user across all containers, nor can you enforce a common audit policy for a local user.

• CDB common audit policy. This type of policy is available to all PDBs in the multitenant environment. Only common users who have been granted the AUDIT_ADMIN role can create and maintain common audit policies. You can enable common audit policies only for common users. You must create common audit policies only in the root. This type of policy can contain object audit options of only common objects, and be enabled only for common users. You can enable a common audit policy for common users and roles only.

You cannot enforce a common audit policy for a local user across all containers.

• Application common audit policy. Similar to CDB common audit policies, this type of policy is available to all PDBs in the multitenant environment. You can create common audit policies for application common objects and application common roles, as well as system action options and system privilege options. You can only create this type of policy in the application root container, but you can enable it on both application common users and CDB common users. If you want to audit objects, then ensure that these objects are application common objects. You can determine whether an object is an application common object by querying the SHARING column of the DBA OBJECTS data dictionary view.

By default, audit policies are local to the current PDB, for both CDB and application scenarios.

The following table explains how audit policies apply in different multitenant environments.

Table 27-19 How Audit Policies Apply to the CDB Root, Application Root, and Individual PDBs

Audit Option Type	CDB Root	Application Root	Individual PDB
Common audit statement or audit policy		Applies to CDB common users	Applies to CDB common users



Table 27-19 (Cont.) How Audit Policies Apply to the CDB Root, Application Root, and Individual PDBs

Audit Option Type	CDB Root	Application Root	Individual PDB
Application container common audit statement or audit policy	Not applicable	 Applies to CDB common users and are valid for the current application container only Applies to application container common users 	 Applies to CDB common users and are valid for this application container only Applies to application common users
Local audit statement or audit policy	Local configurations not allowed	Local configurations not allowed	 Applies to CDB common users Applies to application common users

27.2.20.2 Traditional Auditing in a Multitenant Environment

In traditional auditing (not unified auditing), the AUDIT and NOAUDIT statements can audit statements and privileges in a multitenant environment.

To configure the audit policy to be either a local audit policy or a common audit policy, you must include the CONTAINER clause, as you normally do for other SQL creation or modification statements. If you want to audit an application container, then you can audit SQL statement and system privileges performed by local and common users and roles. The audit record will be created in the container in which the action was performed.

• If you want to apply the AUDIT or NOAUDIT statement to the current CDB or application PDB, then in this PDB, you must set CONTAINER to CURRENT. For example:

AUDIT DROP ANY TABLE BY SYSTEM BY ACCESS CONTAINER = CURRENT;

• If you want to apply the AUDIT or NOAUDIT statement to the entire multitenant environment, then in the CDB root, then you must set CONTAINER to ALL. For an application container, you would set it in the application root. For example:

AUDIT DROP ANY TABLE BY SYSTEM BY ACCESS CONTAINER = ALL;

To find if a traditional audit option is designed for use in an application container, perform a join query with the <code>DBA_OBJ_AUDIT_OPTS</code> and <code>DBA_OBJECTS</code> data dictionary views, by using the <code>OWNER</code> and <code>OBJECT_NAME</code> columns in both views, and the <code>APPLICATION</code> column in <code>DBA_OBJECTS</code>.

See Also

Oracle Database SQL Language Reference for more information about the traditional AUDIT and NOAUDIT SQL statements



27.2.20.3 Configuring a Local Unified Audit Policy or Common Unified Audit Policy

The CONTAINER clause is specific to multitenant environment use for the CREATE AUDIT POLICY statement.

To create a local or common (CDB or application) unified audit policy in either the CDB environment or an application container environment, include the CONTAINER clause in the CREATE AUDIT POLICY statement.

Use the following syntax to create a local or common unified audit policy:

```
CREATE AUDIT POLICY policy_name
  action1 [,action2 ]
  [CONTAINER = {CURRENT | ALL}];
```

In this specification:

- CURRENT sets the audit policy to be local to the current PDB.
- ALL makes the audit policy a common audit policy, that is, available to the entire multitenant environment.

For example, for a common unified audit policy:

```
CREATE AUDIT POLICY dict_updates
ACTIONS UPDATE ON SYS.USER$,
DELETE ON SYS.USER$,
UPDATE ON SYS.LINK$,
DELETE ON SYS.LINK$
CONTAINER = ALL;
```

Note the following:

- You can set the CONTAINER clause for the CREATE AUDIT POLICY statement but not for ALTER AUDIT POLICY or DROP AUDIT POLICY. If you want to change the scope of an existing unified audit policy to use this setting, then you must drop and re-create the policy.
- For AUDIT statements, you can set the CONTAINER clause for audit settings only if you have an Oracle database that has not been migrated to the Release 12.x and later audit features. You cannot use the CONTAINER clause in an AUDIT statement that is used to enable a unified audit policy.
- If you are in a PDB, then you can only set the CONTAINER clause to CURRENT, not ALL. If you omit the setting while in the PDB, then the default is CONTAINER = CURRENT.
- If you are in the root, then you can set the CONTAINER clause to either CURRENT if you want the policy to apply to the root only, or to ALL if you want the policy to apply to the entire CDB. If you omit the CONTAINER clause, then default is CONTAINER = CURRENT.
- For objects:
 - Common audit policies can have common objects only and local audit policies can have both local objects and common objects.
 - You cannot set CONTAINER to ALL if the objects involved are local. They must be common objects.
- For privileges:



- You can set the CONTAINER to CURRENT (or omit the CONTAINER clause) if the user accounts involved are a mixture of local and common accounts. This creates a local audit configuration that applies only to the current PDB.
- You cannot set CONTAINER to ALL if the users involved are local users. They must be common users.
- If you set CONTAINER to ALL and do not specify a user list (using the BY clause in the AUDIT statement), then the configuration applies to all common users in each PDB.
- For application containers, you can run a common unified audit policy from the application container script that is used for application install, upgrade, patch, and uninstall operations.
 To do so:
 - 1. Create a common unified audit policy in the application container root, and set this policy to CONTAINER = ALL. Alternatively, you can include this policy in the script that is described in this next step.
 - 2. Create a custom version of the script you normally would use to install, upgrade, patch, or uninstall Oracle Database.
 - 3. Within this script, include the SQL statements that you want to audit within the following lines:

```
ALTER PLUGGABLE DATABASE APPLICATION BEGIN INSTALL

List SQL statements here. Separate each statement with a semi-colon.

ALTER PLUGGABLE DATABASE APPLICATION END INSTALL
```

If you include the unified audit policy in the script, then ensure that you include both the CREATE AUDIT POLICY and AUDIT POLICY statements.

After the audit policy is created and enabled, all user access to the application common objects is audited irrespective of whether the audit policy is defined in the database or from the script.

To audit application install, upgrade, patch, and uninstall operations locally in an application root or an application PDB, follow a procedure similar to the preceding procedure for common unified audit policies, but synchronize the application PDB afterward. For example:

ALTER PLUGGABLE DATABASE APPLICATION application name SYNC;

Related Topics

Oracle Multitenant Administrator's Guide

27.2.20.4 Example: Local Unified Audit Policy

The CREATE AUDIT POLICY statement can create a local unified audit policy in either the root or a PDB.

When you create a local unified audit policy in the root, it only applies to the root and not across the multitenant environment.

The following example shows a local unified audit policy that has been created by the common user c##sec admin from a PDB and applied to common user c##hr admin.



Example 27-36 Local Unified Audit Policy

```
CONNECT c##sec_admin@hrpdb
Enter password: password
Connected.

CREATE AUDIT POLICY table_privs
PRIVILEGES CREATE ANY TABLE, DROP ANY TABLE
CONTAINER = CURRENT;

AUDIT POLICY table privs BY c##hr admin;
```

27.2.20.5 Example: CDB Common Unified Audit Policy

The CREATE AUDIT POLICY statement can create a CDB common unified audit policy.

Example 27-37 shows a common unified audit policy that has been created by the common user c##sec admin from the root and applied to common user c##hr admin.

Example 27-37 Common Unified Audit Policy

```
CONNECT c##sec_admin
Enter password: password
Connected.

CREATE AUDIT POLICY admin_pol
   ACTIONS CREATE TABLE, ALTER TABLE, DROP TABLE
   ROLES c##hr_mgr, c##hr_sup
   CONTAINER = ALL;

AUDIT POLICY admin pol BY c##hr admin;
```

27.2.20.6 Example: Application Common Unified Audit Policy

For application container common unified audit policies, you can audit action options and system privilege options, and refer to common objects and roles.

You can create the application common audit policy only from the application root, and enable the policy for both application common users and CDB common users.

The following example shows how to create a policy that audits the application common user SYSTEM for the application container app_pdb. The audit policy audits SELECT actions on the SYSTEM.utils_tab table and on DROP TABLE actions on any of the PDBs in the container database, including the CDB root. The policy also audits the use of the SELECT ANY TABLE system privilege across all containers.

Example 27-38 Application Common Unified Audit Policy

```
CONNECT c##sec_admin@app_pdb
Enter password: password
Connected.

CREATE AUDIT POLICY app_pdb_admin_pol
ACTIONS SELECT ON hr_app_cdb.utils_tab, DROP TABLE
PRIVILEGES SELECT ANY TABLE
```



```
CONTAINER = ALL;

AUDIT POLICY app pdb admin pol by SYSTEM, c##hr admin;
```

In the preceding example, setting CONTAINER to ALL applies the policy only to all the relevant object accesses in the application root and on all the application PDBs that belong to the application root. It does not apply the policy outside this scope.

27.2.20.7 How Local or Common Audit Policies or Settings Appear in the Audit Trail

You can query unified audit policy views from either the root or the PDB in which the action occurred.

You can perform the following types of gueries:

- Audit records from all PDBs. The audit trail reflects audited actions that have been performed in the PDBs. For example, if user lbrown in PDB1 performs an action that has been audited by either a common or a local audit policy, then the audit trail will capture this action. The DBID column in the UNIFIED_AUDIT_TRAIL data dictionary view indicates the PDB in which the audited action takes place and to which the policy applies. If you want to see audit records from all PDBs, you should query the CDB_UNIFIED_AUDIT_TRAIL data dictionary view from the root.
- Audit records from common audit policies. This location is where the common audit policy results in an audit record. The audit record can be generated anywhere in the multitenant environment—the root or the PDBs, depending on where the action really occurred. For example, the common audit policy fga_pol audits the EXECUTE privilege on the DBMS_FGA PL/SQL package, and if this action occurs in PDB1, then the audit record is generated in PDB1 and not in the root. Hence, the audit record can be seen in PDB1.

You can query the UNIFIED_AUDIT_TRAIL data dictionary view for the policy from either the root or a PDB if you include a WHERE clause for the policy name (for example, WHERE UNIFIED_AUDIT_POLICIES = 'FGA_POL').

The following example shows how to find the results of a common unified audit policy:

27.2.21 Altering Unified Audit Policies

You can use the ALTER AUDIT POLICY statement to modify a unified audit policy.

About Altering Unified Audit Policies
 You can change most properties in a unified audit policy, except for its CONTAINER setting.

Altering a Unified Audit Policy

The ALTER AUDIT POLICY statement can modify a unified audit policy.

- Example: Altering a Condition in a Unified Audit Policy
 The ALTER AUDIT POLICY statement can alter conditions in unified audit policies.
- Example: Altering an Oracle Label Security Component in a Unified Audit Policy
 The ALTER AUDIT POLICY statement can alter Oracle Label Security components in an audit policy.
- Example: Altering Roles in a Unified Audit Policy
 The ALTER AUDIT POLICY statement can alter roles in a unified audit policy.
- Example: Dropping a Condition from a Unified Audit Policy
 The ALTER AUDIT POLICY statement can drop a condition from a unified audit policy.
- Example: Altering an Existing Unified Audit Policy Top-Level Statement Audits
 The ALTER AUDIT POLICY statement can modify an existing unified audit policy so that the unified audit trail captures top-level SQL statements only.

27.2.21.1 About Altering Unified Audit Policies

You can change most properties in a unified audit policy, except for its CONTAINER setting.

You cannot alter unified audit policies in a multitenant environment. For example, you cannot turn a common unified audit policy into a local unified audit policy.

To find existing unified audit policies, query the AUDIT_UNIFIED_POLICIES data dictionary view. If you want to find only the enabled unified audit policies, then query the AUDIT_UNIFIED_ENABLED_POLICIES view. You can alter both enabled and disabled audit policies. If you alter an enabled audit policy, it remains enabled after you alter it.

After you alter an object unified audit policy, the new audit settings take place immediately, for both the active and subsequent user sessions. If you alter system audit options, or audit conditions of the policy, then they are activated for new user sessions, but not the current user session.

27.2.21.2 Altering a Unified Audit Policy

The ALTER AUDIT POLICY statement can modify a unified audit policy.

 Use the following syntax to alter a unified audit policy, you use the ALTER AUDIT POLICY statement.

```
ALTER AUDIT POLICY policy_name

[ADD [privilege_audit_clause] [action_audit_clause]
        [role_audit_clause] [ONLY TOPLEVEL] ]

[DROP [privilege_audit_clause] [action_audit_clause]
        [role_audit_clause] [ONLY TOPLEVEL]]

[CONDITION {DROP | audit_condition_EVALUATE_PER {STATEMENT|SESSION|INSTANCE}}]
```

In this specification:

- ADD enables you to alter the following the following settings:
 - privilege_audit_clause describes privilege-related audit options. See Auditing
 System Privileges for details. The detailed syntax for configuring privilege audit options is as follows:

```
ADD privilege audit clause := PRIVILEGES privilege1 [, privilege2]
```

action_audit_clause and standard_actions describe object action-related audit options. See Auditing Object Actions. The syntax is as follows:

 role_audit_clause enables you to add or drop the policy for roles. See Auditing Roles. The syntax is:

```
ADD role audit clause := ROLES role1 [, role2]
```

- ONLY TOPLEVEL includes in the unified audit trail only the top-level SQL statements that are affected by this policy.
- DROP enables you to drop the same components that are described for the ADD clause. For example:

```
DROP role audit clause := ROLES role1 [, role2 ONLY TOPLEVE1]
```

• CONDITION {DROP... enables you to add or drop a condition for the policy. If you are altering an existing condition, then you must include the EVALUATE PER clause with the condition. See Creating a Condition for a Unified Audit Policy. The syntax is:

```
CONDITION 'audit_condition := function operation value_list' EVALUATE PER {STATEMENT|SESSION|INSTANCE}
```

If you want to drop a condition, then omit the condition definition and the EVALUATE PER clause. For example:

CONDITION DROP

27.2.21.3 Example: Altering a Condition in a Unified Audit Policy

The ALTER AUDIT POLICY statement can alter conditions in unified audit policies.

Example 27-39 shows how to change a condition in an existing unified audit policy.

Example 27-39 Altering a Condition in a Unified Audit Policy

```
ALTER AUDIT POLICY orders_unified_audpol
ADD ACTIONS INSERT ON SCOTT.EMP
CONDITION 'SYS_CONTEXT(''ENTERPRISE'', ''GROUP'') = ''ACCESS_MANAGER'''
EVALUATE PER SESSION;
```

27.2.21.4 Example: Altering an Oracle Label Security Component in a Unified Audit Policy

The ALTER AUDIT POLICY statement can alter Oracle Label Security components in an audit policy.

Example 27-40 shows how to alter an Oracle Label Security component in an audit policy.

Example 27-40 Altering an Oracle Label Security Component in a Unified Audit Policy

ALTER AUDIT POLICY audit_ols
ADD ACTIONS SELECT ON HR.EMPLOYEES
ACTIONS COMPONENT=OLS DROP POLICY, DISABLE POLICY, REMOVE POLICY;

27.2.21.5 Example: Altering Roles in a Unified Audit Policy

The ALTER AUDIT POLICY statement can alter roles in a unified audit policy.

Example 27-41 shows how to add roles to a common unified audit policy.

Example 27-41 Altering Roles in a Unified Audit Policy

CONNECT c##sec_admin
Enter password: password
Connected.

ALTER AUDIT POLICY RoleConnectAudit
ADD ROLES c##role1, c##role2;

27.2.21.6 Example: Dropping a Condition from a Unified Audit Policy

The ALTER AUDIT POLICY statement can drop a condition from a unified audit policy.

Example 27-42 shows how to drop a condition from an existing unified audit policy.

Example 27-42 Dropping a Condition from a Unified Audit Policy

ALTER AUDIT POLICY orders_unified_audpol CONDITION DROP;

27.2.21.7 Example: Altering an Existing Unified Audit Policy Top-Level Statement Audits

The ALTER AUDIT POLICY statement can modify an existing unified audit policy so that the unified audit trail captures top-level SQL statements only.

The following example shows how to modify the orders_unified_audpol policy to capture only top-level SQL statements.

Example 27-43 Altering an Existing Unified Audit Policy to Audit for Top-Level Statements

ALTER AUDIT POLICY orders_unified_audpol ADD ONLY TOPLEVEL;

Similarly, to remove the top-level SQL statement audit, use the DROP clause:

ALTER AUDIT POLICY orders_unified_audpol DROP ONLY TOPLEVEL;

27.2.22 Enabling and Applying Unified Audit Policies to Users and Roles

You can use the AUDIT POLICY statement to enable and apply unified audit policies to users and roles.

About Enabling Unified Audit Policies

The AUDIT statement with the POLICY clause enables a unified audit policy, applying for all types of audit options, including object-level options.

Enabling a Unified Audit Policy

The AUDIT POLICY statement can enable a unified audit policy.

Example: Enabling a Unified Audit Policy

The AUDIT POLICY statement can enable a unified audit policy using conditions, such as WHENEVER NOT SUCCESSFUL.

27.2.22.1 About Enabling Unified Audit Policies

The AUDIT statement with the POLICY clause enables a unified audit policy, applying for all types of audit options, including object-level options.

The policy does not take effect until after the audited users (or users who have been granted the roles associated with the policy) log into the database instance. In other words, if you create and enable a policy while the audited users are logged in, then the policy cannot collect audit data; the users must log out and then log in again before auditing can begin. Once the session is set up with auditing for it, then the setting lasts as long as the user session and then ends when the session ends.

You can enable the audit policy for individual users or for roles. Enabling the audit policy for roles allows you to enable the policy for a group of users who have been directly granted the role. When the role has been directly granted to a new user, then the policy automatically applies to the user. When the role is revoked from a user, then the policy no longer applies to the user.

You can check the results of the audit by querying the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view. To find a list of existing unified audit policies, query the <code>AUDIT_UNIFIED_POLICIES</code> data dictionary view.

The AUDIT statement lets you specify the following optional additional settings:

Whether to apply the unified audit policy to one or more users or roles. To apply the
policy to one or more users or roles, including administrative users who log in with the
SYSDBA administrative privilege (such as SYS), use the BY clause. For example, to apply the
policy to users SYS and SYSTEM:

For example, to apply the policy to two users:

```
AUDIT POLICY role_connect_audit_pol BY SYS, SYSTEM;
```

To apply a policy to users who have been directly granted the DBA and CDB DBA roles:

```
AUDIT POLICY admin audit pol BY USERS WITH GRANTED ROLES DBA, CDB DBA;
```

• Whether to exclude users from the unified audit policy. To exclude users from the audit policy, include the EXCEPT clause.

For example:

```
AUDIT POLICY role connect audit pol EXCEPT rlee, jrandolph;
```

- Whether to create an audit record if the activity succeeds or fails. This method of auditing reduces the audit trail, helping you to focus on specific actions. This can aid in maintaining good database performance. Enter one of the following clauses:
 - WHENEVER SUCCESSFUL audits only successful executions of the user's activity.
 - WHENEVER NOT SUCCESSFUL audits only failed executions of the user's activity.
 Monitoring unsuccessful SQL statement can expose users who are snooping or acting maliciously, though most unsuccessful SQL statements are neither.

For example:



```
AUDIT POLICY role connect audit pol WHENEVER NOT SUCCESSFUL;
```

If you omit this clause, then both failed and successful user activities are written to the audit trail.

Note the following:

- The unified audit policy only can have either the BY, BY USERS WITH GRANTED ROLES, or the EXCEPT clause, but not more than one of these clauses for the same policy.
- If you run multiple AUDIT statements on the same unified audit policy but specify different BY users or different BY USERS WITH GRANTED ROLES roles, then Oracle Database audits all of these users or roles.
- If you run multiple AUDIT statements on the same unified audit policy but specify different EXCEPT users, then Oracle Database uses the last exception user list, not any of the users from the preceding lists. This means the effect of the earlier AUDIT POLICY ... EXCEPT statements are overridden by the latest AUDIT POLICY ... EXCEPT statement.
- You cannot use the EXCEPT clause for roles. It applies to users only.
- You can only enable common unified audit policies for common users or roles.
- In a multitenant environment, you can enable a common audit policy only from the root and a local audit policy only from the PDB to which it applies.

27.2.22.2 Enabling a Unified Audit Policy

The AUDIT POLICY statement can enable a unified audit policy.

• Use the following syntax to enable a unified audit policy:

```
AUDIT POLICY { policy_auditing } [WHENEVER [NOT] SUCCESSFUL]
```

In this specification:

- policy auditing refers to the following components:
 - The name of the unified audit policy. To find all existing policies, query the
 AUDIT_UNIFIED_POLICIES data dictionary view. To find currently enabled policies,
 query AUDIT UNIFIED ENABLED POLICIES.
 - Users or roles to whom the unified audit policy applies. To apply the policy to one
 or more users (including user SYS), enter the BY clause. For example:

```
BY psmith, rlee
```

To apply the policy to one or more users to whom the list of roles are directly granted, use the BY USERS WITH GRANTED ROLES clause. For example:

```
BY USERS WITH GRANTED ROLES HS ADMIN ROLE, HS ADMIN SELECT ROLE
```

 Users to exclude from the unified audit policy. To exclude one or more users from the policy, enter the EXCEPT clause. For example:

```
EXCEPT psmith, rlee
```

Mandatory audit records are captured in the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view for the <code>AUDIT_POLICY SQL</code> statement. To find users who have been excluded in the audit records, you can query the <code>EXCLUDED_USER</code> column in the <code>UNIFIED AUDIT TRAIL</code> view to list the excluded users.



You cannot enable the same audit policy with the BY, BY USERS WITH GRANTED ROLES, and EXCEPT clauses in the same statement. This action throws an error for the subsequent AUDIT statement with the conflicting clause

WHENEVER [NOT] SUCCESSFUL enables the policy to generate audit records based on
whether the user's actions failed or succeeded. See About Enabling Unified Audit Policies
for more information.

After you enable the unified audit policy and it is generating records, you can find the audit records by querying the UNIFIED AUDIT TRAIL data dictionary view.

27.2.22.3 Example: Enabling a Unified Audit Policy

The AUDIT POLICY statement can enable a unified audit policy using conditions, such as WHENEVER NOT SUCCESSFUL.

Example 27-44 shows how to enable a unified audit policy to record only failed actions by the user dv admin.

Example 27-44 Enabling a Unified Audit Policy

AUDIT POLICY dv_admin_pol BY tjones WHENEVER NOT SUCCESSFUL;

27.2.23 Disabling Unified Audit Policies

You can use the NOAUDIT POLICY statement to disable a unified audit policy.

- About Disabling Unified Audit Policies

 The NOAUDIT statement with the POLICY clause can disable a unified audit policy.
- Disabling a Unified Audit Policy
 The NOAUDIT statement can disable a unified audit policy using supported audit options.
- Example: Disabling a Unified Audit Policy
 The NOAUDIT POLICY statement disable a unified audit policy using filtering, such as by user name.

27.2.23.1 About Disabling Unified Audit Policies

The NOAUDIT statement with the POLICY clause can disable a unified audit policy.

In the NOAUDIT statement, you can specify a BY user or BY USERS WITH GRANTED ROLES role list, but not an EXCEPT user list. The disablement of a unified audit policy takes effect on subsequent user sessions.

You can find a list of existing unified audit policies by querying the AUDIT_UNIFIED_POLICIES data dictionary view.

In a multitenant environment, you can disable a common audit policy only from the root and a local audit policy only from the PDB to which it applies.

27.2.23.2 Disabling a Unified Audit Policy

The NOAUDIT statement can disable a unified audit policy using supported audit options.

Use the following syntax to disable a unified audit policy:

```
NOAUDIT POLICY {policy_auditing | existing_audit_options};
```



In this specification:

- policy_auditing is the name of the policy. To find all currently enabled policies, query the AUDIT_UNIFIED_ENABLED_POLICIES data dictionary view. As part of this specification, you optionally can include the BY or BY USERS WITH GRANTED ROLES clause, but not the EXCEPT clause. See About Enabling Unified Audit Policies for more information.
- existing_audit_options refers to AUDIT options that were available in releases earlier than Oracle Database 12c release 1 (12.1), such as the following:
 - * SELECT ANY TABLE, UPDATE ANY TABLE BY SCOTT, HR
 - * UPDATE ON SCOTT.EMP

If the unified policy had been applied to all users, then you only need to specify the policy name. For example:

NOAUDIT POLICY logons pol;

27.2.23.3 Example: Disabling a Unified Audit Policy

The NOAUDIT POLICY statement disable a unified audit policy using filtering, such as by user name.

Example 27-45 shows examples of how to disable a unified audit policy for a user and for a role.

Example 27-45 Disabling a Unified Audit Policy

```
NOAUDIT POLICY dv_admin_pol BY tjones;

NOAUDIT POLICY dv admin pol BY USERS WITH GRANTED ROLES emp admin;
```

27.2.24 Dropping Unified Audit Policies

You can use the DROP AUDIT POLICY statement to drop a unified audit policy.

- About Dropping Unified Audit Policies
 The DROP AUDIT POLICY statement can be used to unified audit policies.
- Dropping a Unified Audit Policy
 To drop a unified audit policy, you must first disable it, and then run the DROP AUDIT POLICY statement to remove it.
- Example: Disabling and Dropping a Unified Audit Policy
 The NOAUDIT POLICY and DROP AUDIT POLICY statements can disable and drop a unified audit policy.

27.2.24.1 About Dropping Unified Audit Policies

The DROP AUDIT POLICY statement can be used to unified audit policies.

If a unified audit policy is already enabled for a session, the effect of dropping the policy is not seen by this existing session. Until that time, the unified audit policy's settings remain in effect. For object-related unified audit policies, however, the effect is immediate.

You can find a list of existing unified audit policies by querying the AUDIT_UNIFIED_POLICIES data dictionary view.

When you disable an audit policy before dropping it, ensure that you disable it using the same settings that you used to enable it. For example, suppose you enabled the <code>logon_pol</code> policy as follows:

```
AUDIT POLICY logon pol BY HR, OE;
```

Before you can drop it, your NOAUDIT statement must include the HR and OE users as follows:

```
NOAUDIT POLICY logon pol BY HR, OE;
```

In a multitenant environment, you can drop a common audit policy only from the root and a local audit policy only from the PDB to which it applies.

27.2.24.2 Dropping a Unified Audit Policy

To drop a unified audit policy, you must first disable it, and then run the DROP AUDIT POLICY statement to remove it.

Use the following the following syntax to drop a unified audit policy:

```
DROP AUDIT POLICY policy name;
```

In a multitenant environment, the unified audit policy drop applies to the current PDB. If the unified audit policy was created as a common unified audit policy, then you cannot drop it from the local PDB.

Related Topics

Unified Audit Policies or AUDIT Settings in a Multitenant Environment
 In a multitenant environment, you can create unified audit policies for individual PDBs and in the root.

27.2.24.3 Example: Disabling and Dropping a Unified Audit Policy

The NOAUDIT POLICY and DROP AUDIT POLICY statements can disable and drop a unified audit policy.

Example 27-46 shows how to disable and drop a common unified audit policy.

Example 27-46 Disabling and Dropping a Unified Audit Policy

```
CONNECT c##sec_admin
Enter password: password
Connected.

NOAUDIT POLICY dv_admin_pol;

DROP AUDIT POLICY dv_admin_pol
```

27.2.25 Tutorial: Auditing Nondatabase Users

This tutorial shows how to create a unified audit policy that uses a client identifier to audit a nondatabase user's actions.

- Step 1: Create the User Accounts and Ensure the User OE Is Active You must first create users and ensure that the user OE is active.
- Step 2: Create the Unified Audit Policy
 Next, you are ready to create the unified audit policy.

Step 3: Test the Policy

To test the policy, use OE must try to select from the OE.ORDERS table.

Step 4: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

27.2.25.1 Step 1: Create the User Accounts and Ensure the User OE Is Active

You must first create users and ensure that the user OE is active.

1. Log on as user SYS with the SYSDBA administrative privilege.

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

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

For example:

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

To find the available PDBs, run the show pdbs command. To check the current PDB, run the show con name command.

3. Create the local user policy admin, who will create the fine-grained audit policy.

```
CREATE USER policy_admin IDENTIFIED BY password; GRANT CREATE SESSION, AUDIT_ADMIN TO policy_admin;
```

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure.

4. Create the local user account auditor, who will check the audit trail for this policy.

```
CREATE USER policy_auditor IDENTIFIED BY password;
GRANT CREATE SESSION, AUDIT VIEWER TO policy auditor;
```

5. The sample user OE will also be used in this tutorial, so query the DBA_USERS data dictionary view to ensure that OE is not locked or expired.

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

The account status should be OPEN. If the DBA_USERS view lists user OE as locked and expired, log in as user SYSTEM and then enter the following statement to unlock the OE account and create a new password:

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

Follow the guidelines in Minimum Requirements for Passwords to replace password with a password that is secure. For greater security, do **not** give the OE account the same password from previous releases of Oracle Database.

27.2.25.2 Step 2: Create the Unified Audit Policy

Next, you are ready to create the unified audit policy.

Connect to SQL*Plus as user policy admin.

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

Create the following policy:

```
CREATE AUDIT POLICY orders_unified_audpol
   ACTIONS INSERT ON OE.ORDERS, UPDATE ON OE.ORDERS, DELETE ON OE.ORDERS, SELECT ON
OE.ORDERS
   WHEN 'SYS_CONTEXT(''USERENV'', ''CLIENT_IDENTIFIER'') = ''robert'''
   EVALUATE PER STATEMENT;

AUDIT POLICY orders_unified_audpol;
```

In this example, the AUDIT_CONDITION parameter assumes that the nondatabase user is named robert. The policy will monitor any INSERT, UPDATE, DELETE, and SELECT statements that robert will attempt. Remember that the user's CLIENT_IDENTITIFER setting that you enter in the policy is case sensitive and that the policy only recognizes the case used for the identity that you specify here. In other words, later on, if the user session is set to Robert or ROBERT, the policy's condition will not be satisfied.

27.2.25.3 Step 3: Test the Policy

To test the policy, use OE must try to select from the OE.ORDERS table.

A unified auditing policy takes effect in the next user session for the users who are being audited. So, before their audit records can be captured, the users must connect to the database *after* the policy has been created.

1. Connect as user OE and select from the OE.ORDERS table.

```
CONNECT OE -- Or, CONNECT OE@hrpdb
Enter password: password
SELECT COUNT(*) FROM ORDERS;
```

The following output appears:

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

Connect as user policy auditor and then check if any audit records were generated.

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

col dbusername format a10
col client_identifier format a20
col sql_text format a29

SELECT DBUSERNAME, CLIENT_IDENTIFIER, SQL_TEXT FROM UNIFIED_AUDIT_TRAIL
WHERE SQL TEXT LIKE '%FROM ORDERS%';
```

The following output appears:

```
no rows selected
```

3. Reconnect as user OE, set the client identifier to robert, and then reselect from the OE.ORDERS table.

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

EXEC DBMS_SESSION.SET_IDENTIFIER('robert');

SELECT COUNT(*) FROM ORDERS;
```



The following output should appear:

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

4. Reconnect as user auditor and then check the audit trail again.

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

SELECT DBUSERNAME, CLIENT_IDENTIFIER, SQL_TEXT FROM UNIFIED_AUDIT_TRAIL
WHERE SQL TEXT LIKE '%FROM ORDERS%';
```

This time, because robert has made his appearance and queried the OE.ORDERS table, the audit trail captures his actions:

```
DBUSERNAME CLIENT_IDENTIFIER SQL_TEXT
------
OE robert SELECT COUNT(*) FROM ORDERS;
```

27.2.25.4 Step 4: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

 Connect to SQL*Plus as user policy_admin, and then manually disable and drop the orders unified audpol policy.

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

NOAUDIT POLICY orders_unified_audpol;
DROP AUDIT policy orders unified audpol;
```

(Unified audit policies reside in the ${\tt SYS}$ schema, not the schema of the user who created them.)

Connect to SQL*Plus as user SYSTEM.

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

Drop users policy admin and policy auditor.

```
DROP USER policy_admin;
DROP USER policy_auditor;
```

If you want, lock and expire OE, unless other users want to use this account:

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

27.3 Auditing Activities with the Predefined Unified Audit Policies

Oracle Database provides predefined unified audit policies that cover commonly used security-relevant audit settings.

Logon Failures Predefined Unified Audit Policy

The ORA_LOGON_FAILURES unified audit policy tracks failed logons only, but not any other kinds of logons.

Secure Options Predefined Unified Audit Policy

The $\mbox{ORA_SECURECONFIG}$ unified audit policy provides all the secure configuration audit options.

- Oracle Database Parameter Changes Predefined Unified Audit Policy
 The ORA_DATABASE_PARAMETER policy audits commonly used Oracle Database parameter settings.
- User Account and Privilege Management Predefined Unified Audit Policy
 The ORA ACCOUNT MGMT policy audits commonly used user account and privilege settings.
- Center for Internet Security Recommendations Predefined Unified Audit Policy
 The ORA_CIS_RECOMMENDATIONS policy performs audits that the Center for Internet Security
 (CIS) recommends.
- Oracle Database Real Application Security Predefined Audit Policies
 You can use predefined unified audit policies for Oracle Database Real Application
 Security events.
- Oracle Database Vault Predefined Unified Audit Policy for DVSYS and LBACSYS Schemas

The ORA_DV_AUDPOL predefined unified audit policy audits Oracle Database Vault DVSYS and LBACSYS schema objects.

 Oracle Database Vault Predefined Unified Audit Policy for Default Realms and Command Rules

The $ORA_DV_AUDPOL2$ predefined unified audit policy audits the Oracle Database Vault default realms and command rules.

Related Topics

Auditing Commonly Used Security-Relevant Activities
 Oracle Database provides a set default unified audit policies that you can choose from for commonly used security-relevant audits.

27.3.1 Logon Failures Predefined Unified Audit Policy

The <code>ORA_LOGON_FAILURES</code> unified audit policy tracks failed logons only, but not any other kinds of logons.

For new databases, this policy is enabled by default for both pure unified auditing and mixed-mode auditing environments. This policy is not enabled for databases that were upgraded from earlier versions, except if you have created a new database from the previous release and then upgrade it to the current release.



Only user SYS can alter or drop this predefined policy.

The following CREATE AUDIT POLICY statement shows the ORA_LOGON_FAILURES unified audit policy definition:

CREATE AUDIT POLICY ORA LOGON FAILURES ACTIONS LOGON;

You should enable the ORA LOGON FAILURES unified audit policy as follows:

AUDIT POLICY ORA LOGON FAILURES WHENEVER NOT SUCCESSFUL;

27.3.2 Secure Options Predefined Unified Audit Policy

The ORA SECURECONFIG unified audit policy provides all the secure configuration audit options.

For new databases, this policy is enabled by default for both pure unified auditing and mixed-mode auditing environments. This policy is not enabled for databases that were upgraded from earlier versions, except if you have created a new database from the previous release and then upgrade it to the current release.



Only user SYS can alter or drop this predefined policy.

The following CREATE AUDIT POLICY statement shows the ORA_SECURECONFIG unified audit policy definition.

```
CREATE AUDIT POLICY ORA SECURECONFIG
PRIVILEGES ALTER ANY TABLE, CREATE ANY TABLE, DROP ANY TABLE,
           CREATE ANY PROCEDURE, DROP ANY PROCEDURE, ALTER ANY PROCEDURE,
           GRANT ANY PRIVILEGE, GRANT ANY OBJECT PRIVILEGE, GRANT ANY ROLE,
           AUDIT SYSTEM, CREATE EXTERNAL JOB, CREATE ANY JOB,
           CREATE ANY LIBRARY,
           EXEMPT ACCESS POLICY,
           CREATE USER, DROP USER,
           ALTER DATABASE, ALTER SYSTEM,
           CREATE PUBLIC SYNONYM, DROP PUBLIC SYNONYM,
           CREATE SQL TRANSLATION PROFILE, CREATE ANY SQL TRANSLATION
PROFILE,
           DROP ANY SQL TRANSLATION PROFILE, ALTER ANY SQL TRANSLATION
PROFILE,
           TRANSLATE ANY SQL,
           EXEMPT REDACTION POLICY,
           PURGE DBA RECYCLEBIN, LOGMINING,
           ADMINISTER KEY MANAGEMENT, BECOME USER
ACTIONS ALTER USER, CREATE ROLE, ALTER ROLE, DROP ROLE,
           SET ROLE, CREATE PROFILE, ALTER PROFILE,
           DROP PROFILE, CREATE DATABASE LINK,
           ALTER DATABASE LINK, DROP DATABASE LINK,
           CREATE DIRECTORY, DROP DIRECTORY,
           CREATE PLUGGABLE DATABASE,
           DROP PLUGGABLE DATABASE,
           ALTER PLUGGABLE DATABASE,
           EXECUTE ON DBMS RLS,
           ALTER DATABASE DICTIONARY;
```

27.3.3 Oracle Database Parameter Changes Predefined Unified Audit Policy

The ORA_DATABASE_PARAMETER policy audits commonly used Oracle Database parameter settings.



Only user SYS can alter or drop this predefined policy.



The following CREATE AUDIT POLICY statement shows the ORA_DATABASE_PARAMETER unified audit policy definition. By default, this policy is not enabled.

CREATE AUDIT POLICY ORA_DATABASE_PARAMETER
ACTIONS ALTER DATABASE, ALTER SYSTEM, CREATE SPFILE;

27.3.4 User Account and Privilege Management Predefined Unified Audit Policy

The ORA ACCOUNT MGMT policy audits commonly used user account and privilege settings.



Only user SYS can alter or drop this predefined policy.

The following CREATE AUDIT POLICY statement shows the ORA_ACCOUNT_MGMT unified audit policy definition. By default, this policy is not enabled.

CREATE AUDIT POLICY ORA_ACCOUNT_MGMT

ACTIONS CREATE USER, ALTER USER, DROP USER, CREATE ROLE, DROP ROLE,

ALTER ROLE, SET ROLE, GRANT, REVOKE;

27.3.5 Center for Internet Security Recommendations Predefined Unified Audit Policy

The $ORA_CIS_RECOMMENDATIONS$ policy performs audits that the Center for Internet Security (CIS) recommends.



Only user SYS can alter or drop this predefined policy.

The following CREATE AUDIT POLICY statement shows the ORA_CIS_RECOMMENDATIONS unified audit policy definition. By default, this policy is not enabled.

CREATE AUDIT POLICY ORA CIS RECOMMENDATIONS PRIVILEGES SELECT ANY DICTIONARY, ALTER SYSTEM ACTIONS CREATE USER, ALTER USER, DROP USER, CREATE ROLE, DROP ROLE, ALTER ROLE, GRANT, REVOKE, CREATE DATABASE LINK, ALTER DATABASE LINK, DROP DATABASE LINK, CREATE PROFILE, ALTER PROFILE, DROP PROFILE, CREATE SYNONYM, DROP SYNONYM, CREATE PROCEDURE, DROP PROCEDURE, ALTER PROCEDURE, ALTER SYNONYM, CREATE FUNCTION, CREATE PACKAGE, CREATE PACKAGE BODY, ALTER FUNCTION, ALTER PACKAGE, ALTER SYSTEM, ALTER PACKAGE BODY, DROP FUNCTION, DROP PACKAGE, DROP PACKAGE BODY, CREATE TRIGGER, ALTER TRIGGER, DROP TRIGGER;

27.3.6 Oracle Database Real Application Security Predefined Audit Policies

You can use predefined unified audit policies for Oracle Database Real Application Security events.

- System Administrator Operations Predefined Unified Audit Policy
 The ORA_RAS_POLICY_MGMT predefined unified audit policy audits policies for all Oracle Real
 Application Security administrative actions on application users, roles, and policies.
- Session Operations Predefined Unified Audit Policy
 The ORA_RAS_SESSION_MGMT predefined unified audit policy audits policies for all run-time
 Oracle Real Application Security session actions and namespace actions.

Related Topics

Auditing Oracle Database Real Application Security Events
 You can use CREATE AUDIT POLICY statement to audit Oracle Database Real Application
 Security events.

27.3.6.1 System Administrator Operations Predefined Unified Audit Policy

The ORA_RAS_POLICY_MGMT predefined unified audit policy audits policies for all Oracle Real Application Security administrative actions on application users, roles, and policies.



Only user SYS can alter or drop this predefined policy.

The following CREATE AUDIT POLICY statement describes the ORA_RAS_POLICY_MGMT audit policy. By default, this policy is not enabled.

```
CREATE AUDIT POLICY ORA_RAS_POLICY_MGMT

ACTIONS COMPONENT=XS

CREATE USER, UPDATE USER, DELETE USER,
CREATE ROLE, UPDATE ROLE, DELETE ROLE, GRANT ROLE, REVOKE ROLE,
ADD PROXY, REMOVE PROXY,
SET USER PASSWORD, SET USER VERIFIER, SET USER PROFILE,
CREATE ROLESET, UPDATE ROLESET, DELETE ROLESET,
CREATE SECURITY CLASS, UPDATE SECURITY CLASS, DELETE SECURITY CLASS,
CREATE NAMESPACE TEMPLATE, UPDATE NAMESPACE TEMPLATE, DELETE NAMESPACE TEMPLATE,
CREATE ACL, UPDATE ACL, DELETE ACL,
CREATE DATA SECURITY, UPDATE DATA SECURITY,
ENABLE DATA SECURITY, DISABLE DATA SECURITY,
ADD GLOBAL CALLBACK, DELETE GLOBAL CALLBACK, ENABLE GLOBAL CALLBACK;
```

27.3.6.2 Session Operations Predefined Unified Audit Policy

The ORA_RAS_SESSION_MGMT predefined unified audit policy audits policies for all run-time Oracle Real Application Security session actions and namespace actions.

Note:

Only user SYS can alter or drop this predefined policy.

The following CREATE AUDIT POLICY statement describes the ORA_RAS_SESSION_MGMT policy. By default, this policy is not enabled.

```
CREATE AUDIT POLICY ORA_RAS_SESSION_MGMT

ACTIONS COMPONENT=XS

CREATE SESSION, DESTROY SESSION,

ENABLE ROLE, DISABLE ROLE,
SET COOKIE, SET INACTIVE TIMEOUT,
SWITCH USER, ASSIGN USER,
CREATE SESSION NAMESPACE, DELETE SESSION NAMESPACE,
CREATE NAMESPACE ATTRIBUTE, GET NAMESPACE ATTRIBUTE, SET NAMESPACE ATTRIBUTE,
DELETE NAMESPACE ATTRIBUTE;
```

27.3.7 Oracle Database Vault Predefined Unified Audit Policy for DVSYS and LBACSYS Schemas

The ORA_DV_AUDPOL predefined unified audit policy audits Oracle Database Vault DVSYS and LBACSYS schema objects.

The <code>ORA_DV_AUDPOL</code> policy audits all actions that are performed on the Oracle Database Vault <code>DVSYS</code> (including <code>DVF</code>) schema objects and the Oracle Label Security <code>LBACSYS</code> schema objects. It does not capture actions on the F\$* factor functions in the <code>DVF</code> schema. By default, this policy is not enabled.



Only user SYS can alter or drop this predefined policy.

To view the complete definition of this policy, query the AUDIT_UNIFIED_POLICIES data dictionary view, where policy name is ORA DV AUDPOL.

Related Topics

Auditing Oracle Database Vault Events
 In an Oracle Database Vault environment, the CREATE AUDIT POLICY statement can audit Database Vault activities.

27.3.8 Oracle Database Vault Predefined Unified Audit Policy for Default Realms and Command Rules

The $\mbox{ORA_DV_AUDPOL2}$ predefined unified audit policy audits the Oracle Database Vault default realms and command rules.

The ORA_DV_AUDPOL2 policy constitutes the audit settings of the Oracle Database Vault-supplied default realms and command rules. By default, this policy is not enabled.



Only user SYS can alter or drop this predefined policy.

To view the complete definition of this policy, query the AUDIT_UNIFIED_POLICIES data dictionary view, where policy name is ORA DV AUDPOL2.

Related Topics

Auditing Oracle Database Vault Events
 In an Oracle Database Vault environment, the CREATE AUDIT POLICY statement can audit Database Vault activities.

27.4 Auditing Specific Activities with Fine-Grained Auditing

Fine-grained auditing enables you to create audit policies at the granular level.

- About Fine-Grained Auditing
 - Fine-grained auditing enables you to create policies that define specific conditions that must take place for the audit to occur.
- Where Are Fine-Grained Audit Records Stored?
 Fine-grained auditing records are stored in the AUDSYS schema.
- Who Can Perform Fine-Grained Auditing?
 - Oracle provides roles for privileges needed to create fine-grained audit policies and to view and analyze fine-grained audit policy data.
- Fine-Grained Auditing on Tables or Views That Have Oracle VPD Policies
 The audit trail captures the VPD predicate for fine-grained audited tables or views that are included in an Oracle VPD policy.
- Fine-Grained Auditing in a Multitenant Environment
 You can create fine-grained audit policies in the CDB root, application root, CDB PDBs,
 and application PDBs.
- Fine-Grained Audit Policies with Editions
 - You can prepare an application for edition-based redefinition, and cover each table that the application uses with an editioning view.
- Using the DBMS_FGA PL/SQL Package to Manage Fine-Grained Audit Policies
 The DBMS_FGA PL/SQL package manages fine-grained audit policies.
- Tutorial: Adding an Email Alert to a Fine-Grained Audit Policy
 This tutorial demonstrates how to create a fine-grained audit policy that generates an email alert when users violate the policy.

Related Topics

Auditing Specific, Fine-Grained Activities
 Use fine-grained auditing if you want to audit individual columns and use event handlers.

27.4.1 About Fine-Grained Auditing

Fine-grained auditing enables you to create policies that define specific conditions that must take place for the audit to occur.

You cannot create unified audit policies using fine-grained auditing but you can use fine-grained auditing to create very customized audit settings, such as auditing the times that data is accessed.

This enables you to monitor data access based on content. It provides granular auditing of queries, and INSERT, UPDATE, and DELETE operations. You can use fine-grained auditing to audit the following types of actions:

- Accessing a table between 9 p.m. and 6 a.m. or on Saturday and Sunday
- Using an IP address from outside the corporate network
- Selecting or updating a table column
- Modifying a value in a table column

In general, fine-grained audit policies are based on simple, user-defined SQL predicates on table objects as conditions for selective auditing. During fetching, whenever policy conditions are met for a row, the query is audited.

Unified audit policies can perform most of the operations that fine-grained audit policies can perform, except for the following actions:

- Auditing specific columns. You can audit specific relevant columns that hold sensitive information, such as salaries or Social Security numbers.
- Using event handlers. For example, you can write a function that sends an email alert to a security administrator when an audited column that should not be changed at midnight is updated.

Fine-grained auditing has the following advantages over unified auditing:

- It enables you to perform row value-based auditing. For example, you can audit updates to a salary column when the updated value is higher than a specified threshold, but not otherwise.
- You can use fine-grained auditing event handlers to proactively notify administrators or other users of specific events.
- During a bulk data processing operation using BULK COLLECT and FORALL in PL/SQL to
 execute the same DML statement repeatedly for different bind variables, fine-grained
 auditing can capture the repeatedly executed statement with the appropriate bind variable
 values.

Note:

- Fine-grained auditing is supported only with cost-based optimization. For queries
 using rule-based optimization, fine-grained auditing checks before applying row
 filtering, which could result in an unnecessary audit event trigger.
- Policies currently in force on an object involved in a flashback query are applied to the data returned from the specified flashback snapshot based on time or system change number (SCN).
- If you want to use fine-grained auditing to audit data that is being directly loaded (for example, using Oracle Warehouse Builder to execute DML statements), then Oracle Database transparently makes all direct loads that are performed in the database instance into conventional loads. If you want to preserve the direct loading of data, consider using unified audit policies instead.



27.4.2 Where Are Fine-Grained Audit Records Stored?

Fine-grained auditing records are stored in the AUDSYS schema.

These audit records are stored in the SYSAUX tablespace by default. You can supply a new tablespace by using the DBMS_AUDIT_MGMT.SET_AUDIT_TRAIL_LOCATION procedure. This tablespace can be an encrypted tablespace. To find the records have been generated for the audit policies that are in effect, you can query UNIFIED AUDIT TRAIL data dictionary view.

The audit trail captures an audit record for each reference of a table or a view within a SQL statement. For example, if you run a UNION statement that references the HR.EMPLOYEES table twice, then an audit policy for statement generates two audit records, one for each access of the HR.EMPLOYEES table.

See Also:

- Activities That Are Mandatorily Audited
- Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS AUDIT MGMT.SET AUDIT TRAIL LOCATION procedure
- Oracle Database Reference for more information about the UNIFIED AUDIT TRAIL data dictionary view

27.4.3 Who Can Perform Fine-Grained Auditing?

Oracle provides roles for privileges needed to create fine-grained audit policies and to view and analyze fine-grained audit policy data.

The fine-grained audit privileges are as follows:

- To create fine-grained audit policies, you must be granted d the AUDIT_ADMIN role or the EXECUTE privilege on the DBMS FGA package.
- To view and analyze fine-grained audit data, you must be granted the AUDIT VIEWER role.

The PL/SQL package is already granted to AUDIT_ADMIN role. As with all privileges, grant these roles to trusted users only. You can find the roles that user have been granted by querying the DBA ROLE PRIVS data dictionary view.

27.4.4 Fine-Grained Auditing on Tables or Views That Have Oracle VPD Policies

The audit trail captures the VPD predicate for fine-grained audited tables or views that are included in an Oracle VPD policy.

This behavior is similar to how the unified audit trail captures the VPD predicate for unified audit policies.

The audit trail also captures internal predicates from Oracle Label Security and Oracle Real Application Security policies.

You do not need to create a special audit policy to capture the VPD predicate audit records. The predicate information is automatically stored in the RLS_INFO column of the DBA FGA AUDIT TRAIL and UNIFIED AUDIT TRAIL data dictionary views.

If there are multiple VPD policies applied to the same table or view, then by default the predicates for these policies are concatenated in the RLS_INFO column. You can reformat the output so that each predicate is in its own row (identified by its corresponding VPD policy name and other information) by using the functions in the DBMS AUDIT UTIL PL/SQL package.

See Also:

- Auditing of Oracle Virtual Private Database Predicates for more information about the auditing of VPD predicates and for an example of how to use the DBMS AUDIT UTIL package functions to format captured audit data
- Oracle Database PL/SQL Packages and Types Reference for more information about the DBMS AUDIT UTIL PL/SQL package

27.4.5 Fine-Grained Auditing in a Multitenant Environment

You can create fine-grained audit policies in the CDB root, application root, CDB PDBs, and application PDBs.

Note the following general rules about fine-grained audit policies in a multitenant environment:

- You cannot create fine-grained audit policies on SYS objects.
- You cannot create fine-grained audit policies, either local or application common, for extended data link objects.
- When you create a fine-grained audit policy in the CDB root, the policy cannot be applied
 to all PDBs. It only applies to objects within the CDB root. (In other words, there is no such
 thing as a common fine-grained audit policy for the CDB root.) If you want to create a finegrained audit policy to audit a common object's access in all the PDBs, then you must
 explicitly create that policy in each PDB and then enable it on the common objects that is
 accessible in the PDB.
- When you create a fine-grained audit policy in a PDB, it applies only to objects within the PDB
- You can create application common fine-grained audit policies only if you are connected to the application root and only within the BEGIN/END block. If you are connected to the application root and create the fine-grained audit policy outside the BEGIN/END block, then the fine-grained audit policy is created in the application root.
- You cannot create application common fine-grained audit policies on local PDB objects.
- If the application common fine-grained audit policy has a handler, then this handler must be owned by either an application common user or a CDB common user.
- You can create an application fine-grained audit policy on local (PDB) objects and CDB common objects. Because the policy is local to its container, the object on which the policy is defined is audited only in the particular container where the policy is defined. For example, if you create a fine-grained audit policy in the hr_pdb PDB, the object for which you create this policy must exist in the hr_pdb PDB.



- You cannot create local fine-grained audit policies in an application PDB on object linked and extended data link objects. On metadata-linked objects are allowed in the fine-grained audit policy.
- Application root local policies are allowed for all application common objects.
- When you create a fine-grained audit policy as a common audit policy in an application root, it will be effective in each PDB that belongs to this application root. Therefore, any access to the application common object and CDB common object (on which the application common fine-grained audit policy is defined) from the application PDB is audited in the fine-grained audit trail in that application PDB.
- When you create scripts for application install, upgrade, patch, or uninstall operations, you can include SQL statements within the ALTER PLUGGABLE DATABASE app_name BEGIN INSTALL and ALTER PLUGGABLE DATABASE app_name END INSTALL blocks to perform various operations. You can include fine-grained audit policy statements only within these blocks.
- You can only enable, disable, or drop application common fine-grained audit policies from the application root, and from within a ALTER PLUGGABLE DATABASE app_name BEGIN INSTALL and ALTER PLUGGABLE DATABASE app name END INSTALL block in a script.

27.4.6 Fine-Grained Audit Policies with Editions

You can prepare an application for edition-based redefinition, and cover each table that the application uses with an editioning view.

If you do this, then you must move the fine-grained audit polices that protect these tables to the editioning view. You can find information about the currently configured editions by querying the DBA_EDITIONS data dictionary view. To find information about fine-grained audit policies, query DBA_AUDIT_POLICIES.

27.4.7 Using the DBMS_FGA PL/SQL Package to Manage Fine-Grained Audit Policies

The DBMS FGA PL/SQL package manages fine-grained audit policies.

- About the DBMS_FGA PL/SQL PL/SQL Package
 The DBMS_FGA PL/SQL package can be used to combine statements into one policy and perform other fine-grained auditing management tasks.
- The DBMS_FGA PL/SQL Package with Editions
 You can create DBMS FGA policies for use in an editions environment.
- The DBMS_FGA PL/SQL Package in a Multitenant Environment
 In a multitenant environment, the DBMS_FGA PL/SQL package applies only to the current
 local PDBs.
- Creating a Fine-Grained Audit Policy
 The DBMS FGA.ADD POLICY procedure creates a fine-grained audit policy.
- Example: Using DBMS_FGA.ADD_POLICY to Create a Fine-Grained Audit Policy
 The DBMS_FGA.ADD_POLICY procedure can create a fine-grained audit policy using multiple
 statement types.
- Disabling a Fine-Grained Audit Policy
 The DBMS FGA.DISABLE POLICY procedure disables a fine-grained audit policy.



- Enabling a Fine-Grained Audit Policy
 The DBMS FGA. ENABLE POLICY procedure enables a fine-grained audit policy.
- Dropping a Fine-Grained Audit Policy
 The DBMS_FGA.DROP_POLICY procedure drops a fine-grained audit policy.

27.4.7.1 About the DBMS FGA PL/SQL PL/SQL Package

The DBMS_FGA PL/SQL package can be used to combine statements into one policy and perform other fine-grained auditing management tasks.

However, unless you want to perform column-level auditing or use event handlers with your audit policy, you should create audit policies as described in Auditing Activities with Unified Audit Policies and the AUDIT Statement.

The <code>DBMS_FGA PL/SQL</code> package enables you to add all combinations of <code>SELECT</code>, <code>INSERT</code>, <code>UPDATE</code>, and <code>DELETE</code> statements to one policy. You also can audit <code>MERGE</code> statements, by auditing the underlying actions of <code>INSERT</code> and <code>UPDATE</code>. To audit <code>MERGE</code> statements, configure fine-grained access on the <code>INSERT</code> and <code>UPDATE</code> statements. Only one record is generated for each policy for successful <code>MERGE</code> operations.

To administer fine-grained audit policies, you must have be granted the AUDIT_ADMIN role. Note also that the EXECUTE privilege for the DBMS FGA package is mandatorily audited.

The audit policy is bound to the table for which you created it. This simplifies the management of audit policies because the policy only needs to be changed once in the database, not in each application. In addition, no matter how a user connects to the database—from an application, a Web interface, or through SQL*Plus or Oracle SQL Developer—Oracle Database records any actions that affect the policy.

If any rows returned from a query match the audit condition that you define, then Oracle Database inserts an audit entry into the fine-grained audit trail. This entry excludes all the information that is reported in the regular audit trail. In other words, only one row of audit information is inserted into the audit trail for every fine-grained audit policy that evaluates to true.



Oracle Database PL/SQL Packages and Types Reference for detailed information about the $\tt DBMS_FGA$ package

27.4.7.2 The DBMS_FGA PL/SQL Package with Editions

You can create DBMS FGA policies for use in an editions environment.

If you plan to use the <code>DBMS_FGA</code> package policy across different editions, then you can control the results of the policy: whether the results are uniform across all editions, or specific to the edition in which the policy is used.

Related Topics

How Editions Affects the Results of a Global Application Context PL/SQL Package
Global application context packages, Oracle Virtual Private Database packages, and finegrained audit policies can be used across multiple editions.

27.4.7.3 The DBMS FGA PL/SQL Package in a Multitenant Environment

In a multitenant environment, the $DBMS_FGA$ PL/SQL package applies only to the current local PDBs.

You cannot create one policy for the entire multitenant environment. The policy must be specific to objects within a PDB. To find PDBs, you can query the DBA_PDBS data dictionary view. To find the name of the current PDB, issue the show con name command.

27.4.7.4 Creating a Fine-Grained Audit Policy

The DBMS FGA.ADD POLICY procedure creates a fine-grained audit policy.

- About Creating a Fine-Grained Audit Policy
 - The DBMS_FGA.ADD_POLICY procedure creates an audit policy using the supplied predicate as the audit condition.
- Syntax for Creating a Fine-Grained Audit Policy
 The DBMS_FGA.ADD_POLICY procedure includes many settings, such as the ability to use a handler for complex auditing.
- Audits of Specific Columns and Rows
 You can fine-tune audit behavior by targeting a specific column (relevant column) to be
 audited if a condition is met.

27.4.7.4.1 About Creating a Fine-Grained Audit Policy

The DBMS_FGA.ADD_POLICY procedure creates an audit policy using the supplied predicate as the audit condition.

By default, Oracle Database executes the policy predicate with the privileges of the user who owns the policy. The maximum number of fine-grained policies on any table or view object is 256. Oracle Database stores the policy in the data dictionary table, but you can create the policy on any table or view that is not in the SYS schema. In a multitenant environment, the fine grained policy is only created in the local PDB.

If you plan to create a materialized view on the the base table on which you want to create a fine-grained audit policy, then you must create the fine-grained audit policy on the base table before you create the materialized view on the same table. Otherwise, any refresh operations on the materialized view will fail with an ORA-12008: error in materialized view refresh path error.

You cannot modify a fine-grained audit policy after you have created it. If you must modify the policy, then drop and recreate it.

You can find information about a fine-grained audit policy by querying the ALL_AUDIT_POLICIES, DBA_AUDIT_POLICIES, and USER_AUDIT_POLICIES views. The UNIFIED_AUDIT_TRAIL view contains a column entitled FGA_POLICY_NAME, which you can use to filter out rows that were generated using a specific fine-grained audit policy.

27.4.7.4.2 Syntax for Creating a Fine-Grained Audit Policy

The DBMS_FGA.ADD_POLICY procedure includes many settings, such as the ability to use a handler for complex auditing.

The DBMS_FGA.ADD_POLICY procedure syntax is as follows:

In this specification:

- object_schema specifies the schema of the object to be audited. (If NULL, the current log-on user schema is assumed.)
- object name specifies the name of the object to be audited.
- policy_name specifies the name of the policy to be created. Ensure that this name is unique.
- audit_condition specifies a Boolean condition in a row. NULL is allowed and acts as TRUE.
 See Audits of Specific Columns and Rows for more information. If you specify NULL or no audit condition, then any action on a table with that policy creates an audit record, whether or not rows are returned.

Follow these guidelines:

- Do not include functions, which execute the auditable statement on the same base table, in the audit_condition setting. For example, suppose you create a function that executes an INSERT statement on the HR.EMPLOYEES table. The policy's audit_condition contains this function and it is for INSERT statements (as set by statement_types). When the policy is used, the function executes recursively until the system has run out of memory. This can raise the error ORA-1000: maximum open cursors exceeded Or ORA-00036: maximum number of recursive SQL levels (50) exceeded.
- Do not issue the DBMS_FGA.ENABLE_POLICY or DBMS_FGA.DISABLE_POLICY statement from a function in a policy's condition.
- audit_column specifies one or more columns to audit, including hidden columns. If set to
 NULL or omitted, all columns are audited. These can include Oracle Label Security hidden
 columns or object type columns. The default, NULL, causes audit if any column is accessed
 or affected.
- handler_schema: If an alert is used to trigger a response when the policy is violated, specifies the name of the schema that contains the event handler. The default, NULL, uses the current schema. See also Tutorial: Adding an Email Alert to a Fine-Grained Audit Policy.
- handler_module specifies the name of the event handler. Include the package the event handler is in. This function is invoked only after the first row that matches the audit condition in the query is processed.

Follow these guidelines:

Do not create recursive fine-grained audit handlers. For example, suppose you create
a handler that executes an INSERT statement on the HR.EMPLOYEES table. The policy
that is associated with this handler is for INSERT statements (as set by the

statement_types parameter). When the policy is used, the handler executes recursively until the system has run out of memory. This can raise the error ORA-1000: maximum open cursors exceeded Or ORA-00036: maximum number of recursive SQL levels (50) exceeded.

- Do not issue the DBMS_FGA.ENABLE_POLICY or DBMS_FGA.DISABLE_POLICY statement from a policy handler. Doing so can raise the ORA-28144: Failed to execute finegrained audit handler error.
- enable enables or disables the policy using true or false. If omitted, the policy is enabled.
 The default is TRUE.
- statement_types: Specifies the SQL statements to be audited: INSERT, UPDATE, DELETE, or SELECT only. If you want to audit a MERGE operation, then set statement_types to 'INSERT, UPDATE'. The default is SELECT.
- audit_trail: If you have migrated to unified auditing, then Oracle Database ignores this
 parameter and writes the audit records immediately to the unified audit trail. If you have
 migrated to unified auditing, then omit this parameter.

Be aware that sensitive data, such as credit card information, can be recorded in clear text.

- audit_column_opts: If you specify more than one column in the audit_column parameter, then this parameter determines whether to audit all or specific columns. See Audits of Specific Columns and Rows for more information.
- policy_owner is the user who owns the fine-grained auditing policy. However, this setting
 is not a user-supplied argument. The Oracle Data Pump client uses this setting internally to
 recreate the fine-grained audit policies appropriately.

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about the $\tt DBMS$ FGA.ADD POLICY syntax

27.4.7.4.3 Audits of Specific Columns and Rows

You can fine-tune audit behavior by targeting a specific column (*relevant column*) to be audited if a condition is met.

To accomplish this, you use the <code>audit_column</code> parameter to specify one or more sensitive columns. In addition, you can audit data in specific rows by using the <code>audit_condition</code> parameter to define a Boolean condition. (However, if your policy needs only to audit for conditions, consider using an audit policy condition described in Creating a Condition for a Unified Audit Policy.)

The following settings from Example 27-47 enable you to perform an audit if anyone in Department 50 (DEPARTMENT_ID = 50) tries to access the SALARY and COMMISSION_PCT columns.

```
audit_condition => 'DEPARTMENT_ID = 50',
audit_column => 'SALARY,COMMISSION PCT,'
```

As you can see, this feature is enormously beneficial. It not only enables you to pinpoint particularly important types of data to audit, but it provides increased protection for columns that contain sensitive data, such as Social Security numbers, salaries, patient diagnoses, and so on.

If the <code>audit_column</code> lists more than one column, then you can use the <code>audit_column_opts</code> parameter to specify whether a statement is audited when the query references <code>any</code> column specified in the <code>audit_column</code> parameter or only when <code>all</code> columns are referenced. For example:

```
audit_column_opts => DBMS_FGA.ANY_COLUMNS,
audit column opts => DBMS FGA.ALL COLUMNS,
```

If you do not specify a relevant column, then auditing applies to all columns.

See Also:

Oracle Database PL/SQL Packages and Types Reference for more information about the audit_condition, audit_column, and audit_column_opts parameters in the DBMS_FGA.ADD_POLICY procedure (see also the usage notes for the ADD_POLICY procedure in that section)

27.4.7.5 Example: Using DBMS_FGA.ADD_POLICY to Create a Fine-Grained Audit Policy

The DBMS_FGA.ADD_POLICY procedure can create a fine-grained audit policy using multiple statement types.

Example 27-47 shows how to audit statements INSERT, UPDATE, DELETE, and SELECT on table HR.EMPLOYEES.

Note that this example omits the $audit_column_opts$ parameter, because it is not a mandatory parameter.

Example 27-47 Using DBMS_FGA.ADD_POLICY to Create a Fine-Grained Audit Policy

After you create the policy, if you query the DBA_AUDIT_POLICIES view, you will find the new policy listed:

Afterwards, any of the following SQL statements log an audit event record.

```
SELECT COUNT(*) FROM HR.EMPLOYEES WHERE COMMISSION_PCT = 20 AND SALARY > 4500;
SELECT SALARY FROM HR.EMPLOYEES WHERE DEPARTMENT_ID = 50;
```

DELETE FROM HR.EMPLOYEES WHERE SALARY > 1000000;

27.4.7.6 Disabling a Fine-Grained Audit Policy

The DBMS_FGA.DISABLE_POLICY procedure disables a fine-grained audit policy.

Use the following syntax to disable a fine-grained audit policy:

For example, to disable the fine-grained audit policy that was created in Example 27-47.

✓ See Also:

Oracle Database PL/SQL Packages and Types Reference for detailed information about the DISABLE POLICY syntax

27.4.7.7 Enabling a Fine-Grained Audit Policy

The DBMS FGA. ENABLE POLICY procedure enables a fine-grained audit policy.

Use the following syntax to enable a fine-grained audit policy:

For example, to reenable the <code>chk_hr_emp</code> policy by using the <code>DBMS_FGA.ENABLE_POLICY</code> procedure



See Also:

Oracle Database PL/SQL Packages and Types Reference for detailed information about the ${\tt ENABLE_POLICY}$ syntax

27.4.7.8 Dropping a Fine-Grained Audit Policy

The DBMS FGA. DROP POLICY procedure drops a fine-grained audit policy.

Oracle Database automatically drops the audit policy if you remove the object specified in the object_name parameter of the DBMS_FGA.ADD_POLICY procedure, or if you drop the user who created the audit policy.

Use the following syntax to drop a fine-grained audit policy:

For example, to drop a fine-grained audit policy manually by using the <code>DBMS_FGA.DROP_POLICY</code> procedure:

See Also:

Oracle Database PL/SQL Packages and Types Reference for detailed information about the DROP_POLICY syntax

27.4.8 Tutorial: Adding an Email Alert to a Fine-Grained Audit Policy

This tutorial demonstrates how to create a fine-grained audit policy that generates an email alert when users violate the policy.

About This Tutorial

This tutorial shows how you can add an email alert to a fine-grained audit policy that goes into effect when a user (or an intruder) violates the policy.

- Step 1: Install and Configure the UTL_MAIL PL/SQL Package
 The UTL_MAIL PL/SQL manages email that includes commonly used email features, such as attachments, CC, and BCC.
- Step 2: Create User Accounts

You must create an administrative account and an auditor user.

Step 3: Configure an Access Control List File for Network Services
 An access control list (ACL) file can be used to enable fine-grained access to external network services.

- Step 4: Create the Email Security Alert PL/SQL Procedure
 The email security alert PL/SQL procedure generates a message describing the violation and then sends this message to the appropriate users.
- Step 5: Create and Test the Fine-Grained Audit Policy Settings
 The fine-grained audit policy will trigger the alert when the policy is violated.
- Step 6: Test the Alert
 With the components in place, you are ready to test the alert.
- Step 7: Remove the Components of This Tutorial
 If you no longer need the components of this tutorial, then you can remove them.

27.4.8.1 About This Tutorial

This tutorial shows how you can add an email alert to a fine-grained audit policy that goes into effect when a user (or an intruder) violates the policy.

Note:

- To complete this tutorial, you must use a database that has an SMTP server.
- If you are using a multitenant environment, then this tutorial applies to the current PDB only.

To add an email alert to a fine-grained audit policy, you first must create a procedure that generates the alert, and then use the following <code>DBMS_FGA.ADD_POLICY</code> parameters to call this function when someone violates this policy:

- handler schema: The schema in which the handler event is stored
- handler module: The name of the event handler

The alert can come in any form that best suits your environment: an email or pager notification, updates to a particular file or table, and so on. Creating alerts also helps to meet certain compliance regulations, such as California Senate Bill 1386. In this tutorial, you will create an email alert.

In this tutorial, you create an email alert that notifies a security administrator that a Human Resources representative is trying to select or modify salary information in the HR.EMPLOYEES table. The representative is permitted to make changes to this table, but to meet compliance regulations, we want to create a record of all salary selections and modifications to the table.

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

The $\mathtt{UTL_MAIL}$ PL/SQL manages email that includes commonly used email features, such as attachments, CC, and BCC.

You must install and configure this package before you can use it. It is not installed and configured by default.

1. Log on as user SYS with the SYSDBA administrative privilege.

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

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

For example:

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

To find the available PDBs, run the show pdbs command. To check the current PDB, run the show con name command.

3. Install the UTL MAIL package.

```
@$ORACLE_HOME/rdbms/admin/utlmail.sql
@$ORACLE_HOME/rdbms/admin/prvtmail.plb
```

The UTL MAIL package enables you to manage email.

Be aware that currently, the UTL MAIL PL/SQL package does not support SSL servers.

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

For example:

```
SHOW PARAMETER SMTP OUT SERVER
```

If the ${\tt SMTP_OUT_SERVER}$ parameter has already been set, then output similar to the following appears:

NAME	TYPE	VALUE
SMTP_OUT_SERVER	string	some_imap_server.example.com

5. Issue the following ALTER SYSTEM statement:

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

Replace <code>imap_mail_server.example.com</code> with the name of your SMTP server, which you can find in the account settings in your email tool. Enclose these settings in quotation marks. For example:

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

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

```
CONNECT SYS AS SYSOPER -- Or, CONNECT SYS@hrpdb AS SYSOPER
Enter password: password
SHUTDOWN IMMEDIATE
STARTUP
```

7. Ensure that the SMTP OUT SERVER parameter setting is correct.

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

SHOW PARAMETER SMTP OUT SERVER
```

Output similar to the following appears:

NAME	TYPE	VALUE
SMTP_OUT_SERVER	string	<pre>my_imap_server.example.com</pre>





Oracle Database PL/SQL Packages and Types Reference for more information about the $\verb"utl mail"$ package

27.4.8.3 Step 2: Create User Accounts

You must create an administrative account and an auditor user.

1. Ensure that you are connected as SYS using the SYSDBA administrative privilege, and then create the fga admin user, who will create the fine-grained audit policy.

For example:

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

CREATE USER fga_admin IDENTIFIED BY password;
GRANT CREATE SESSION, CREATE PROCEDURE, AUDIT_ADMIN TO fga_admin;
GRANT EXECUTE ON UTL_TCP TO fga_admin;
GRANT EXECUTE ON UTL_MAIL TO fga_admin;
GRANT EXECUTE ON UTL_MAIL TO fga_admin;
GRANT EXECUTE ON DBMS NETWORK ACL ADMIN TO fga admin;
```

Follow the guidelines in Minimum Requirements for Passwordsto replace *password* with a password that is secure.

The UTL_TCP, UTL_SMTP, UTL_MAIL, and DBMS_NETWORK_ACL_ADMIN PL/SQL packages are used by the email security alert that you create.

2. Create the auditor user, who will check the audit trail for this policy.

```
GRANT CREATE SESSION TO fga_auditor IDENTIFIED BY password; GRANT AUDIT VIEWER TO fga auditor;
```

3. Connect as user SYSTEM.

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

4. Ensure that the HR schema account is unlocked and has a password. If necessary, unlock HR and grant this user a password.

```
SELECT USERNAME, ACCOUNT STATUS FROM DBA USERS WHERE USERNAME = 'HR';
```

The account status should be <code>OPEN.</code> If the <code>DBA_USERS</code> view lists user HR as locked and expired, then enter the following statement to unlock the HR account and create a new password:

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

Follow the guidelines in Minimum Requirements for Passwords to create a password that is secure. For greater security, do **not** give the HR account the same password from previous releases of Oracle Database.

5. Create a user account for Susan Mavris, who is an HR representative whose actions you will audit, and then grant this user access to the HR.EMPLOYEES table.

```
GRANT CREATE SESSION TO smavris IDENTIFIED BY password;
GRANT SELECT, INSERT, UPDATE, DELETE ON HR.EMPLOYEES TO SMAVRIS;
```

27.4.8.4 Step 3: Configure an Access Control List File for Network Services

An access control list (ACL) file can be used to enable fine-grained access to external network services.

Before you can use PL/SQL network utility packages such as UTL_MAIL, you must configure this type of access control list (ACL) file.

1. Connect to SQL*Plus as user fga admin.

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

2. Configure the following access control setting and its privilege definitions.

In this example:

- SMTP_OUT_SERVER_setting: Enter the SMTP_OUT_SERVER setting that you set for the SMTP_OUT_SERVER parameter in Step 1: Install and Configure the UTL_MAIL PL/SQL Package. This setting should match exactly the setting that your email tool specifies for its outgoing server.
- lower_port: Enter the port number that your email tool specifies for its outgoing server. Typically, this setting is 25. Enter this value for the lower_port setting. (Currently, the UTL_MAIL package does not support SSL. If your email server is an SSL server, then enter 25 for the port number, even if the email server uses a different port number.)
- ace: Define the privileges here.

See Also:

Managing Fine-Grained Access in PL/SQL Packages and Types for detailed information about configuring an access control list (ACL) file

27.4.8.5 Step 4: Create the Email Security Alert PL/SQL Procedure

The email security alert PL/SQL procedure generates a message describing the violation and then sends this message to the appropriate users.

As user fga admin, create the following procedure.

```
CREATE OR REPLACE PROCEDURE email_alert (sch varchar2, tab varchar2, pol
varchar2)
AS
msg varchar2(20000) := 'HR.EMPLOYEES table violation. The time is: ';
```

In this example:

- CREATE OR REPLACE PROCEDURE ...AS: You must include a signature that describes
 the schema name (sch), table name (tab), and the name of the audit procedure (pol)
 that you will define in audit policy in the next step.
- sender and recipients: Replace youremail@example.com with your email address, and recipientemail@example.com with the email address of the person you want to receive the notification.

27.4.8.6 Step 5: Create and Test the Fine-Grained Audit Policy Settings

The fine-grained audit policy will trigger the alert when the policy is violated.

1. As user fga admin, create the chk hr emp policy fine-grained audit policy as follows.

2. Commit the changes you have made to the database.

COMMIT;

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

```
EXEC email_alert ('hr', 'employees', 'chk_hr_emp');
```

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

If you receive an ORA-24247: network access denied by access control list (ACL) error followed by ORA-06512: at string line string errors, then check the settings in the access control list file.

27.4.8.7 Step 6: Test the Alert

With the components in place, you are ready to test the alert.

Connect to SQL*Plus as user smavris, check your salary, and give yourself a nice raise.

By now, depending on the speed of your email server, you (or your recipient) should have received an email with the subject header Table modification on HR.EMPLOYEES notifying you of the tampering of the HR.EMPLOYEES table. Now all you need to do is to query the UNIFIED AUDIT TRAIL data dictionary view to find who the violator is.

2. As user fga auditor, query the UNIFIED AUDIT TRAIL data dictionary view as follows:

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

col dbusername format a20
col sql_text format a66
col audit_type format a17

SELECT DBUSERNAME, SQL_TEXT, AUDIT_TYPE
FROM UNIFIED_AUDIT_TRAIL
WHERE OBJECT SCHEMA = 'HR' AND OBJECT NAME = 'EMPLOYEES';
```

Output similar to the following appears:

DBUSERNAME	SQL_TEXT	AUDIT_TYPE
SMAVRIS	UPDATE HR.EMPLOYEES SET SALARY = 38000 WHERE LAST NAME = 'Mavris'	FineGrainedAudit

The audit trail captures the SQL statement that Susan Mavris ran that affected the SALARY column in the HR.EMPLOYEES table. The first statement she ran, in which she asked about her current salary, was not recorded because it was not affected by the audit policy. This is because Oracle Database executes the audit function as an autonomous transaction, committing only the actions of the handler_module setting and not any user transaction. The function has no effect on any user SQL transaction.

27.4.8.8 Step 7: Remove the Components of This Tutorial

If you no longer need the components of this tutorial, then you can remove them.

 Connect to SQL*Plus as user SYSTEM privilege, and then drop users fga_admin (including the objects in the fga admin schema), fga auditor, and smavris.

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

DROP USER fga_admin CASCADE;
DROP USER fga_auditor;
DROP USER smavris;
```

2. Connect as user HR and remove the loftiness of Susan Mavris's salary.

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

UPDATE HR.EMPLOYEES SET SALARY = 6500 WHERE LAST NAME = 'Mavris';
```

3. If you want, lock and expire HR, unless other users want to use this account:

ALTER USER HR PASSWORD EXPIRE ACCOUNT LOCK;

4. Issue the following ALTER SYSTEM statement to restore the SMTP_OUT_SERVER parameter to the previous value, from Step 5 under Step 1: Install and Configure the UTL_MAIL PL/SQL Package:

ALTER SYSTEM SET SMTP_OUT_SERVER="previous_value";

Enclose this setting in quotation marks. For example:

ALTER SYSTEM SET SMTP OUT SERVER="some imap server.example.com"

5. Restart the database instance.

27.5 Audit Policy Data Dictionary Views

Data dictionary and dynamic views can be used to find detailed auditing information.

Table 27-20 lists these views.



Tip:

To find error information about audit policies, check the trace files. The $\tt USER DUMP DEST initialization parameter sets$ the location of the trace files.

Table 27-20 Views That Display Information about Audited Activities

View	Description
ALL_AUDIT_POLICIES	Displays information about all fine-grained audit policies
ALL_DEF_AUDIT_OPTS	Lists default object-auditing options that are to be applied when objects are created
AUDIT_UNIFIED_CONTEXTS	Describes application context values that have been configured to be captured in the audit trail
AUDIT_UNIFIED_ENABLED_POLICIES	Describes all unified audit policies that are enabled in the database
AUDIT_UNIFIED_POLICIES	Describes all unified audit policies created in the database
AUDIT_UNIFIED_POLICY_COMMENTS	Shows the description of each unified audit policy, if a description was entered for the unified audit policy using the COMMENT SQL statement
AUDITABLE_SYSTEM_ACTIONS	Maps the auditable system action numbers to the action names
CDB_UNIFIED_AUDIT_TRAIL	Similar to the UNIFIED_AUDIT_TRAIL view, displays the audit records but from all PDBs in a multitenant environment. This view is available only in the CDB root and must be queried from there.
DBA_AUDIT_POLICIES	Displays information about fine-grained audit policies
DBA_SA_AUDIT_OPTIONS	Describes audited Oracle Label Security events performed by users, and indicates if the user's action failed or succeeded
DBA_XS_AUDIT_TRAIL	Displays audit trail information related to Oracle Database Real Application Security
DV\$CONFIGURATION_AUDIT	Displays configuration changes made by Oracle Database Vault administrators
DV\$ENFORCEMENT_AUDIT	Displays user activities that are affected by Oracle Database Vault policies

Table 27-20 (Cont.) Views That Display Information about Audited Activities

View	Description
SYSTEM_PRIVILEGE_MAP (table)	Describes privilege (auditing option) type codes. This table can be used to map privilege (auditing option) type numbers to type names.
USER_AUDIT_POLICIES	Displays information about all fine-grained audit policies on table and views owned by the current user
UNIFIED_AUDIT_TRAIL	Displays all audit records
V\$OPTION	You can query the PARAMETER column for Unified Auditing to find if unified auditing is enabled
V\$XML_AUDIT_TRAIL	Displays standard, fine-grained, ${\tt SYS},$ and mandatory audit records written in XML format files.

Related Topics

Oracle Database Reference



Administering the Audit Trail

Users who have been granted the AUDIT_ADMIN role can manage the audit trail, archive the audit trail, and purge audit trail records.

- Managing the Unified Audit Trail
 Auditing is enabled by default, but you can control when audit records are written to disk.
- Archiving the Audit Trail
 You can archive the traditional operating system, unified database, and traditional database audit trails.
- Purging Audit Trail Records
 The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.
- Audit Trail Management Data Dictionary Views
 Oracle Database provides data dictionary views that list information about audit trail
 management settings.

28.1 Managing the Unified Audit Trail

Auditing is enabled by default, but you can control when audit records are written to disk.

- When and Where Are Audit Records Created?
 Auditing is always enabled. Oracle Database generates audit records during or after the execution phase of the audited SQL statements.
- Activities That Are Mandatorily Audited
 Certain security sensitive database activities are always audited and such audit configuration cannot be disabled.
- How Do Cursors Affect Auditing?
 For each execution of an auditable operation within a cursor, Oracle Database inserts one audit record into the audit trail.
- Writing the Unified Audit Trail Records to the AUDSYS Schema
 Oracle Database automatically writes audit records to an internal relational table in the AUDSYS schema.
- Writing the Unified Audit Trail Records to SYSLOG or the Windows Event Viewer
 You can write the unified audit trail records to SYSLOG or the Windows Event Viewer by
 setting an initialization parameter.
- When Audit Records Are Written to the Operating System
 In situations where the database table is unable to accept unified audit records, these records will be written to operating system spillover audit files (.bin format).
- Moving Operating System Audit Records into the Unified Audit Trail
 Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.
- Exporting and Importing the Unified Audit Trail Using Oracle Data Pump
 You can include the unified audit trail in Oracle Database Pump export and import dump files.

Disabling Unified Auditing
 You can disable unified auditing.

Related Topics

Purging Audit Trail Records

The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

28.1.1 When and Where Are Audit Records Created?

Auditing is always enabled. Oracle Database generates audit records during or after the execution phase of the audited SQL statements.

Oracle Database individually audits SQL statements inside PL/SQL program units, as necessary, when the program unit is run.

To improve read performance of the unified audit trail, the unified audit records are written immediately to disk to an internal relational table in the AUDSYS schema. In the previous release, the unified audit records were written to SecureFile LOBs. If you had migrated to unified auditing in Oracle Database 12c release 1 (12.1), then you can manually transfer the unified audit records from the SecureFile LOBS to this internal table. If the version of the database that you are using supports partitioned tables, then this internal table is a partitioned table. In this case, you can modify the partition interval of the table by using the DBMS_AUDIT_MGMT.ALTER_PARTITION_INTERVAL procedure. The partitioned version of this table is based on the EVENT_TIMESTAMP timestamp as a partition key with a default partition interval of one month. If the database version does not support partitioning, then the internal table is a regular, non-partitioned table.

The generation and insertion of an audit trail record is independent of the user transaction being committed. That is, even if a user transaction is rolled back, the audit trail record remains committed.

Statement and privilege audit options from unified audit policies that are in effect at the time a database user connects to the database remain in effect for the duration of the session. When the session is already active, setting or changing statement or privilege unified audit options does not take effect in that session. The modified statement or privilege audit options take effect only when the current session ends and a new session is created.

In contrast, changes to schema object audit options become immediately effective for current sessions.

By default, audit trail records are written to the AUDSYS schema in the SYSAUX tablespace. You can designate a different tablespace, including one that is encrypted, by using the DBMS AUDIT MGMT.SET AUDIT TRAIL LOCATION procedure.

Related Topics

- Writing the Unified Audit Trail Records to the AUDSYS Schema
 Oracle Database automatically writes audit records to an internal relational table in the AUDSYS schema.
- Oracle Database Upgrade Guide
- Oracle Database PL/SQL Packages and Types Reference

28.1.2 Activities That Are Mandatorily Audited

Certain security sensitive database activities are always audited and such audit configuration cannot be disabled.

The UNIFIED_AUDIT_TRAIL data dictionary view captures activities from administrative users such as SYSDBA, SYSBACKUP, and SYSKM. You do not need to audit the unified audit trail. The unified audit trail resides in a read-only table in the AUDSYS schema. Hence, DMLs are not permitted on the unified audit trail views. Even DML and DDL operations on the underlying dictionary tables from AUDSYS schema are not permitted.

Mandatorily audited activities will have audit policy by name ORA\$MANDATORY in the UNIFIED_AUDIT_POLICIES column of the UNIFIED_AUDIT_TRAIL data dictionary view. The ORA\$MANDATORY is always listed first in this column, if there are other unified audit policies that are tracking mandatorily audited activities.

The SYSTEM_PRIVILEGE_USED column shows the type of administrative privilege that was used for the activity.

Mandatorily Audited Non-Audit-Related Activities

ORADEBUG utility

Mandatorily Audited Audit-Related Activities

- CREATE AUDIT POLICY
- ALTER AUDIT POLICY
- DROP AUDIT POLICY
- AUDIT
- NOAUDIT
- EXECUTE of the DBMS FGA PL/SQL package
- EXECUTE of the DBMS AUDIT MGMT PL/SQL package
- ALTER TABLE attempts on the AUDSYS audit trail table (remember that this table cannot be altered)
- Top level statements by the administrative users SYS, SYSDBA, SYSDER, SYSASM, SYSBACKUP, SYSDG, and SYSKM, until the database opens. When the database opens, Oracle Database audits these users using the audit configurations in the system—not just the ones that were applied using the BY clause in the AUDIT statement, for example, but those that were applied for all users when AUDIT statement does not have a BY clause or when the EXCEPT clause was used and these users were not excluded.
- All user-issued DML statements on the SYS.AUD\$ and SYS.FGA LOG\$ dictionary tables
- Any attempts to modify the data or metadata of the unified audit internal table. SELECT statements on this table are not audited by default or mandatorily.
- All configuration changes that are made to Oracle Database Vault

Mandatorily Audited Access to Sensitive Columns in the Oracle Optimizer Dictionary Tables

Be aware that internal access to these table columns by the <code>DBMS_STATS</code> package does not generate mandatory audit records. The optimizer dictionary tables are as follows:

Optimizer Dictionary Table	Columns	
SYS.HIST_HEAD\$	minimum, maximum, lowval, hival	
SYS.HISTGRM\$	endpoint, epvalue_raw	



Optimizer Dictionary Table	Columns
SYS.WRI\$_OPTSTAT_HISTHEAD_HISTORY	minimum, maximum, lowval, hival
SYS.WRI\$_OPSTAT_HISTGRM_HISTORY	endpoint, epvalue_raw

Related Topics

Auditing Administrative Users

You can create unified audit policies to capture the actions of administrative user accounts, such as sys.

28.1.3 How Do Cursors Affect Auditing?

For each execution of an auditable operation within a cursor, Oracle Database inserts one audit record into the audit trail.

Events that cause cursors to be reused include the following:

- An application, such as Oracle Forms, holding a cursor open for reuse
- Subsequent execution of a cursor using new bind variables
- Statements executed within PL/SQL loops where the PL/SQL engine optimizes the statements to reuse a single cursor

Auditing is *not* affected by whether or not a cursor is shared. Each user creates her or his own audit trail records on first execution of the cursor.

28.1.4 Writing the Unified Audit Trail Records to the AUDSYS Schema

Oracle Database automatically writes audit records to an internal relational table in the AUDSYS schema.

In Oracle Database 12c release 1 (12.1), you had the option of queuing the audit records in memory (queued-write mode) and be written periodically to the AUDSYS schema audit table. However, starting with Oracle Database 12c release 2 (12.2), immediate-write mode and queued-write mode are deprecated. The parameters that controlled them (UNIFIED_AUDIT_SGA_QUEUE_SIZE, DBMS_AUDIT_MGMT.AUDIT_TRAIL_IMMEDIATE_WRITE, and DBMS_AUDIT_MGMT.AUDIT_TRAIL_QUEUED_WRITE), while still viewable, no longer have any functionality.

The new functionality of having audit records always written to a relational table in the AUDSYS schema prevents the risk of audit records being lost in the event of an instance crash or during a Shutdown Abort operation. The new functionality also improves the performance of the audit trail and the database as a whole.

If you have upgraded from Oracle Database 12c release 1 (12.1) and migrated to unified auditing in that release, then Oracle recommends that you use the <code>DBMS_AUDIT_MGMT.TRANSFER_UNIFIED_AUDIT_RECORDS</code> procedure to transfer the audit records as generated in the previous release to the <code>AUDSYS</code> audit internal table. *Oracle Database Upgrade Guide* provides information about transferring unified audit records after an upgrade.

Related Topics

Oracle Database Upgrade Guide



28.1.5 Writing the Unified Audit Trail Records to SYSLOG or the Windows Event Viewer

You can write the unified audit trail records to SYSLOG or the Windows Event Viewer by setting an initialization parameter.

- About Writing the Unified Audit Trail Records to SYSLOG or the Windows Event Viewer
 With this feature, you can copy some of the key unified audit fields to SYSLOG or the
 Windows Event Viewer.
- Enabling SYSLOG and Windows Event Viewer Captures for the Unified Audit Trail
 You can write a subset of unified audit trail records to the UNIX SYSLOG or to the
 Windows Event Viewer.

28.1.5.1 About Writing the Unified Audit Trail Records to SYSLOG or the Windows Event Viewer

With this feature, you can copy some of the key unified audit fields to SYSLOG or the Windows Event Viewer.

Unlike traditional audit, only key fields of unified audit records in the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view are copied to SYSLOG. SYSLOG records in a unified audit environment provide proof of operational integrity.

You can configure this feature on both UNIX and Microsoft Windows systems. On Windows systems, you either enable it or disable it. If enabled, it writes the records to the Windows Event Viewer.

On UNIX systems, you can fine-tune the capture of unified audit trail records for SYSLOG to specify the facility where the SYSLOG records are sent and the severity level of the records (for example, DEBUG if it is capturing debugging-related messages).

Table 28-1 maps the names given to the unified audit records fields that are written to SYSLOG and the Windows Event Viewer to the corresponding column names in the UNIFIED AUDIT TRAIL view.

Table 28-1 Audit Record Field Names for SYSLOG and the Windows Event Viewer

Field Name	Column Name in UNIFIED_AUDIT_TRAI L	Column Type	Column Description
TYPE	AUDIT_TYPE	NUMBER	Type of the audit record
DBID	DBID	NUMBER	Database identifier
SESID	SESSION_ID	NUMBER	Session identifier
CLIENTID	CLIENT_IDENTIFIER	VARCHAR2	Client identifier in the session
STMTID	STATEMENT_ID	NUMBER	Identifier for each statement run in the system
DBUSER	DB_USERNAME	VARCHAR2	Session user
CURUSER	CURRENT_USER	VARCHAR2	Effective user for the audited event



Table 28-1	(Cont.) Audit Record Field Names for SYSLOG and the Windows Event
Viewer	

Field Name	Column Name in UNIFIED_AUDIT_TRAI L	Column Type	Column Description
ACTION	ACTION	NUMBER	Action code of the audited event
RETCODE	RETURN_CODE	NUMBER	Return code for the audited event
SCHEMA	OBJECT_SCHEMA	VARCHAR2	Schema name of the object
OBJNAME	OBJECT_NAME	VARCHAR2	Name of the object
PDB_GUID	NULL (there are no columns in UNIFIED_AUDIT_TRAIL for this field)	VARCHAR2	GUID of the container in which the unified audit record is generated

28.1.5.2 Enabling SYSLOG and Windows Event Viewer Captures for the Unified Audit Trail

You can write a subset of unified audit trail records to the UNIX SYSLOG or to the Windows Event Viewer.

- Locate the init.ora initialization file, which by default is in the \$ORACLE_HOME/dbs directory.
- 2. Edit the init.ora file to include the UNIFIED AUDIT SYSTEMLOG parameter.

You can set UNIFIED AUDIT SYSTEMLOG in either the CDB root or in a PDB.

In an Oracle Database Real Application Clusters (Oracle RAC) environment, set UNIFIED_AUDIT_SYSTEMLOG to the same value on each Oracle RAC instance.

• On Windows, set UNIFIED_AUDIT_SYSTEMLOG to either TRUE or FALSE. TRUE writes the SYSLOG values to the Windows Event Viewer; FALSE disables the parameter. On Windows, the default is FALSE. For example:

```
UNIFIED_AUDIT_SYSTEMLOG = TRUE
```

On UNIX systems, use the following syntax:

```
UNIFIED AUDIT SYSTEMLOG = 'facility clause.priority clause'
```

There is no default setting for <code>UNIFIED_AUDIT_SYSTEMLOG</code> on UNIX systems. In this specification:

- facility_clause refers to the facility to which you will write the audit trail records.
 Valid choices are USER and LOCAL. If you enter LOCAL, then optionally append 0-7 to designate a local custom facility for the SYSLOG records.
- priority_clause refers to the type of warning in which to categorize the record.
 Valid choices are NOTICE, INFO, DEBUG, WARNING, ERR, CRIT, ALERT, and EMERG.

For example:

```
UNIFIED AUDIT SYSTEMLOG = 'LOCAL7.EMERG'
```

 On UNIX platforms, to write unified audit records to SYSLOG set the UNIFIED_AUDIT_COMMON_SYSTEMLOG parameter to either TRUE or FALSE in the init.ora file in the root.

Setting UNIFIED_AUDIT_COMMON_SYSTEMLOG to TRUE writes predefined columns of unified audit records from common unified audit policies to SYSLOG. FALSE disables these columns from being written to SYSLOG.

You cannot set this parameter in a pluggable database (PDB). There is no Windows equivalent of the ${\tt UNIFIED}$ AUDIT COMMON SYSTEMLOG parameter.

4. Add the audit file destination to the SYSLOG configuration file /etc/syslog.conf.

For example, assuming you had set the <code>UNIFIED_AUDIT_SYSTEMLOG</code> to <code>LOCAL7.EMERG</code>, enter the following:

```
local7.emerg /var/log/audit.log
```

This setting logs all emergency messages to the /var/log/audit.log file.

5. Restart the SYSLOG logger.

```
$/etc/rc.d/init.d/syslog restart
```

Now, all audit records will be captured in the file /var/log/audit.log through the syslog daemon.

- 6. Log back in to the database instance.
- Restart the database.

For example:

```
SHUTDOWN IMMEDIATE STARTUP
```

If you set UNIFIED AUDIT SYSTEMLOG in a PDB, then close and reopen the PDB:

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

Related Topics

Oracle Database Reference

28.1.6 When Audit Records Are Written to the Operating System

In situations where the database table is unable to accept unified audit records, these records will be written to operating system spillover audit files (.bin format).

The default locations for unified audit spillover .bin files are as follows:

For pluggable databases (PDBs): \$ORACLE_BASE/audit/\$ORACLE_SID/PDB_GUID

 For non-consolidated databases, or for the CDB root: \$ORACLE_BASE/ audit/\$ORACLE_SID/

The ability to write to the database table can fail in situations such as the following: the audit tablespace is offline, the tablespace is read-only, the tablespace is full, the database is read-only, and so on. The unified audit records will continue to be written to OS spillover files until the OS disk space becomes full. At this point, when there is no room in the OS for the audit records, user auditable transactions will fail with ORA-02002 error while writing to audit trail errors. To prevent this problem, Oracle recommends that you purge the audit trail on a regular basis.

Related Topics

Purging Audit Trail Records

The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

28.1.7 Moving Operating System Audit Records into the Unified Audit Trail

Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.

When the database is not writable (such as during database mounts), if the database is closed, or if it is read-only, then Oracle Database writes the audit records to these external files. The default location for these external files is the <code>\$ORACLE BASE/audit/\$ORACLE SID</code> directory.

You can load the files into the database by running the

DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES procedure. Be aware that if you are moving a large number of operating system audit records in the external files, performance may be affected.

To move the audit records in these files to the AUDSYS schema audit table when the database is writable:

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

For example:

CONNECT aud_admin
Enter password: password
Connected.

In a multitenant environment, log into the CDB root with the AUDIT_ADMIN role. Before you can upgrade to the current release or Oracle Database, you must execute the DBMS_AUDIT_MGMT.LOAD_UNIFIED_AUDIT_FILES procedure from the CDB root to avoid losing operating system spillover files during the upgrade process.

For example:

CONNECT c##aud_admin
Enter password: password
Connected.

2. Ensure that the database is open and writable.

For a non-CDB architecture, to find if the database is open and writable, query the V\$DATABASE view.

For example, in a CDB environment:

SELECT NAME, OPEN_MODE FROM V\$DATABASE;



In a multitenant environment, you can run the show pdbs command to find information about PDBs associated with the current instance.

3. Run the DBMS AUDIT MGMT.LOAD UNIFIED AUDIT FILES procedure.

```
EXEC DBMS AUDIT MGMT.LOAD UNIFIED AUDIT FILES;
```

4. In a multitenant environment, if you want to load individual PDB audit records, then log in to each PDB and run the DBMS AUDIT MGMT.LOAD UNIFIED AUDIT FILES procedure again.

The audit records are loaded into the AUDSYS schema audit table immediately, and then deleted from the <code>\$ORACLE BASE/audit/\$ORACLE SID</code> directory.

28.1.8 Exporting and Importing the Unified Audit Trail Using Oracle Data Pump

You can include the unified audit trail in Oracle Database Pump export and import dump files.

The unified audit trail is automatically included in either full database or partial database export and import operations using Oracle Data Pump. As part of the schema level export or import operation, Oracle Database does not include the audit policy's metadata in the SYS schema during the export or import operation. Instead, use full export (expdp) or import (impdp) for the export and import of the metadata in unified audit policies.

For example, for a partial database export operation that does not use schema level export or import, if you wanted to export only the unified audit trail tables, then you could enter the following commands:

- In SQL*Plus, move any operating system audit records that have been written to the spillover audit files to the unified audit trail table. Doing so ensures that all records will be exported.
- 2. From the operating system prompt, run the following command:

```
expdp system
full=y
directory=aud_dp_dir
logfile=audexp_log.log
dumpfile=audexp_dump.dmp
version=18.02.00.02.00
INCLUDE=AUDIT_TRAILS
```

Next, you can import all the exported content by reading the export dump file. This operation imports only the unified audit trail tables.

```
impdp system
full=y
directory=aud_dp_dir
dumpfile=audexp_dump.dmp
logfile=audimp_log.log
```

Password: password

Password: password



You do not need to perform any special configuration to achieve this operation. However, you must have the <code>EXP_FULL_DATABASE</code> role if you are performing the export operation and the <code>IMP_FULL_DATABASE</code> role if you are performing the import operation.

Related Topics

Moving Operating System Audit Records into the Unified Audit Trail
 Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.

28.1.9 Disabling Unified Auditing

You can disable unified auditing.

1. Disable any unified audit policies that are currently enabled.

This step prevents the database from going into mixed mode auditing after you complete this procedure.

- a. Log into the database instance as a user who has been granted the AUDIT ADMIN role.
- b. Query the POLICY_NAME and ENABLED_OPT columns of the AUDIT_UNIFIED_ENABLED_POLICIES data dictionary view to find unified audit policies that are enabled.
- c. Run the NOAUDIT POLICY statement to disable each enabled policy.

For example, to disable a policy that had been applied to user psmith:

```
NOAUDIT POLICY audit pol BY psmith;
```

2. Connect as user SYS with the SYSOPER privilege.

```
CONNECT sys as sysoper Enter password: password
```

In a multitenant environment, this command connects you to the root.

Shut down the database.

For example:

```
SHUTDOWN IMMEDIATE
```

In a multitenant environment, this command shuts down all PDBs in the CDB.

4. On a Windows system, stop the <code>OracleServiceSID</code> process.

```
net stop OracleServiceSID
```

- Depending on your platform, do the following:
 - UNIX systems: Run the following commands:

```
cd $ORACLE_HOME/rdbms/lib
make -f ins rdbms.mk uniaud off ioracle
```

 Windows systems: Rename the %ORACLE_HOME%/bin/orauniaud19.dll file to %ORACLE_HOME%/bin/orauniaud19.dll.dbl.

In a multitenant environment, these actions disable unified auditing in all PDBs in the CDB.

On a Windows system, before you restart the database, restart the OracleServiceSID process.

net start OracleServiceSID

7. In SQL*Plus, restart the database.

STARTUP

In a multitenant environment, this command restarts all PDBs in the CDB.

Related Topics

About Mixed Mode Auditing

Mixed mode auditing enables both traditional (that is, the audit facility from releases earlier than release 12c) and the new audit facilities (unified auditing).

Disabling Unified Audit Policies
 You can use the NOAUDIT POLICY statement to disable a unified audit policy.

28.2 Archiving the Audit Trail

You can archive the traditional operating system, unified database, and traditional database audit trails.

- Archiving the Traditional Operating System Audit Trail
 You can create an archive of the traditional operating system audit files after you have
 upgraded Oracle Database.
- Archiving the Unified and Traditional Database Audit Trails
 You should periodically archive and then purge the audit trail to prevent it from growing too large

28.2.1 Archiving the Traditional Operating System Audit Trail

You can create an archive of the traditional operating system audit files after you have upgraded Oracle Database.

To archive the traditional operating system audit trail from an upgraded database, use your platform-specific operating system tools to create an archive of the traditional operating system audit files.

- Use the following methods to archive the traditional operating system audit files:
 - Use Oracle Audit Vault and Database Firewall. You install Oracle Audit Vault and Database Firewall separately from Oracle Database.
 - Create tape or disk backups. You can create a compressed file of the audit files, and then store it on tapes or disks. Consult your operating system documentation for more information.

Afterwards, you should purge (delete) the traditional operating system audit records to facilitate audit trail management.

Related Topics

Moving Operating System Audit Records into the Unified Audit Trail
 Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.



Purging Audit Trail Records

The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

28.2.2 Archiving the Unified and Traditional Database Audit Trails

You should periodically archive and then purge the audit trail to prevent it from growing too large.

Archiving and purging facilitate the purging of the database audit trail.

You can create an archive of the unified and traditional database audit trail by using Oracle Audit Vault and Database Firewall. You install Oracle Audit Vault and Database Firewall separately from Oracle Database.

After you complete the archive, you can purge the database audit trail contents.

 To archive the unified, traditional standard, and traditional fine-grained audit records, copy the relevant records to a normal database table.

For example:

```
INSERT INTO table SELECT ... FROM UNIFIED_AUDIT_TRAIL ...;
INSERT INTO table SELECT ... FROM SYS.AUD$ ...;
INSERT INTO table SELECT ... FROM SYS.FGA LOG$ ...;
```

Related Topics

Purging Audit Trail Records

The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

28.3 Purging Audit Trail Records

The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

- About Purging Audit Trail Records
 - You can use a variety of ways to purge audit trail records.
- Selecting an Audit Trail Purge Method
 - You can perform the purge on a regularly scheduled basis or at a specified times.
- Scheduling an Automatic Purge Job for the Audit Trail
 Scheduling an automatic purge job requires planning beforehand, such as tuning the online and archive redo log sizes.
- Manually Purging the Audit Trail
 - You can use the <code>DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL</code> procedure to manually purge the audit trail.
- Other Audit Trail Purge Operations
 - Other kinds of audit trail purge include enabling or disabling the audit trail purge job or setting the default audit trail purge job interval.
- Example: Directly Calling a Unified Audit Trail Purge Operation
 You can create a customized archive procedure to directly call a unified audit trail purge operation.

Related Topics

Managing the Unified Audit Trail
 Auditing is enabled by default, but you can control when audit records are written to disk.

28.3.1 About Purging Audit Trail Records

You can use a variety of ways to purge audit trail records.

You should periodically archive and then delete (purge) audit trail records. You can purge a subset of audit trail records or create a purge job that performs at a specified time interval. Oracle Database either purges the audit trail records that were created before the archive timestamp, or it purges all audit trail records. You can purge audit trail records in both readwrite and read-only databases.

The purge process takes into account not just the unified audit trail, but audit trails from earlier releases of Oracle Database. For example, if you have migrated an upgraded database that still has operating system or XML audit records, then you can use the procedures in this section to archive and purge them.

To perform the audit trail purge tasks, you use the <code>DBMS_AUDIT_MGMT PL/SQL</code> package. You must have the <code>AUDIT_ADMIN</code> role before you can use the <code>DBMS_AUDIT_MGMT</code> package. Oracle <code>Database</code> mandatorily audits all executions of the <code>DBMS_AUDIT_MGMT PL/SQL</code> package procedures.

If you have Oracle Audit Vault and Database Firewall installed, the audit trail purge process differs from the procedures described in this manual. For example, Oracle Audit Vault archives the audit trail for you.



Oracle Database audits all deletions from the audit trail, without exception.

Related Topics

Oracle Database PL/SQL Packages and Types Reference

28.3.2 Selecting an Audit Trail Purge Method

You can perform the purge on a regularly scheduled basis or at a specified times.

- Purging the Audit Trail on a Regularly Scheduled Basis
 You can purge all audit records, or audit records that were created before a specified
 timestamp, on a regularly scheduled basis.
- Manually Purging the Audit Trail at a Specific Time
 You can manually purge the audit records right away in a one-time operation, rather than
 creating a purge schedule.

28.3.2.1 Purging the Audit Trail on a Regularly Scheduled Basis

You can purge all audit records, or audit records that were created before a specified timestamp, on a regularly scheduled basis.

For example, you can schedule the purge for every Saturday at 2 a.m.

- If necessary, tune online and archive redo log sizes to accommodate the additional records generated during the audit table purge process.
- 2. Plan a timestamp and archive strategy.
- Optionally, set an archive timestamp for the audit records.
- Create and schedule the purge job.

Related Topics

Scheduling an Automatic Purge Job for the Audit Trail
 Scheduling an automatic purge job requires planning beforehand, such as tuning the online and archive redo log sizes.

28.3.2.2 Manually Purging the Audit Trail at a Specific Time

You can manually purge the audit records right away in a one-time operation, rather than creating a purge schedule.

- 1. If necessary, tune online and archive redo log sizes to accommodate the additional records generated during the audit table purge process.
- 2. Plan a timestamp and archive strategy.
- 3. Optionally, set an archive timestamp for the audit records.
- Run the purge operation.

Related Topics

Manually Purging the Audit Trail
 You can use the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL procedure to manually purge the
 audit trail.

28.3.3 Scheduling an Automatic Purge Job for the Audit Trail

Scheduling an automatic purge job requires planning beforehand, such as tuning the online and archive redo log sizes.

- About Scheduling an Automatic Purge Job
 You can purge the entire audit trail, or only a portion of the audit trail that was created
 before a timestamp.
- Step 1: If Necessary, Tune Online and Archive Redo Log Sizes
 The purge process may generate additional redo logs.
- Step 2: Plan a Timestamp and Archive Strategy
 You must record the timestamp of the audit records before you can archive them.
- Step 3: Optionally, Set an Archive Timestamp for Audit Records
 If you want to delete all of the audit trail, then you can bypass this step.
- Step 4: Create and Schedule the Purge Job
 You can use the DBMS AUDIT MGMT PL/SQL package to create and schedule the purge job.

28.3.3.1 About Scheduling an Automatic Purge Job

You can purge the entire audit trail, or only a portion of the audit trail that was created before a timestamp.

The individual audit records created before the timestamp can be purged.

Be aware that purging the audit trail, particularly a large one, can take a while to complete. Consider scheduling the purge job so that it runs during a time when the database is not busy.

You can create multiple purge jobs for different audit trail types, so long as they do not conflict. For example, you can create a purge job for the standard audit trail table and then the fine-grained audit trail table. However, you cannot then create a purge job for both or all types, that is, by using the <code>DBMS_AUDIT_MGMT.AUDIT_TRAIL_DB_STD</code> or <code>DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL</code> property. In addition, be aware that the jobs created by the <code>DBMS_SCHEDULER PL/SQL</code> package do not execute on a read-only database. An automatic purge job created with <code>DBMS_AUDIT_MGMT</code> uses the <code>DBMS_SCHEDULER</code> package to schedule the tasks. Therefore, these jobs cannot run on a database or PDB that is open in read-only mode.

28.3.3.2 Step 1: If Necessary, Tune Online and Archive Redo Log Sizes

The purge process may generate additional redo logs.

 If necessary, tune online and archive redo log sizes to accommodate the additional records generated during the audit table purge process.

In a unified auditing environment, the purge process does not generate as many redo logs as in a mixed mode auditing environment, so if you have migrated to unified auditing, then you may want to bypass this step.



Oracle Database Administrator's Guide for more information about tuning log files

28.3.3.3 Step 2: Plan a Timestamp and Archive Strategy

You must record the timestamp of the audit records before you can archive them.

To find the timestamp date, query the DBA AUDIT MGMT LAST ARCH TS data dictionary view.

Later on, when the purge takes place, Oracle Database purges only the audit trail records that were created before the date of this archive timestamp.

After you have timestamped the records, you are ready to archive them.

Related Topics

- Step 3: Optionally, Set an Archive Timestamp for Audit Records
 If you want to delete all of the audit trail, then you can bypass this step.
- Archiving the Audit Trail
 You can archive the traditional operating system, unified database, and traditional
 database audit trails.

28.3.3.4 Step 3: Optionally, Set an Archive Timestamp for Audit Records

If you want to delete all of the audit trail, then you can bypass this step.

You can set a timestamp for when the last audit record was archived. Setting an archive timestamp provides the point of cleanup to the purge infrastructure. If you are setting a timestamp for a read-only database, then you can use the

DBMS AUDIT.MGMT.GET LAST ARCHIVE TIMESTAMP function to find the last archive timestamp

that was configured for the instance on which it was run. For a read-write database, you can query the DBA AUDIT MGMT LAST ARCH TS data dictionary view.

To find the last archive timestamps for the unified audit trail, you can query the DBA_AUDIT_MGMT_LAST_ARCH_TS data dictionary view. After you set the timestamp, all audit records in the audit trail that indicate a time earlier than that timestamp are purged when you run the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL PL/SQL procedure. If you want to clear the archive timestamp setting, see Clearing the Archive Timestamp Setting.

If you are using Oracle Database Real Application Clusters, then use Network Time Protocol (NTP) to synchronize the time on each computer where you have installed an Oracle Database instance. For example, suppose you set the time for one Oracle RAC instance node at 11:00:00 a.m. and then set the next Oracle RAC instance node at 11:00:05. As a result, the two nodes have inconsistent times. You can use Network Time Protocol (NTP) to synchronize the times for these Oracle RAC instance nodes.

To set the timestamp for the purge job:

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

In a multitenant environment, log into either the root or the PDB in which you want to schedule the purge job. In most cases, you may want to schedule the purge job on individual PDBs.

For example, to log into a PDB called hrpdb:

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.
```

2. Run the DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP PL/SQL procedure to set the timestamp.

For example:

```
BEGIN

DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP(
   AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
   LAST_ARCHIVE_TIME => '12-OCT-2013 06:30:00.00',
   RAC_INSTANCE_NUMBER => 1,
   CONTAINER => DBMS_AUDIT_MGMT.CONTAINER_CURRENT);
END;
//
```

In this example:

AUDIT_TRAIL_TYPE specifies the audit trail type.
 DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED sets it for the unified audit trail.

For upgraded databases that still have traditional audit data from previous releases:

- DBMS_AUDIT_MGMT.AUDIT_TRAIL_AUD_STD is used for the traditional standard audit trail table, AUD\$. (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD is used for the traditional fine-grained audit trail table, FGA_LOG\$. (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS is used for the traditional operating system audit trail files with the .aud extension. (This setting does not apply to Windows Event Log entries.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML is used for the XML traditional operating system audit trail files.

To archive records from the <code>AUDSYS.AUD\$UNIFIED</code> table or from the operating system spillover files:

- DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_TABLE archives records from the AUDSYS.AUD\$UNIFIED table.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_FILES archives records from the operating system spillover files in each database (primary or standby).
- LAST_ARCHIVE_TIME specifies the timestamp in YYYY-MM-DD HH:MI:SS.FF UTC
 (Coordinated Universal Time) format for AUDIT_TRAIL_UNIFIED, AUDIT_TRAIL_AUD_STD,
 and AUDIT_TRAIL_FGA_STD, and in the Local Time Zone for AUDIT_TRAIL_OS and
 AUDIT_TRAIL_XML. Do not enter a future system date or timestamp (for example,
 SYSDATE + 1, or a date in the future) for this value.
- RAC_INSTANCE_NUMBER specifies the instance number for an Oracle RAC installation. This setting is not relevant for single instance databases. If you specified the DBMS_AUDIT_MGMT.AUDIT_TRAIL_AUD_STD or DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD audit trail types, then you can omit the RAC_INSTANCE_NUMBER argument. This is because there is only one AUD\$ or FGA_LOG\$ table, even for an Oracle RAC installation. The default is NULL. You can find the instance number for the current instance by issuing the SHOW_PARAMETER_INSTANCE_NUMBER command in SQL*Plus.
- CONTAINER applies the timestamp to a multitenant environment.
 DBMS_AUDIT_MGMT.CONTAINER_CURRENT specifies the current PDB;
 DBMS_AUDIT_MGMT.CONTAINER_ALL applies to all PDBs in the multitenant environment.

Note that you can set CONTAINER to DBMS_MGMT.CONTAINER_ALL only from the root, and DBMS_MGMT.CONTAINER_CURRENT only from a PDB.

Typically, after you set the timestamp, you can use the <code>DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL PL/SQL</code> procedure to remove the audit records that were created before the timestamp date.

28.3.3.5 Step 4: Create and Schedule the Purge Job

You can use the DBMS AUDIT MGMT PL/SQL package to create and schedule the purge job.

 Create and schedule the purge job by running the DBMS_AUDIT_MGMT.CREATE_PURGE_JOB PL/SQL procedure.

For example:

In this example:

AUDIT_TRAIL_TYPE: Specifies the audit trail type. DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED sets it for the unified audit trail.

For upgraded databases that still have audit data from previous releases:

- DBMS_AUDIT_MGMT.AUDIT_TRAIL_AUD_STD is used for the standard audit trail table,
 AUD\$. (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD is used for the fine-grained audit trail table,
 FGA_LOG\$. (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_DB_STD is used for both standard and fine-grained audit trail tables. (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS is used for the operating system audit trail files with the .aud extension. (This setting does not apply to Windows Event Log entries.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML is used for the XML operating system audit trail files.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_FILES is used for both operating system and XML audit trail files.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL is used for all audit trail records, that is, both
 database audit trail and operating system audit trail types. (This setting does not apply
 to read-only databases.)

To purge records from the AUDSYS.AUD\$UNIFIED table or from the operating system spillover files:

- DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_TABLE purges records from the AUDSYS.AUD\$UNIFIED table.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_FILES purges records from the operating system spillover files in each database (primary or standby).
- AUDIT_TRAIL_PURGE_INTERVAL specifies the hourly interval for this purge job to run. The timing begins when you run the DBMS_AUDIT_MGMT.CREATE_PURGE_JOB procedure, in this case, 12 hours after you run this procedure. Later on, if you want to update this value, run the DBMS_AUDIT_MGMT.SET_PURGE_JOB_INTERVAL procedure.
- USE_LAST_ARCH_TIMESTAMP accepts either of the following settings:
 - TRUE deletes audit records created before the last archive timestamp. To check the last recorded timestamp, query the LAST_ARCHIVE_TS column of the DBA_AUDIT_MGMT_LAST_ARCH_TS data dictionary view for read-write databases and the DBMS_AUDIT_MGMT.GET_LAST_ARCHIVE_TIMESTAMP function for read-only databases. The default value is TRUE. Oracle recommends that you set USE_LAST_ARCH_TIMESTAMP to TRUE.
 - FALSE deletes all audit records without considering last archive timestamp. Be careful
 about using this setting, in case you inadvertently delete audit records that should not
 have been deleted.
- CONTAINER is used for a multitenant environment to define where to create the purge job. If you set CONTAINER to DBMS_AUDIT_MGMT.CONTAINER_CURRENT, then it is available, visible, and managed only from the current PDB. The DBMS_AUDIT_MGMT.CONTAINER_ALL setting creates the job in the root. This defines the job as a global job, which runs according to the defined job schedule. When the job is invoked, it cleans up audit trails in all the PDBs in the multitenant environment. If you create the job in the root, then it is visible only in the root. Hence, you can enable, disable, and drop it from the root only.



28.3.4 Manually Purging the Audit Trail

You can use the <code>DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL</code> procedure to manually purge the audit trail.

- About Manually Purging the Audit Trail
 You can manually purge the audit trail right away, without scheduling a purge job.
- Using DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL to Manually Purge the Audit Trail After you complete preparatory steps, you can use the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL procedure to manually purge the audit trail.

28.3.4.1 About Manually Purging the Audit Trail

You can manually purge the audit trail right away, without scheduling a purge job.

Similar to a purge job, you can purge audit trail records that were created before an archive timestamp date or all the records in the audit trail. Only the current audit directory is cleaned up when you run this procedure.

For upgraded databases that may still have audit trails from earlier releases, note the following about the DBMS AUDIT MGMT.CLEAN AUDIT TRAIL PL/SQL procedure:

- On Microsoft Windows, because the <code>DBMS_AUDIT_MGMT</code> package does not support cleanup of Windows Event Viewer, setting the <code>AUDIT_TRAIL_TYPE</code> property to <code>DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS</code> has no effect. This is because operating system audit records on Windows are written to Windows Event Viewer. The <code>DBMS_AUDIT_MGMT</code> package does not support this type of cleanup operation.
- On UNIX platforms, if you had set the AUDIT_SYSLOG_LEVEL initialization parameter, then
 Oracle Database writes the operating system log files to syslog files. (Be aware that when
 you configure the use of syslog files, the messages are sent to the syslog daemon
 process. The syslog daemon process does not return an acknowledgement to Oracle
 Database indicating a committed write to the syslog files.) If you set the AUDIT_TRAIL_TYPE
 property to DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS, then the procedure only removes .aud files
 under audit directory (This directory is specified by the AUDIT_FILE_DEST initialization
 parameter).

28.3.4.2 Using DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL to Manually Purge the Audit Trail

After you complete preparatory steps, you can use the <code>DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL</code> procedure to manually purge the audit trail.

- 1. If you have set the AUDIT_SYSLOG_LEVEL initialization parameter so that the audit trail will be written to operating system log files (syslog), then check for the following:
 - Ensure that no one is currently writing to the audit trail files.
 - Ensure that the session ID that is associated with the audit trail files is not owned by the PMON process.

If either of these conditions is true, then the audit trail cannot be purged.

- Follow these steps under Scheduling an Automatic Purge Job for the Audit Trail:
 - Step 1: If Necessary, Tune Online and Archive Redo Log Sizes



- Step 2: Plan a Timestamp and Archive Strategy
- Step 3: Optionally, Set an Archive Timestamp for Audit Records
- If you are using a multitenant environment, then connect to the database in which you created the purge job.

If you created the purge job in the root, then you must log into the root. If you created the purge job in a specific PDB, then log into that PDB.

For example:

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.
```

4. Purge the audit trail records by running the DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL PL/SQL procedure.

For example:

```
BEGIN

DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL(

AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,

USE_LAST_ARCH_TIMESTAMP => TRUE,

CONTAINER => DBMS_AUDIT_MGMT.CONTAINER_CURRENT);

END;
```

In this example:

AUDIT_TRAIL_TYPE: Specifies the audit trail type.
 DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED sets it for the unified audit trail.

For upgraded databases that still have audit data from previous releases:

- DBMS_AUDIT_MGMT.AUDIT_TRAIL_AUD_STD: Standard audit trail table, AUD\$. (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD: Fine-grained audit trail table, FGA_LOG\$.
 (This setting does not apply to read-only databases.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_DB_STD: Both standard and fine-grained audit trail tables. (This setting does not apply to read-only databases)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS: Operating system audit trail files with the .aud extension. (This setting does not apply to Windows Event Log entries.)
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML: XML Operating system audit trail files.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_FILES: Both operating system and XML audit trail files.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL: All audit trail records, that is, both database audit trail and operating system audit trail types. (This setting does not apply to read-only databases.)

To purge records from the AUDSYS.AUD\$UNIFIED table or from the operating system spillover files:

- DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_TABLE purges records from the AUDSYS.AUD\$UNIFIED table.
- DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_FILES purges records from the operating system spillover files in each database (primary or standby).
- USE LAST ARCH TIMESTAMP: Enter either of the following settings:

- TRUE: Deletes audit records created before the last archive timestamp. To set the
 archive timestamp, see Step 3: Optionally, Set an Archive Timestamp for Audit
 Records. The default (and recommended) value is TRUE. Oracle recommends that
 you set USE LAST ARCH TIMESTAMP to TRUE.
- FALSE: Deletes all audit records without considering last archive timestamp. Be careful about using this setting, in case you inadvertently delete audit records that should not have been deleted.
- CONTAINER: Applies the cleansing to a multitenant environment.
 DBMS_AUDIT_MGMT.CONTAINER_CURRENT specifies the local PDB;
 DBMS_AUDIT_MGMT.CONTAINER_ALL applies to all databases.

28.3.5 Other Audit Trail Purge Operations

Other kinds of audit trail purge include enabling or disabling the audit trail purge job or setting the default audit trail purge job interval.

- Enabling or Disabling an Audit Trail Purge Job
 The DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS procedure enables or disables an audit trail purge job.
- Setting the Default Audit Trail Purge Job Interval for a Specified Purge Job You can set a default purge operation interval, in hours, that must pass before the next purge job operation takes place.
- Deleting an Audit Trail Purge Job You can delete existing audit trail purge jobs.
- Clearing the Archive Timestamp Setting
 The DBMS_AUDIT_MGMT.CLEAR_LAST_ARCHIVE_TIMESTAMP procedure can clear the archive timestamp setting.

28.3.5.1 Enabling or Disabling an Audit Trail Purge Job

The <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS</code> procedure enables or disables an audit trail purge job.

In a multitenant environment, where you run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS</code> procedure depends on the location of the purge job, which is determined by the <code>CONTAINER</code> parameter of the <code>DBMS_MGMT.CREATE_PURGE_JOB</code> procedure. If you had set <code>CONTAINER</code> to <code>CONTAINER_ALL</code> (to create the purge job in the root), then you must run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS</code> procedure from the root. If you had set <code>CONTAINER</code> to <code>CONTAINER_CURRENT</code>, then you must run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS</code> procedure from the <code>PDB</code> in which it was created.

 To enable or disable an audit trail purge job, use the DBMS AUDIT MGMT.SET PURGE JOB STATUS PL/SQL procedure.

For example, assuming that you had created the purge job in a the hppdb PDB:

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.

BEGIN

DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS(
AUDIT_TRAIL_PURGE_NAME => 'Audit_Trail_PJ',
AUDIT_TRAIL_STATUS_VALUE => DBMS_AUDIT_MGMT.PURGE_JOB_ENABLE);
```



```
END;
```

In this example:

- AUDIT_TRAIL_PURGE_NAME specifies a purge job called Audit_Trail_PJ. To find existing
 purge jobs, query the JOB_NAME and JOB_STATUS columns of the
 DBA AUDIT MGMT CLEANUP JOBS data dictionary view.
- AUDIT TRAIL STATUS VALUE accepts either of the following properties:
 - DBMS AUDIT MGMT.PURGE JOB ENABLE enables the specified purge job.
 - DBMS AUDIT MGMT.PURGE JOB DISABLE disables the specified purge job.

28.3.5.2 Setting the Default Audit Trail Purge Job Interval for a Specified Purge Job

You can set a default purge operation interval, in hours, that must pass before the next purge job operation takes place.

The interval setting that is used in the <code>DBMS_AUDIT_MGMT.CREATE_PURGE_JOB</code> procedure takes precedence over this setting.

 To set the default audit trail purge job interval for a specific purge job, run the DBMS_AUDIT_MGMT.SET_PURGE_JOB_INTERVAL procedure.

For example, assuming that you had created the purge job in the hrpdb PDB:

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.

BEGIN
DBMS_AUDIT_MGMT.SET_PURGE_JOB_INTERVAL(
AUDIT_TRAIL_PURGE_NAME => 'Audit_Trail_PJ',
AUDIT_TRAIL_INTERVAL_VALUE => 24);
END;
```

In this example:

- AUDIT_TRAIL_PURGE_NAME specifies the name of the audit trail purge job. To find a list of
 existing purge jobs, query the JOB_NAME and JOB_STATUS columns of the
 DBA_AUDIT_MGMT_CLEANUP_JOBS data dictionary view.
- AUDIT_TRAIL_INTERVAL_VALUE updates the default hourly interval set by the
 DBMS_AUDIT_MGMT.CREATE_PURGE_JOB procedure. Enter a value between 1 and 999. The
 timing begins when you run the purge job.

In a multitenant environment, where you run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_INTERVAL</code> procedure depends on the location of the purge job, which is determined by the <code>CONTAINER</code> parameter of the <code>DBMS_MGMT.CREATE_PURGE_JOB</code> procedure. If you had set <code>CONTAINER</code> to <code>CONTAINER_ALL</code>, then the purge job exists in the root, so you must run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS</code> procedure from the root. If you had set <code>CONTAINER</code> to <code>CONTAINER_CURRENT</code>, then you must run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_INTERVAL</code> procedure from the <code>PDB</code> in which it was created.

28.3.5.3 Deleting an Audit Trail Purge Job

You can delete existing audit trail purge jobs.

To find existing purge jobs, query the <code>JOB_NAME</code> and <code>JOB_STATUS</code> columns of the <code>DBA</code> AUDIT MGMT CLEANUP JOBS data dictionary view.

To delete an audit trail purge job, use the DBMS_AUDIT_MGMT.DROP_PURGE_JOB PL/SQL procedure.

For example, assuming that you had created the purge job in the hrpdb PDB:

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.

BEGIN
   DBMS_AUDIT_MGMT.DROP_PURGE_JOB(
   AUDIT_TRAIL_PURGE_NAME => 'Audit_Trail_PJ');
END;
//
```

In a multitenant environment, where you run the <code>DBMS_AUDIT_MGMT.DROP_PURGE_JOB</code> procedure depends on the location of the purge job, which is determined by the <code>CONTAINER</code> parameter of the <code>DBMS_MGMT.CREATE_PURGE_JOB</code> procedure. If you had set <code>CONTAINER</code> to <code>CONTAINER_ALL</code>, then the purge job exists in the root, so you must run the <code>DBMS_AUDIT_MGMT.SET_PURGE_JOB_STATUS</code> procedure from the root. If you had set <code>CONTAINER</code> to <code>CONTAINER_CURRENT</code>, then you must run the <code>DBMS_AUDIT_MGMT.DROP_PURGE_JOB_INTERVAL</code> procedure from the <code>PDB</code> in which it was created.

28.3.5.4 Clearing the Archive Timestamp Setting

The DBMS_AUDIT_MGMT.CLEAR_LAST_ARCHIVE_TIMESTAMP procedure can clear the archive timestamp setting.

To find a history of audit trail log cleanup, you can query the <code>UNIFIED_AUDIT_TRAIL</code> data dictionary view, using the following criteria: <code>OBJECT_NAME</code> is <code>DBMS_AUDIT_MGMT</code>, <code>OBJECT_SCHEMA</code> is <code>SYS</code>, and <code>SQL_TEXT</code> is set to <code>LIKE %DBMS_AUDIT_MGMT</code>. <code>CLEAN_AUDIT_TRAIL</code>%.

• To clear the archive timestamp setting, use the DBMS_AUDIT_MGMT.CLEAR_LAST_ARCHIVE_TIMESTAMP PL/SQL procedure to specify the audit trail type and for a multitenant environment, the container type.

For example, assuming that you had created the purge job in the hppdb PDB:

```
CONNECT aud_admin@hrpdb
Enter password: password
Connected.

BEGIN
   DBMS_AUDIT_MGMT.CLEAR_LAST_ARCHIVE_TIMESTAMP(
   AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
   CONTAINER => DBMS_AUDIT_MGMT.CONTAINER_CURRENT);
END;
//
```

In this example:

AUDIT_TRAIL_TYPE is set for the unified audit trail. If the AUDIT_TRAIL_TYPE property is set
to DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS or DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML, then you
cannot set RAC_INSTANCE_NUMBER to 0. You can omit the RAC_INSTANCE_NUMBER setting if
you set AUDIT_TRAIL_TYPE to DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED.

You can clear the archive timestamps from the AUDSYS.AUD\$UNIFIED table by setting DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_TABLE. To clear the archive timestamps from the operating system spillover files in each database (primary or standby), set DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED_FILES.

CONTAINER applies the timestamp to a multitenant environment.
 DBMS_AUDIT_MGMT.CONTAINER_CURRENT specifies the local PDB;
 DBMS_AUDIT_MGMT.CONTAINER_ALL applies to all databases.

28.3.6 Example: Directly Calling a Unified Audit Trail Purge Operation

You can create a customized archive procedure to directly call a unified audit trail purge operation.

The pseudo code in Example 28-1 creates a database audit trail purge operation that the user calls by invoking the DBMS_ADUIT.CLEAN_AUDIT_TRAIL procedure for the unified audit trail.

The purge operation deletes records that were created before the last archived timestamp by using a loop. The loop archives the audit records, calculates which audit records were archived and uses the <code>SetCleanUpAuditTrail</code> call to set the last archive timestamp, and then calls the <code>CLEAN AUDIT TRAIL</code> procedure. In this example, major steps are in **bold** typeface.

Example 28-1 Directly Calling a Database Audit Trail Purge Operation

```
-- 1. Set the last archive timestamp:
PROCEDURE SetCleanUpAuditTrail()
BEGIN
  CALL FindLastArchivedTimestamp(AUD$);
  DBMS AUDIT MGMT.SET LAST ARCHIVE TIMESTAMP(
  AUDIT_TRAIL_TYPE => DBMS_AUDIT_MGMT.AUDIT_TRAIL_UNIFIED,
LAST_ARCHIVE_TIME => '23-AUG-2013 12:00:00',
CONTAINER => DBMS_AUDIT_MGMT.CONTAINER_CURRENT);
 END;
-- 2. Run a customized archive procedure to purge the audit trail records:
  CALL MakeAuditSettings();
  LOOP (/* How long to loop*/)
    -- Invoke function for audit record archival
    CALL DoUnifiedAuditRecordArchival();
    CALL SetCleanUpAuditTrail();
    IF(/* Clean up is needed immediately */)
      DBMS AUDIT MGMT.CLEAN AUDIT TRAIL(
       AUDIT TRAIL TYPE => DBMS AUDIT MGMT.AUDIT TRAIL UNIFIED,
       USE LAST ARCH TIMESTAMP => TRUE,
       CONTAINER
                                => DBMS AUDIT MGMT.CONTAINER CURRENT );
    END IF
  END LOOP /*LOOP*/
END; /* PROCEDURE */
```

28.4 Audit Trail Management Data Dictionary Views

Oracle Database provides data dictionary views that list information about audit trail management settings.

Table 28-2 lists these views.

Table 28-2 Views That Display Information about Audit Trail Management Settings

View	Description	
DBA_AUDIT_MGMT_CLEAN_EVENTS	Displays the history of purge events of the traditional (that is, non-unified) audit trails. Periodically, as a user who has been granted the <code>AUDIT_ADMIN</code> role, you should delete the contents of this view so that it does not grow too large. For example:	
	DELETE FROM DBA_AUDIT_MGMT_CLEAN_EVENTS;	
	This view applies to read-write databases only. For read-only databases, a history of purge events is in the alert log.	
	For unified auditing, you can find a history of purged events by querying the UNIFIED_AUDIT_TRAIL data dictionary view, using the following criteria: OBJECT_NAME is DBMS_AUDIT_MGMT, OBJECT_SCHEMA is SYS, and SQL_TEXT is set to LIKE %DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL%.	
DBA_AUDIT_MGMT_CLEANUP_JOBS	Displays the currently configured audit trail purge jobs	
DBA_AUDIT_MGMT_CONFIG_PARAMS	Displays the currently configured audit trail properties that are used by the DBMS_AUDIT_MGMT PL/SQL package	
DBA_AUDIT_MGMT_LAST_ARCH_TS	Displays the last archive timestamps that have set for audit trail purges	

Related Topics

Oracle Database Reference



Part VII

Appendixes

Part VII contains a set of reference appendixes.

- Keeping Your Oracle Database Secure
 Oracle provides guidelines for keeping your database secure, such as advice on securing user accounts, privileges, roles, passwords, and data.
- Data Encryption and Integrity Parameters
 The sqlnet.ora file has data encryption and integrity parameters.
- Kerberos, TLS, and RADIUS Authentication Parameters
 The sqlnet.ora and the database initialization files provide Kerberos, RADIUS, or TLS authentication parameters.
- Integrating Authentication Devices Using RADIUS
 The RADIUS challenge-response user interface further enhances authentication in a RADIUS configuration.
- Oracle Database FIPS 140-2 Settings
 Oracle supports the Federal Information Processing Standard (FIPS) standard for 140-2.
- Managing Public Key Infrastructure (PKI) Elements
 You can use the orapki command line utility and sqlnet.ora parameters to manage public key infrastructure (PKI) elements.
- How the Unified Auditing Migration Affects Individual Audit Features
 Most of the pre-Oracle Database 12c release 1 (12.1) auditing features can be used before a unified auditing migration.

A

Keeping Your Oracle Database Secure

Oracle provides guidelines for keeping your database secure, such as advice on securing user accounts, privileges, roles, passwords, and data.

- About the Oracle Database Security Guidelines Information security, and privacy and protection of corporate assets and data are critical in any business.
- Downloading Security Patches and Contacting Oracle Regarding Vulnerabilities
 You should always apply security patches as soon as they are available. If problems arise, then you should contact Oracle regarding vulnerabilities.
- Guidelines for Securing User Accounts and Privileges
 Oracle provides guidelines to secure user accounts and privileges.
- Guidelines for Securing Roles
 Oracle provides guidelines for role management.
- Guidelines for Securing Passwords
 Oracle provides guidelines for securing passwords in a variety of situations.
- Guidelines for Securing Data
 Oracle provides guidelines for securing data on your system.
- Guidelines for Securing the ORACLE_LOADER Access Driver
 Oracle provides guidelines to secure the ORACLE LOADER access driver.
- Guidelines for Securing a Database Installation and Configuration
 Oracle provides guidelines to secure the database installation and configuration.
- Guidelines for Securing the Network
 Security for network communications is improved by using client, listener, and network
 guidelines to ensure thorough protection.
- Guideline for Securing External Procedures
 The ENFORCE_CREDENTIAL environment variable controls how an extproc process authenticates user credentials and callout functions.
- Guidelines for Auditing
 Oracle provides guidelines for auditing.
- Addressing the CONNECT Role Change
 The CONNECT role, introduced with Oracle Database version 7, added new and robust support for database roles.

A.1 About the Oracle Database Security Guidelines

Information security, and privacy and protection of corporate assets and data are critical in any business.

Oracle Database comprehensively addresses the need for information security by providing cutting-edge security features such as deep data protection, auditing, scalable security, secure hosting, and data exchange.

Oracle Database leads the industry in security. To maximize the security features offered by Oracle Database in any business environment, it is imperative that the database itself be well protected.

Security guidelines provide advice about how to configure Oracle Database to be secure by adhering to and recommending industry-standard and advisable security practices for operational database deployments. Many of the guidelines described in this section address common regulatory requirements such as those described in the Sarbanes-Oxley Act. For more information about how Oracle Database addresses regulatory compliance, protection of personally identifiable information, and internal threats, visit:

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

A.2 Downloading Security Patches and Contacting Oracle Regarding Vulnerabilities

You should always apply security patches as soon as they are available. If problems arise, then you should contact Oracle regarding vulnerabilities.

- Downloading Security Patches and Workaround Solutions
 Security patches apply to the operating system on which Oracle Database resides, Oracle Database itself, and all installed Oracle Database options and components.
- Contacting Oracle Security Regarding Vulnerabilities in Oracle Database
 You can contact Oracle Security regarding vulnerabilities in Oracle Database.

A.2.1 Downloading Security Patches and Workaround Solutions

Security patches apply to the operating system on which Oracle Database resides, Oracle Database itself, and all installed Oracle Database options and components.

- To download security patches and workaround solutions:
 - For security patches, periodically check the security site on Oracle Technology Network for details about security alerts released by Oracle at http://www.oracle.com/technetwork/topics/security/alerts-086861.html
 - Check the Oracle Worldwide Support Service site, My Oracle Support, for details about available and upcoming security-related patches at

https://support.oracle.com

A.2.2 Contacting Oracle Security Regarding Vulnerabilities in Oracle Database

You can contact Oracle Security regarding vulnerabilities in Oracle Database.

- Contact Oracle Security using either of the following methods:
 - If you are an Oracle customer or an Oracle partner, use My Oracle Support to submit a Service Request on any potential Oracle product security vulnerability.
 - Send an email to secalert_us@oracle.com with a complete description of the problem, including product version and platform, together with any scripts and examples. Oracle encourages those who want to contact Oracle Security to employ email encryption, using our encryption key.



A.3 Guidelines for Securing User Accounts and Privileges

Oracle provides guidelines to secure user accounts and privileges.

1. Lock and expire default (predefined) user accounts.

Oracle Database installs with several default database user accounts. Upon successful installation of the database, the Database Configuration Assistant automatically locks and expires most default database user accounts.

If you perform a manual (without using Database Configuration Assistant) installation of Oracle Database, then no default database users are locked upon successful installation of the database server. Or, if you have upgraded from a previous release of Oracle Database, you may have default accounts from earlier releases. Left open in their default states, these user accounts can be exploited, to gain unauthorized access to data or disrupt database operations.

You should *lock* and *expire* all default database user accounts. Oracle Database provides SQL statements to perform these operations. For example:

ALTER USER ANONYMOUS PASSWORD EXPIRE ACCOUNT LOCK;

Installing additional products and components after the initial installation also results in creating more default database accounts. Database Configuration Assistant automatically locks and expires all additionally created database user accounts. Unlock only those accounts that need to be accessed on a regular basis and assign a strong, meaningful password to each of these unlocked accounts. Oracle provides SQL and password management to perform these operations.

If any default database user account other than the ones left open is required for any reason, then a database administrator (DBA) must unlock and activate that account with a new, secure password.

If a default database user account, other than the ones left open, is required for any reason, then a database administrator (DBA) can unlock and activate that account with a new, secure password.

Securing Oracle Enterprise Manager Accounts

If you install Oracle Enterprise Manager, the SYSMAN and DBSNMP accounts are open, unless you configure Oracle Enterprise Manager for central administration. In this case, the SYSMAN account (if present) will be locked.

If you do not install Oracle Enterprise Manager, then only the SYS and SYSTEM accounts are open. Database Configuration Assistant locks and expires all other accounts (including SYSMAN and DBSNMP).

2. Discourage users from using the NOLOGGING clause in SQL statements.

In some SQL statements, the user has the option of specifying the Nologging clause, which indicates that the database operation is not logged in the online redo log file. Even though the user specifies the clause, a redo record is still written to the online redo log file. However, there is no data associated with this record. Because of this, using Nologging has the potential for malicious code to be entered can be accomplished without an audit trail.

3. Practice the principle of least privilege.

Oracle recommends the following guidelines:

a. Grant necessary privileges only.



Do not provide database users or roles more privileges than are necessary. (If possible, grant privileges to roles, not users.) In other words, the *principle of least privilege* is that users be given only those privileges that are actually required to efficiently perform their jobs.

To implement this principle, restrict the following as much as possible:

- The number of SYSTEM and OBJECT privileges granted to database users.
- The number of people who are allowed to make SYS-privileged connections to the database.
- The number of users who are granted the ANY privileges, such as the DROP ANY TABLE privilege. For example, there is generally no need to grant CREATE ANY TABLE privileges to a non-DBA-privileged user.
- The number of users who are allowed to perform actions that create, modify, or drop database objects, such as the TRUNCATE TABLE, DELETE TABLE, DROP TABLE statements, and so on.
- b. Limit granting the CREATE ANY EDITION and DROP ANY EDITION privileges.

To maintain additional versions of objects, editions can increase resource and disk space consumption in the database. Only grant the CREATE ANY EDITION and DROP ANY EDITION privileges to trusted users who are responsible for performing upgrades.

 Re-evaluate the SELECT object privilege and SELECT ANY TABLE system privileges that you have granted to users.

If you want to restrict users to only being able to query tables, views, materialized views, and synonyms, then grant users the READ object privilege, or for trusted users only, the READ ANY TABLE system privilege. If in addition to performing query operations, you want users to be able to lock tables in exclusive mode or perform SELECT ... FOR UPDATE statements, then grant the user the SELECT object privilege or, for trusted users only, the SELECT ANY TABLE system privilege.

d. Restrict the CREATE ANY JOB, BECOME USER, EXP_FULL_DATABASE, and IMP_FULL_DATABASE privileges. Also restrict grants of the CREATE DIRECTORY and CREATE ANY DIRECTORY privileges.

These are powerful security-related privileges. Only grant these privileges to users who need them.

 Restrict the BECOME USER privilege to users of Oracle Data Pump, and the DBMS_WORKLOAD_CAPTURE and DBMS_WORKLOAD_REPLAY packages.

The BECOME USER privilege is used only for the following subsystems:

- Oracle Data Pump Import utilities impdp and imp, to assume the identity of another
 user to perform operations that cannot be directly performed by a third party (for
 example, loading objects such as object privilege grants). In an Oracle Database
 Vault environment, Database Vault provides several levels of required
 authorization that affect grants of BECOME USER.
- DBMS_WORKLOAD_CAPTURE and DBMS_WORKLOAD_REPLAY PL/SQL packages, as a required privilege to be granted to users who must use these packages.

If you use the AUTHID CURRENT_USER clause when invoking one of these subsystems (for example, in static references in PL/SQL code), then ensure that the CURRENT_USER is granted the BECOME USER privilege, either by a direct grant or through a role.

f. Restrict library-related privileges to trusted users only.



The CREATE LIBRARY, CREATE ANY LIBRARY, ALTER ANY LIBRARY, and EXECUTE ANY LIBRARY privileges, and grants of EXECUTE ON <code>library_name</code> convey a great deal of power to users. If you plan to create PL/SQL interfaces to libraries, only grant the EXECUTE privilege to the PL/SQL interface. Do not grant EXECUTE on the underlying library. You must have the EXECUTE privilege on a library to create the PL/SQL interface to it. However, users have this privilege implicitly on libraries that they create in their own schemas. Explicit grants of EXECUTE ON <code>library_name</code> are rarely required. Only make an explicit grant of these privileges to trusted users, and never to the <code>PUBLIC</code> role.

g. Restrict synonym-related privileges to trusted users only.

The CREATE PUBLIC SYNONYM and DROP PUBLIC SYNONYM system privileges convey a great deal of power to these users. Do not grant these privileges to users, unless they are trusted.

Do not allow non-administrative users access to objects owned by the SYS schema.

Do not allow users to alter table rows or schema objects in the SYS schema, because doing so can compromise data integrity. Limit the use of statements such as DROP TABLE, TRUNCATE TABLE, DELETE, INSERT, or similar object-modification statements on SYS objects only to highly privileged administrative users.

i. Restrict permissions on run-time facilities.

Many Oracle Database products use run-time facilities, such as Oracle Java Virtual Machine (OJVM). Do not assign all permissions to a database run-time facility. Instead, grant specific permissions to the explicit document the root file paths for facilities that might run files and packages outside the database.

Here is an example of a vulnerable run-time call, which individual files are specified:

```
call dbms_java.grant_permission('wsmith', 'SYS:java.io.FilePermission','<<ALL
FILES>>','read');
```

Here is an example of a better (more secure) run-time call, which specifies a directory path instead:

```
call dbms_java.grant_permission('wsmith', 'SYS:java.io.FilePermission','<<actual
directory path>>','read');
```

4. Revoke access to the following:

- The SYS.USER HISTORY\$ table from all users except SYS and DBA accounts
- The RESOURCE role from typical application accounts
- The CONNECT role from typical application accounts
- The DBA role from users who do not need this role

5. Grant privileges only to roles.

Granting privileges to roles and not individual users makes the management and tracking of privileges much easier.

Limit the proxy account (for proxy authorization) privileges to CREATE SESSION only.

7. Use secure application roles to protect roles that are enabled by application code.

Secure application roles allow you to define a set of conditions, within a PL/SQL package, that determine whether or not a user can log on to an application. Users do not need to use a password with secure application roles.

Another approach to protecting roles from being enabled or disabled in an application is the use of role passwords. This approach prevents a user from directly accessing the database in SQL (rather than the application) to enable the privileges associated with the role. However, Oracle recommends that you use secure application roles instead, to avoid having to manage another set of passwords.

- 8. Create privilege captures to find excessively granted privileges. Privilege analysis captures the privileges that users and applications use, and then presents these in a format for easy analysis. From there, you can revoke unnecessary privileges if you want.
- 9. Monitor the granting of the following privileges only to users and roles who need these privileges.

By default, Oracle Database audits the following privileges:

- ALTER SYSTEM
- AUDIT SYSTEM
- CREATE EXTERNAL JOB

Oracle recommends that you also audit the following privileges:

- ALL PRIVILEGES (which includes privileges such as BECOME USER, CREATE LIBRARY, and CREATE PROCEDURE)
- DBMS_BACKUP_RESTORE package
- EXECUTE to DBMS_SYS_SQL
- SELECT ANY TABLE
- SELECT ON PERFSTAT.STATS\$SQLTEXT
- SELECT ON PERFSTAT.STATS\$SQL SUMMARY
- SELECT on SYS.SOURCE\$
- Privileges that have the WITH ADMIN clause
- Privileges that have the WITH GRANT clause
- Privileges that have the CREATE keyword
- Use the following data dictionary views to find information about user access to the database.
 - DBA *
 - DBA ROLES
 - DBA SYS PRIVS
 - DBA ROLE PRIVS
 - DBA TAB PRIVS
 - DBA AUDIT TRAIL (if standard auditing is enabled)
 - DBA FGA AUDIT TRAIL (if fine-grained auditing is enabled)

Related Topics

- Oracle Database Vault Administrator's Guide
- Performing Privilege Analysis to Identify Privilege Use
 Privilege analysis dynamically analyzes the privileges and roles that users use and do not use.



A.4 Guidelines for Securing Roles

Oracle provides guidelines for role management.

1. Grant a role to users only if they need all privileges of the role.

Roles (groups of privileges) are useful for quickly and easily granting permissions to users. Although you can use Oracle-defined roles, you have more control and continuity if you create your own roles containing only the privileges pertaining to your requirements. Oracle may change or remove the privileges in an Oracle Database-defined role, as it has with the CONNECT role, which now has only the CREATE SESSION privilege. Formerly, this role had eight other privileges.

Ensure that the roles you define contain only the privileges that reflect job responsibility. If your application users do not need all the privileges encompassed by an existing role, then apply a different set of roles that supply just the correct privileges. Alternatively, create and assign a more restricted role.

For example, it is imperative to strictly limit the privileges of user SCOTT, because this is a well known account that may be vulnerable to intruders. Because the CREATE DBLINK privilege allows access from one database to another, drop its privilege for SCOTT. Then, drop the entire role for the user, because privileges acquired by means of a role cannot be dropped individually. Re-create your own role with only the privileges needed, and grant that new role to that user. Similarly, for better security, drop the CREATE DBLINK privilege from all users who do not require it.

2. Do not grant user roles to application developers.

Roles are not meant to be used by application developers, because the privileges to access schema objects within stored programmatic constructs need to be granted directly. Remember that roles are not enabled within stored procedures except for invoker's right procedures. See How Roles Work in PL/SQL Blocks for information about this topic.

3. Create and assign roles specific to each Oracle Database installation.

This principle enables the organization to retain detailed control of its roles and privileges. This also avoids the necessity to adjust if Oracle Database changes or removes Oracle Database-defined roles, as it has with CONNECT, which now has only the CREATE SESSION privilege. Formerly, it also had eight other privileges.

For enterprise users, create global roles.

Global roles are managed by an enterprise directory service, such as Oracle Internet Directory. See the following sections for more information about global roles:

- Global User Authentication and Authorization
- Authorizing a Global Role by an Enterprise Directory Service
- Oracle Database Enterprise User Security Administrator's Guide

A.5 Guidelines for Securing Passwords

Oracle provides guidelines for securing passwords in a variety of situations.

When you create a user account, Oracle Database assigns a default password policy for that user. The password policy defines rules for how the password should be created, such as a minimum number of characters, when it expires, and so on. You can strengthen passwords by using password policies. See also Configuring Password Protection for additional ways to protect passwords.



Follow these guidelines to further strengthen passwords:

1. Choose passwords carefully.

Minimum Requirements for Passwords describes the minimum requirements for passwords. Follow these additional guidelines when you create or change passwords:

- Make the password have a length of between 12 and 30 bytes, and include both alphabetic characters and digits in the password.
- Have the password contain at least one digit, one upper-case character, and one lower-case character.
- Use mixed case characters and special characters in the password. (See Ensuring Against Password Security Threats by Using the 12C Password Version for more information.)
- You can include multibyte characters in the password but not in the password of any common user or role.
- Use the database character set for the password's characters, which can include the underscore (_), dollar (\$), and number sign (#) characters.
- You must enclose the following passwords in double-quotation marks:
 - Passwords containing multibyte characters
 - Passwords starting with numbers or special characters and containing alphabetic characters. For example:

```
"123abc"
"#abc"
"123dc$"
```

 Passwords containing any character other than alphabetic characters, numbers, and special characters. For example:

```
"abc>"
"abc@",
```

- You do not need to specify the following passwords in double-quotation marks.
 - Passwords starting with an alphabetic character (a–z, A–Z) and containing numbers (0–9) or special characters (\$, #, _). For example:

```
abc123
ab23a
ab$#_
```

- Passwords containing only numbers
- Passwords containing only alphabetic characters (a–z, A–Z)
- Do not include double-quotation marks within the password.
- Do not use an actual word for the entire password.
- To create a longer, more complex password from a shorter, easier to remember password, create the password from the first letters of the words of an easy-toremember sentence.

For example, "I usually work until 6:00 almost every day of the week" can be Iuwu6aedotw.

3. Ensure that the password is sufficiently complex.

Oracle Database provides a password complexity verification routine, the PL/SQL script utlpwdmg.sql, that you can run to check whether or not passwords are sufficiently complex. Ideally, edit the utlpwdmg.sql script to provide stronger password protections. See also About Password Complexity Verification for a sample routine that you can use to check passwords.

4. In a non-multitenant environment, or in a PDB, if you want to use multibyte characters in the password, then ensure that the database character set is configured as a multibye character set so that the authentication will work properly. Be aware that because multibyte characters consume more bytes that single-byte characters, they tend to provide less entropy per byte. Because the maximum length of the password is currently limited to 30 bytes, to help increase the amount of entropy in a password, Oracle recommends that you also include a number of single-byte characters in the password, even when multibyte characters are being used.

The passwords of common users and common roles cannot contain multibyte passwords.

5. Associate a password complexity function with the user profile or the default profile.

The PASSWORD_VERIFY_FUNCTION clause of the CREATE PROFILE and ALTER PROFILE statements associates a password complexity function with a user profile or the default profile. Password complexity functions ensure that users create strong passwords using guidelines that are specific to your site. Having a password complexity function also requires a user changing his or her own password (without the ALTER USER system privilege) to provide both the old and new passwords. You can create your own password complexity functions or use the password complexity functions that Oracle Database provides.

See Managing the Complexity of Passwords for more information.

Change default user passwords.

Oracle Database installs with a set of predefined, default user accounts. Security is most easily broken when a default database user account still has a default password *even after installation*. This is particularly true for the user account SCOTT, which is a well known account that may be vulnerable to intruders. In Oracle Database, default accounts are installed locked with the passwords expired, but if you have upgraded from a previous release, you may still have accounts that use default passwords.

To find user accounts that have default passwords, query the DBA_USERS_WITH_DEFPWD data dictionary view. See Finding User Accounts That Have Default Passwords for more information.

Change default passwords of administrative users.

You can use the same or different passwords for the SYS, SYSTEM, SYSMAN, and DBSNMP administrative accounts. Oracle recommends that you use different passwords for each. In any Oracle environment (production or test), assign strong, secure, and distinct passwords to these administrative accounts. If you use Database Configuration Assistant to create a new database, then it requires you to enter passwords for the SYS and SYSTEM accounts, disallowing the default passwords CHANGE ON INSTALL and MANAGER.

Similarly, for production environments, do not use default passwords for administrative accounts, including SYSMAN and DBSNMP.

8. Enforce password management.

Apply basic password management rules (such as password length, history, complexity, and so forth) to all user passwords. Oracle Database has password policies enabled for the default profile. Guideline 1 in this section lists these password policies.

You can find information about user accounts by querying the DBA_USERS view. The PASSWORD column of the DBA_USERS view indicates whether the password is global, external, or null. The DBA_USERS view provides useful information such as the user account status, whether the account is locked, and password versions.

Oracle also recommends, if possible, using Oracle strong authentication with network authentication services (such as Kerberos), token cards, smart cards, or X.509 certificates. These services provide strong authentication of users, and provide protection against unauthorized access to Oracle Database.

9. Do not store user passwords in clear text in Oracle tables.

For better security, do not store passwords in clear text (that is, human readable) in Oracle tables. You can correct this problem by using a secure external password store to encrypt the password within an Oracle wallet. (An Oracle wallet is a secure software container that stores authentication and signing credentials.) See Managing the Secure External Password Store for Password Credentials for information.

When you create or modify a password for a user account, Oracle Database automatically creates a cryptographic hash or digest of the password. If you query the DBA_USERS view to find information about a user account, the data in the PASSWORD column indicates if the user password is global, external, or null. The DBA_USERS view also has a column called PASSWORD_VERSIONS, which lists the types of cryptographic hash that exist for the user's password (11G or 12C). See *Oracle Database Reference* for information in the section on DBA_USERS about the cryptographic algorithm that these abbreviations correspond to.

10. Disable the HTTP verifier if the user is not going to be using either XDB authentication or HTTP Digest authentication.

The HTTP verifier is used only for XDB authentication and HTTP Digest authentication. If a user is not going to use XDB authentication or HTTP Digest authentication, then you can safely remove the HTTP verifier from the user's list of verifiers. To remove a user's HTTP verifier, execute the following statement:

ALTER USER username DIGEST DISABLE;

A.6 Guidelines for Securing Data

Oracle provides guidelines for securing data on your system.

1. Restrict operating system access.

Follow these guidelines:

- · Limit the number of operating system users.
- Limit the privileges of the operating system accounts (administrative, root-privileged, or database administrative) on the Oracle Database host computer to the least privileges required for a user to perform necessary tasks.
- Restrict the ability to modify the default file and directory permissions for the Oracle Database home (installation) directory or its contents. Even privileged operating system users and the Oracle owner should not modify these permissions, unless instructed otherwise by Oracle.
- Restrict symbolic links. Ensure that when you provide a path or file to the database, neither the file nor any part of the path is modifiable by an untrusted user. The file and all components of the path should be owned by the database administrator or trusted account, such as *root*.

This recommendation applies to all types of log files, trace files, external tables, BFILE data types, and so on.

Encrypt sensitive data and all backup media that contains database files.

According to common regulatory compliance requirements, you must encrypt sensitive data such as credit card numbers and passwords. When you delete sensitive data from the database, encrypted data does not linger in data blocks, operating system files, or sectors on disk.

In most cases, you may want to use Transparent Data Encryption to encrypt your sensitive data. See *Oracle Database Advanced Security Guide* for more information. See also Security Problems That Encryption Does Not Solve for when you should not encrypt data.

For Oracle Automatic Storage Management (Oracle ASM) environments on Linux and UNIX systems, use Oracle ASM File Access Control to restrict access to the Oracle ASM disk groups.

If you use different operating system users and groups for Oracle Database installations, then you can configure Oracle ASM File Access Control to restrict the access to files in Oracle ASM disk groups to only authorized users. For example, a database administrator would only be able to access the data files for the databases he or she manages. This administrator would not be able to see or overwrite the data files belonging (or used by) other databases.

For more information about managing Oracle ASM File Access Control for disk groups, see *Oracle Automatic Storage Management Administrator's Guide*. For information about the various privileges required for multiple software owners, see also *Oracle Automatic Storage Management Administrator's Guide*.

A.7 Guidelines for Securing the ORACLE_LOADER Access Driver

Oracle provides guidelines to secure the ORACLE LOADER access driver.

- 1. Create a separate operating system directory to store the access driver preprocessors. You (or the operating system manager) may need to create multiple directories if different Oracle Database users will run different preprocessors. If you want to prevent one set of users from using one preprocessor while allowing those users access to another preprocessor, then place the preprocessors in separate directories. If all the users need equal access, then you can place the preprocessors together in one directory. After you create these operating system directories, in SQL*Plus, you can create a directory object for each directory.
- 2. Grant the operating system user ORACLE the correct operating system privileges to run the access driver preprocessor. In addition, protect the preprocessor program from WRITE access by operating system users other than the user responsible for managing the preprocessor program.
- 3. Grant the EXECUTE privilege to each user who will run the preprocessor program in the directory object. Do not grant this user the WRITE privilege on the directory object. Never grant users both the EXECUTE and WRITE privilege for directory objects.
- 4. Grant the WRITE privilege sparingly to anyone who will manage directory objects that contain preprocessors. This prevents database users from accidentally or maliciously overwriting the preprocessor program.
- 5. Create a separate operating system directory and directory object for any data files that are required for external tables. Ensure that these are separate from the directory and directory object used by the access directory preprocessor.



Work with the operating system manager to ensure that only the appropriate operating system users have access to this directory. Grant the <code>ORACLE</code> operating system user <code>READ</code> access to any directory that has a directory object with <code>READ</code> privileges granted to database users. Similarly, grant the <code>ORACLE</code> operating system user <code>WRITE</code> access to any directory that has the <code>WRITE</code> privilege granted to database users.

- 6. Create a separate operating system directory and directory object for any files that the access driver generates. This includes log files, bad files, and discarded files. You and the operating system manager must ensure that this directory and directory object have the proper protections, similar to those described in Guideline 5. The database user may need to access these files when resolving problems in data files, so you and the operating system manager must determine a way for this user to read those files.
- 7. Grant the CREATE ANY DIRECTORY and DROP ANY DIRECTORY privileges sparingly. Users who have these privileges and users who have been granted the DBA role have full access to all directory objects.
- **8. Consider auditing the DROP ANY DIRECTORY privilege.** See Auditing System Privileges for more information about auditing privileges.
- Consider auditing the directory object. See Auditing Object Actions for more information.



Oracle Database Utilities for more information about the ORACLE_DATAPUMP access driver

A.8 Guidelines for Securing a Database Installation and Configuration

Oracle provides guidelines to secure the database installation and configuration.

Changes were made to the default configuration of Oracle Database to make it more secure. The recommendations in this section augment the new, secure default configuration.

1. Before you begin an Oracle Database installation on UNIX systems, ensure that the umask value is 022 for the Oracle owner account.

See Oracle Database Administrator's Reference for Linux and UNIX-Based Operating Systems for more information about managing Oracle Database on Linux and UNIX systems.

2. Install only what is required.

Options and Products: The Oracle Database CD pack contains products and options in addition to the database. Install additional products and options only as necessary. Use the Custom Installation feature to avoid installing unnecessary products, or perform a typical installation, and then deinstall options and products that are not required. There is no need to maintain additional products and options if they are not being used. They can always be properly installed, as required.

Sample Schemas: Oracle Database provides sample schemas to provide a common platform for examples. If your database will be used in a production environment, then do not install the sample schema. If you have installed the sample schema on a test



database, then before going to production, remove or relock the sample schema accounts. See *Oracle Database Sample Schemas* for more information about the sample schemas.

During installation, when you are prompted for a password, create a secure password.

Follow Guidelines 1, 6, and 7 in Guidelines for Securing Passwords.

Immediately after installation, lock and expire default user accounts.

See Guideline 1 in Guidelines for Securing User Accounts and Privileges.

A.9 Guidelines for Securing the Network

Security for network communications is improved by using client, listener, and network guidelines to ensure thorough protection.

- Client Connection Security
 Authenticating clients stringently, configuring encryption for the connection, and using strong authentication strengthens client connections.
- Network Connection Security
 Protecting the network and its traffic from inappropriate access or modification is the
 essence of network security.
- Transport Layer Security Connection Security
 Oracle provides guidelines for securing Transport Layer Security (TLS).

A.9.1 Client Connection Security

Authenticating clients stringently, configuring encryption for the connection, and using strong authentication strengthens client connections.

Because authenticating client computers is problematic, typically, user authentication is performed instead. This approach avoids client system issues that include falsified IP addresses, hacked operating systems or applications, and falsified or stolen client system identities.

Nevertheless, the following guidelines improve the security of client connections:

Enforce access controls effectively and authenticate clients stringently.

By default, Oracle allows operating system-authenticated logins only over secure connections, which precludes using Oracle Net and a shared server configuration. This default restriction prevents a user over a network connection.

Setting the initialization parameter REMOTE_OS_AUTHENT to TRUE forces the database to accept the client operating system user name received over an unsecure connection and use it for account access. Because clients, such as PCs, are not trusted to perform operating system authentication properly, it is poor security practice to use this feature.

The default setting, REMOTE_OS_AUTHENT = FALSE, creates a more secure configuration that enforces proper, server-based authentication of clients connecting to an Oracle database. Be aware that the REMOTE_OS_AUTHENT was deprecated in Oracle Database Release 11*g* (11.1) and is retained only for backward compatibility.

You should not alter the default setting of the REMOTE_OS_AUTHENT initialization parameter, which is FALSE.

Setting this parameter to FALSE does not mean that users cannot connect remotely. It means that the database will not trust that the client has already authenticated, and will therefore apply its standard authentication processes.

Be aware that the REMOTE_OS_AUTHENT parameter was deprecated in Oracle Database 11*g* Release 1 (11.1), and is retained only for backward compatibility.

2. Configure the connection to use encryption.

Oracle native network encryption makes eavesdropping difficult.

3. Set up strong authentication.

See Configuring Kerberos Authentication , for more information about using Kerberos and public key infrastructure (PKI).

4. In an Oracle Data Guard environment, set the ADG_ACCOUNT_INFO_TRACKING initialization parameter.

The ADG_ACCOUNT_INFO_TRACKING parameter controls login attempts on Oracle Active Data Guard standby databases. It provides more security against login attacks across an Oracle Database production environment and all Active Data Guard standby databases. Use one of the following settings:

- LOCAL (default) enforces the existing behavior, which maintains a local copy of user
 account information in the standby database's in-memory view. This setting only tracks
 login failures locally on a per-database basis. It denies the login when the maximum of
 failed logins is reached.
- GLOBAL increases the security of logins by maintaining a single global copy of user
 account information across all Data Guard primary and standby databases. Login
 failures across all databases in the Data Guard environment count toward the
 maximum count. When this count is reached, then logins anywhere are denied access.

To learn more about the ADG_ACCOUNT_INFO_TRACKING parameter, see *Oracle Database Reference*

A.9.2 Network Connection Security

Protecting the network and its traffic from inappropriate access or modification is the essence of network security.

You should consider all paths the data travels, and assess the threats on each path and node. Then, take steps to lessen or eliminate those threats and the consequences of a security breach. In addition, monitor and audit to detect either increased threat levels or penetration attempts.

To manage network connections, you can use Oracle Net Manager. For more information about Net Manager, see *Oracle Database Net Services Administrator's Guide*.

The following practices improve network security:

1. Use Transport Layer Security (TLS) when administering the listener.

TLS can protect the messages sent and received by you or by applications and servers, supporting secure authentication, authorization, and messaging through certificates and, if necessary, encryption.

- 2. Prevent online administration by requiring the administrator to have the write privilege on the listener password and on the listener.ora file on the server.
 - a. Add or alter this line in the listener.ora file:



ADMIN RESTRICTIONS LISTENER=ON

- **b.** Use RELOAD to reload the configuration.
- **c.** Use TLS when administering the listener by making the TCPS protocol the first entry in the address list, as follows:

```
LISTENER=
(DESCRIPTION=
(ADDRESS_LIST=
(ADDRESS=
(PROTOCOL=tcps)
(HOST = sales.us.example.com)
(PORT = 8281)))
```

To administer the listener remotely, you define the listener in the listener.ora file on the client computer. For example, to access listener USER281 remotely, use the following configuration:

```
user281 =
  (DESCRIPTION =
    (ADDRESS =
        (PROTOCOL = tcps)
        (HOST = sales.us.example.com)
        (PORT = 8281))
    )
}
```

3. Do not set the listener password.

Ensure that the password has not been set in the <code>listener.ora</code> file. The local operating system authentication will secure the listener administration. The remote listener administration is disabled when the password has not been set. This prevents brute force attacks of the listener password.

The listener password has been deprecated in this release. It will not be supported in the next release of Oracle Database.

4. When a host computer has multiple IP addresses associated with multiple network interface controller (NIC) cards, configure the listener to the specific IP address.

This allows the listener to listen on all the IP addresses. You can restrict the listener to listen on a specific IP address. Oracle recommends that you specify the specific IP addresses on these types of computers, rather than allowing the listener to listen on all IP addresses. Restricting the listener to specific IP addresses helps to prevent an intruder from stealing a TCP end point from under the listener process.

5. Restrict the privileges of the listener, so that it cannot read or write files in the database or the Oracle server address space.

This restriction prevents external procedure agents spawned by the listener (or procedures executed by an agent) from inheriting the ability to perform read or write operations. The owner of this separate listener process should not be the owner that installed Oracle Database or executes the Oracle Database instance (such as ORACLE, the default owner).

For more information about configuring external procedures in the listener, see *Oracle Database Net Services Administrator's Guide*.

Use encryption to secure the data in flight.

Strong authentication will help to protect network data encryption.

7. Use a firewall.

Appropriately placed and configured firewalls can prevent outside access to your databases.

- Keep the database server behind a firewall. Oracle Database network infrastructure,
 Oracle Net Services (formerly known as SQL*Net), provides support for a variety of
 firewalls from various vendors. Supported proxy-enabled firewalls include Gauntlet
 from Network Associates and Raptor from Axent. Supported packet-filtering firewalls
 include PIX Firewall from Cisco, and supported stateful inspection firewalls (more
 sophisticated packet-filtered firewalls) include Firewall-1 from CheckPoint.
- Ensure that the firewall is placed outside the network to be protected.
- Configure the firewall to accept only those protocols, applications, or client/server sources that you know are safe.
- Use a product such as Net8 and Oracle Connection Manager to manage multiplex multiple client network sessions through a single network connection to the database.
 It can filter on source, destination, and host name. This product enables you to ensure that connections are accepted only from physically secure terminals or from application Web servers with known IP addresses. (Filtering on IP address alone is not enough for authentication, because it can be falsified.)

8. Prevent unauthorized administration of the Oracle listener.

For more information about the listener, see *Oracle Database Net Services Administrator's Guide*.

9. Check network IP addresses.

Use the Oracle Net *valid node checking* security feature to allow or deny access to Oracle server processes from network clients with specified IP addresses. To use this feature, set the following sqlnet.ora configuration file parameters:

```
tcp.validnode_checking = YES
tcp.excluded_nodes = {list of IP addresses}
tcp.invited nodes = {list of IP addresses}
```

The tcp.validnode_checking parameter enables the feature. The tcp.excluded_nodes and tcp.invited_nodes parameters deny and enable specific client IP addresses from making connections to the Oracle listener. This helps to prevent potential Denial of Service attacks.

10. Encrypt network traffic.

If possible, use Oracle native network data encryption to encrypt network traffic among clients, databases, and application servers.

Secure the host operating system (the system on which Oracle Database is installed).

Secure the host operating system by disabling all unnecessary operating system services. Both UNIX and Windows provide a variety of operating system services, most of which are not necessary for typical deployments. These services include FTP, TFTP, TELNET, and so forth. Be sure to close both the UDP and TCP ports for each service that is being disabled. Disabling one type of port and not the other does not make the operating system more secure.

12. Configure database link communication protocol.

To specify the protocols over which the database link communication takes place, set the OUTBOUND DBLINK PROTOCOLS initialization parameter to one of the following settings:

- ALL (default) enables all net protocols to be used for the database links.
- comma-separated_list_of_protocols can be set TPC, TCPS, or IPC. For example, for a single protocol:

ALTER SYSTEM SET OUTBOUND DBLINK PROTOCOLS=TCPS;

For multiple protocols:

ALTER SYSTEM SET OUTBOUND DBLINK PROTOCOLS=TCP, TCPS, IPC;

NONE disables any database link communication.

13. If necessary, disable LDAP lookup for global database links.

Set the ALLOW_GLOBAL_DBLINKS initialization parameter to enable or disable LDAP lookup for global database links. Settings are as follows:

- ON enables LDAP lookup for global database links.
- OFF (default) disables LDAP lookup for global database links.

Related Topics

- Oracle Database Net Services Administrator's Guide
- Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.
- Oracle Database Net Services Administrator's Guide
- Introduction to Strong Authentication
 Strong authentication supports tools such as Transport Layer Security (TLS) to verify the identities of users who log in to the database.
- Oracle Database Net Services Administrator's Guide
- Configuring Oracle Database Native Network Encryption and Data Integrity
 You can configure native Oracle Net Services data encryption and data integrity for both
 servers and clients.

A.9.3 Transport Layer Security Connection Security

Oracle provides guidelines for securing Transport Layer Security (TLS).

Transport Layer Security (TLS) is the Internet standard protocol for secure communication, providing mechanisms for data integrity and data encryption. These mechanisms can protect the messages sent and received by you or by applications and servers, supporting secure authentication, authorization, and messaging through certificates and, if necessary, encryption. Good security practices maximize protection and minimize gaps or disclosures that threaten security.

 Ensure that configuration files (for example, for clients and listeners) use the correct port for TLS, which is the port configured upon installation.

You can run HTTPS on any port, but the standards specify port 443, where any HTTPS-compliant browser looks by default. The port can also be specified in the URL, for example:

https://secure.example.com:4445/

If a firewall is in use, then it too must use the same ports for secure (TLS) communication.

Ensure that TCPS is specified as the PROTOCOL in the ADDRESS parameter in the tnsnames.ora file (typically on the client or in the LDAP directory). An identical specification must appear in the listener.ora file (typically in the \$ORACLE HOME/network/admin directory).

Ensure that the TLS mode is consistent for both ends of every communication. For example, the database (on one side) and the user or application (on the other) must have the same TLS mode.

The mode can specify either client or server authentication (one-way), both client and server authentication (two-way), or no authentication.

- 4. Ensure that the server supports the client cipher suites and the certificate key algorithm in use.
- Enable DN matching for both the server and client, to prevent the server from falsifying its identity to the client during connections.

This setting ensures that the server identity is correct by matching its global database name against the DN from the server certificate.

You can enable DN matching in the tnsnames.ora file. For example:

```
set:SSL_SERVER_CERT_DN="cn=finance,cn=OracleContext,c=us,o=example"
```

Otherwise, a client application would not check the server certificate, which could allow the server to falsify its identity.

6. Do not remove the encryption from your RSA private key inside your server.key file, which requires that you enter your pass phrase to read and parse this file.



A server without TLS does not require a pass phrase.

If you decide your server is secure enough, you could remove the encryption from the RSA private key while preserving the original file. This enables system boot scripts to start the database server, because no pass phrase is needed. Ideally, restrict permissions to the root user only, and have the Web server start as root, but then log on as another user. Otherwise, anyone who gets this key can impersonate you on the Internet, or decrypt the data that was sent to the server.

Related Topics

- Configuring Transport Layer Security Authentication
 You can configure Oracle Database to use Transport Layer Security authentication.
- Oracle Database Net Services Reference

A.10 Guideline for Securing External Procedures

The <code>ENFORCE_CREDENTIAL</code> environment variable controls how an <code>extproc</code> process authenticates user credentials and callout functions.

You can specify this variable in the <code>extproc.ora</code> file. Before modifying this variable, review your site's security requirements for the handling of external libraries. For maximum security, set the <code>ENFORCE CREDENTIAL</code> variable to <code>TRUE</code>. The default setting is <code>FALSE</code>.



See Also

Securing External Procedures

A.11 Guidelines for Auditing

Oracle provides guidelines for auditing.

- Manageability of Audited Information
 Although auditing is relatively inexpensive, limit the number of audited events as much as possible.
- Audits of Typical Database Activity
 Oracle provides guidelines for when you must gather historical information about particular database activities.
- Audits of Suspicious Database Activity
 Oracle provides guidelines for when you audit to monitor suspicious database activity.
- Audits of Sensitive Data
 Oracle recommends that you include the ACTIONS ALL clause when you create unified audit policies on sensitive objects.
- Recommended Audit Settings
 Oracle provides predefined policies that contain recommended audit settings that apply to most sites.
- Best Practices for Querying the UNIFIED_AUDIT_TRAIL Data Dictionary View
 To get the best results from querying the UNIFIED_AUDIT_TRAIL data dictionary view, you
 should follow these guidelines.

A.11.1 Manageability of Audited Information

Although auditing is relatively inexpensive, limit the number of audited events as much as possible.

This minimizes the performance impact on the execution of audited statements and the size of the audit trail, making it easier to analyze and understand.

Follow these guidelines when devising an auditing strategy:

1. Evaluate your reason for auditing.

After you have a clear understanding of the reasons for auditing, you can devise an appropriate auditing strategy and avoid unnecessary auditing.

For example, suppose you are auditing to investigate suspicious database activity. This information by itself is not specific enough. What types of suspicious database activity do you suspect or have you noticed? A more focused auditing strategy might be to audit unauthorized deletions from arbitrary tables in the database. This purpose narrows the type of action being audited and the type of object being affected by the suspicious activity.

2. Audit knowledgeably.

Audit the minimum number of statements, users, or objects required to get the targeted information. This prevents unnecessary audit information from cluttering the meaningful information and using valuable space in the SYSTEM tablespace. Balance your need to gather sufficient security information with your ability to store and process it.



For example, if you are auditing to gather information about database activity, then determine exactly what types of activities you want to track, audit only the activities of interest, and audit only for the amount of time necessary to gather the information that you want. As another example, do not audit *objects* if you are only interested in logical I/O information for each session.

3. Before you implement an auditing strategy, consult your legal department.

You should have the legal department of your organization review your audit strategy. Because your auditing will monitor other users in your organization, you must ensure that you are correctly following the compliance and corporate policy of your site.

A.11.2 Audits of Typical Database Activity

Oracle provides guidelines for when you must gather historical information about particular database activities.

1. Audit only pertinent actions.

At a minimum, audit user access, the use of system privileges, and changes to the database schema structure. To avoid cluttering meaningful information with useless audit records and reduce the amount of audit trail administration, only audit the targeted database activities. Remember also that auditing too much can affect database performance.

For example, auditing changes to all tables in a database produces far too many audit trail records and can slow down database performance. However, auditing changes to critical tables, such as salaries in a Human Resources table, is useful.

You can audit specific actions by using fine-grained auditing, which is described in Auditing Specific Activities with Fine-Grained Auditing.

2. Archive audit records and purge the audit trail.

After you collect the required information, archive the audit records of interest and then purge the audit trail of this information. See the following sections:

- Archiving the Audit Trail
- Purging Audit Trail Records

3. Remember your company's privacy considerations.

Privacy regulations often lead to additional business privacy policies. Most privacy laws require businesses to monitor access to personally identifiable information (PII), and monitoring is implemented by auditing. A business-level privacy policy should address all relevant aspects of data access and user accountability, including technical, legal, and company policy concerns.

4. Check the Oracle Database log files for additional audit information.

The log files generated by Oracle Database contain useful information that you can use when auditing a database. For example, an Oracle database creates an alert file to record STARTUP and SHUTDOWN operations, and structural changes such as adding data files to the database.

For example, if you want to audit committed or rolled back transactions, you can use the redo log files.

To reduce the size of the audit trail and recursive SQL statements, audit only toplevel statements.

If you have concerns that the unified audit policy that you create will generate a very large number of records, then include the <code>ONLY TOPLEVEL</code> clause in the <code>CREATE AUDIT POLICY</code>

statement. For example, an audit of the <code>DBMS_STATS.GATHER_DATABASE_STATS</code> SQL statement can generate thousands of audit records. You can audit top-level statements from all users, including user <code>SYS</code>.

A.11.3 Audits of Suspicious Database Activity

Oracle provides guidelines for when you audit to monitor suspicious database activity.

1. First audit generally, and then specifically.

When you start to audit for suspicious database activity, often not much information is available to target specific users or schema objects. Therefore, audit generally first, that is, by using the unified audit policies. Configuring Audit Policies explains how you can audit SQL statements, schema objects, privileges, and so on.

After you have recorded and analyzed the preliminary audit information, alter your audit policies to audit specific actions and privileges. You can add conditions to your policies to exclude unnecessary audit records. You an also use the EXCEPT clause in the AUDIT POLICY statement to exclude specific users who do not need to be audited. See Auditing Activities with Unified Audit Policies and the AUDIT Statement for more information about unified audit policies.

You can use fine-grained auditing, which is described in Auditing Specific Activities with Fine-Grained Auditing, to audit specific actions.

Continue this process until you have gathered enough evidence to draw conclusions about the origin of the suspicious database activity.

2. Audit common suspicious activities.

Common suspicious activities are as follows:

- Users who access the database during unusual hours
- Multiple failed user login attempts
- · Login attempts by non-existent users

In addition, be aware that sensitive data, such as credit card numbers, can appear in the audit trail columns, such as SQL text when used in the SQL query. You should also monitor users who share accounts or multiple users who are logging in from the same IP address. You can query the UNIFIED_AUDIT_TRAIL data dictionary view to find this kind of activity. For a very granular approach, create fine-grained audit policies.

A.11.4 Audits of Sensitive Data

Oracle recommends that you include the ACTIONS ALL clause when you create unified audit policies on sensitive objects.

Including this clause ensures the generation of audit record for both direct access and indirect access of these sensitive objects. Only use ACTIONS ALL for the audit of sensitive objects.

Related Topics

Example: Auditing All Actions on a Table
 The CREATE AUDIT POLICY statement can audit all actions on a table.

A.11.5 Recommended Audit Settings

Oracle provides predefined policies that contain recommended audit settings that apply to most sites.



For example:

- ORA_SECURECONFIG audits the same default audit settings from Oracle Database Release 11g. It tracks the use of a number of privileges such as ALTER ANY TABLE, GRANT ANY PRIVILEGE, and CREATE USER. The actions that it tracks include ALTER USER, CREATE ROLE, LOGON, and other commonly performed activities. This policy is enabled by default only when the database is created in Oracle Database Release 12c.
- ORA_DATABASE_PARAMETER audits commonly used Oracle Database parameter settings:

 ALTER_DATABASE, ALTER_SYSTEM, and CREATE_SPFILE. By default, this policy is not enabled.
- ORA_ACCOUNT_MGMT audits the commonly used user account and privilege settings: CREATE USER, ALTER USER, DROP USER, CREATE ROLE, DROP ROLE, ALTER ROLE, SET ROLE, GRANT, and REVOKE. By default, this policy is not enabled.



Auditing Activities with the Predefined Unified Audit Policies for detailed information about these and other predefined audit policies

A.11.6 Best Practices for Querying the UNIFIED_AUDIT_TRAIL Data Dictionary View

To get the best results from querying the ${\tt UNIFIED_AUDIT_TRAIL}$ data dictionary view, you should follow these guidelines.

- Ensure the statistics of unified audit internal table are up to date.
 Execute the DBMS_STATS.GATHER_TABLE_STATS procedure on the AUD\$UNIFIED table in the AUD\$YS schema to ensure that the unified audit table statistics are updated before you query the UNIFIED AUDIT TRAIL data dictionary view.
- 2. Load the unified audit records that were written to operating system spillover files. You can do this either explicitly or by configuring an Oracle Scheduler job, using the DBMS AUDIT MGMT.LOAD UNIFIED AUDIT FILES procedure.
- When the number of records in the unified audit trail reaches a significantly large number (for example, a million), then initiate the proper archiving and purging mechanisms.
 - Archiving and purging the unified audit trial reduces the amount of data that otherwise could grow and cause read performance problems. Oracle recommends that you configure standard purging policies. The purging policies that you create will depend on the rate of audit records that are generated on your system. Frequent purges are required for high audit record generation rates.
- 4. Move the unified audit trail to a custom tablespace. Using a custom tablespace enables you to better manage audit data and reduces the impact on other objects in the SYSAUX tablespace. By default, the unified audit trail records are written to the SYSAUX tablespace. To use a different tablespace, execute the DBMS AUDIT MGMT.SET AUDIT TRAIL LOCATION procedure.
- 5. When you query the UNIFIED_AUDIT_TRAIL data dictionary view, include the EVENT_TIMESTAMP_UTC column in a WHERE clause.



The EVENT_TIMESTAMP_UTC column records the timestamp of audited events in the UTC timezone. Including this column in the query helps to achieve the partition pruning, and thus improves read performance of the UNIFIED AUDIT TRAIL view.

Related Topics

- Moving Operating System Audit Records into the Unified Audit Trail
 Audit records that have been written to the spillover audit files can be moved to the unified audit trail database table.
- Archiving the Audit Trail
 You can archive the traditional operating system, unified database, and traditional database audit trails.
- Purging Audit Trail Records
 The DBMS_AUDIT_MGMT PL/SQL package can schedule automatic purge jobs, manually purge audit records, and perform other audit trail operations.

A.12 Addressing the CONNECT Role Change

The CONNECT role, introduced with Oracle Database version 7, added new and robust support for database roles.

- Why Was the CONNECT Role Changed?
 The CONNECT role is used in sample code, applications, documentation, and technical papers.
- How the CONNNECT Role Change Affects Applications
 The CONNECT role changes can be seen in database upgrades, account provisioning, and installation of applications using new databases.
- How the CONNECT Role Change Affects Users
 The change to the CONNECT role affects general users, application developers, and client/server applications differently.
- Approaches to Addressing the CONNECT Role Change
 Oracle recommends three approaches to address the impact of the CONNECT role change.

A.12.1 Why Was the CONNECT Role Changed?

The CONNECT role is used in sample code, applications, documentation, and technical papers.

In Oracle Database 10*g* Release 2 (10.2), the CONNECT role was changed. If you are upgrading from a release earlier than Oracle Database 10.2 to the current release, then you should be aware of how the CONNECT role has changed in the most recent release.

The CONNECT role was originally established a special set of privileges. These privileges were as follows: ALTER SESSION, CREATE CLUSTER, CREATE DATABASE LINK, CREATE SEQUENCE, CREATE SESSION, CREATE SYNONYM, CREATE TABLE, CREATE VIEW.

Beginning in Oracle Database 10*g* Release 2, the CONNECT role has only the CREATE SESSION privilege, all other privileges are removed. Starting with Oracle Database 12*c* Release 1, the CONNECT role had the CREATE SESSION and SET CONTAINER privileges.

Although the CONNECT role was frequently used to provision new accounts in Oracle Database, connecting to the database does not require all those privileges. Making this change enables you to enforce good security practices more easily.

Each user should have only the privileges needed to perform his or her tasks, an idea called the principle of least privilege. Least privilege mitigates risk by limiting privileges, so that it remains easy to do what is needed while concurrently reducing the ability to do inappropriate things, either inadvertently or maliciously.

A.12.2 How the CONNNECT Role Change Affects Applications

The CONNECT role changes can be seen in database upgrades, account provisioning, and installation of applications using new databases.

- How the CONNECT Role Change Affects Database Upgrades
 You should be aware of how the CONNECT role affects database upgrades.
- How the CONNECT Role Change Affects Account Provisioning
 You should be aware of how the CONNECT role affects accounts provisioning.
- How the CONNECT Role Change Affects Applications Using New Databases
 You should be aware of how the CONNECT role affects applications that use new
 databases.

A.12.2.1 How the CONNECT Role Change Affects Database Upgrades

You should be aware of how the CONNECT role affects database upgrades.

Upgrading your existing Oracle database to Oracle Database 10*g* Release 2 (10.2) automatically changes the CONNECT role to have only the CREATE SESSION privilege.

Most applications are not affected because the applications objects already exist: no new tables, views, sequences, synonyms, clusters, or database links need to be created.

Applications that create tables, views, sequences, synonyms, clusters, or database links, or that use the ALTER SESSION command dynamically, may fail due to insufficient privileges.

A.12.2.2 How the CONNECT Role Change Affects Account Provisioning

You should be aware of how the CONNECT role affects accounts provisioning.

If your application or DBA grants the CONNECT role as part of the account provisioning process, then only CREATE SESSION privileges are included. Any additional privileges must be granted either directly or through another role.

This issue can be addressed by creating a new customized database role.



Approaches to Addressing the CONNECT Role Change

A.12.2.3 How the CONNECT Role Change Affects Applications Using New Databases

You should be aware of how the CONNECT role affects applications that use new databases.

New databases created using the Oracle Database 10g Release 2 (10.2) Utility (DBCA), or using database creation templates generated from DBCA, define the CONNECT role with only the CREATE SESSION privilege.

Installing an application to use a new database may fail if the database schema used for the application is granted privileges solely through the CONNECT role.

A.12.3 How the CONNECT Role Change Affects Users

The change to the CONNECT role affects general users, application developers, and client/server applications differently.

- How the CONNECT Role Change Affects General Users
 You should be aware of how the CONNECT role affects general users.
- How the CONNECT Role Change Affects Application Developers
 You should be aware of how the CONNECT role affects application developers.
- How the CONNECT Role Change Affects Client Server Applications
 You should be aware of how the CONNECT role affects client server applications.

A.12.3.1 How the CONNECT Role Change Affects General Users

You should be aware of how the CONNECT role affects general users.

The new CONNECT role supplies only the CREATE SESSION privilege. Users who connect to the database to use an application are not affected, because the CONNECT role still has the CREATE SESSION privilege.

However, appropriate privileges will not be present for a certain set of users if they are provisioned solely with the CONNECT role. These are users who create tables, views, sequences, synonyms, clusters, or database links, or use the ALTER SESSION command. The privileges they need are no longer provided with the CONNECT role. To authorize the additional privileges needed, the database administrator must create and apply additional roles for the appropriate privileges, or grant them directly to the users who need them.

Note that the ALTER SESSION privilege is required for setting events. Few database users should require the ALTER SESSION privilege.

The ALTER SESSION privilege is *not* required for other alter session commands.

A.12.3.2 How the CONNECT Role Change Affects Application Developers

You should be aware of how the CONNECT role affects application developers.

Application developers provisioned solely with the CONNECT role do not have appropriate privileges to create tables, views, sequences, synonyms, clusters, or database links, nor to use the ALTER SESSION statement.

You must either create and apply additional roles for the appropriate privileges, or grant them directly to the application developers who need them.

A.12.3.3 How the CONNECT Role Change Affects Client Server Applications

You should be aware of how the CONNECT role affects client server applications.

Most client/server applications that use dedicated user accounts will not be affected by this change.

However, applications that create private synonyms or temporary tables using dynamic SQL in the user schema during account provisioning or run-time operations will be affected. They will require additional roles or grants to acquire the system privileges appropriate to their activities.

A.12.4 Approaches to Addressing the CONNECT Role Change

Oracle recommends three approaches to address the impact of the CONNECT role change.

Creating a New Database Role

The privileges removed from the CONNECT role can be managed by creating a new database role.

Restoring the CONNECT Privilege

The rstrconn.sql script restores the CONNECT privileges.

Data Dictionary View to Show CONNECT Grantees

The DBA_CONNECT_ROLE_GRANTEES data dictionary view enables administrators who continue using the old CONNECT role to see which users have that role.

Least Privilege Analysis Studies

Oracle partners and application providers should conduct a least privilege analysis so that they can deliver more secure products to their Oracle customers.

A.12.4.1 Creating a New Database Role

The privileges removed from the CONNECT role can be managed by creating a new database role.

L. Connect to the upgraded Oracle database and create a new database role.

The following example uses a role called my app developer.

```
CREATE ROLE my_app_developer;
GRANT CREATE TABLE, CREATE VIEW, CREATE SEQUENCE, CREATE SYNONYM, CREATE CLUSTER,
CREATE DATABASE LINK, ALTER SESSION TO my app developer;
```

2. Determine which users or database roles have the CONNECT role, and grant the new role to these users or roles.

```
SELECT USER$.NAME, ADMIN_OPTION, DEFAULT_ROLE
FROM USER$, SYSAUTH$, DBA_ROLE_PRIVS
WHERE PRIVILEGE# =
(SELECT USER# FROM USER$ WHERE NAME = 'CONNECT')
AND USER$.USER# = GRANTEE#
AND GRANTEE = USER$.NAME
AND GRANTED_ROLE = 'CONNECT';
```

ADMIN_OPTI	DEF
YES	YES
NO	YES

```
GRANT my_app_developer TO R1 WITH ADMIN OPTION; GRANT my app developer TO R2;
```

3. Determine the privileges that users require by creating a privilege analysis policy.

The information that you gather can then be analyzed and used to create additional database roles with finer granularity. Privileges that are not used can then be revoked for specific users.

For example:



4. After a period of time, disable the privilege analysis policy and then generate a report.

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

EXEC DBMS PRIVILEGE CAPTURE.GENERATE RESULT ('my_app_dev_role_pol');
```

5. After you generate the report, query the privilege analysis data dictionary views.

For example:

```
SELECT USERNAME, SYS_PRIV, OBJECT_OWNER, OBJECT_NAME FROM DBA_USED_PRIVS;
```

Related Topics

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

A.12.4.2 Restoring the CONNECT Privilege

The rstrconn.sql script restores the CONNECT privileges.

After a database upgrade or new database creation, you can use this script to grant the privileges that were removed from the CONNECT role in Oracle Database 10g release 2 (10.2). If you use this approach, then you should revoke privileges that are not used from users who do not need them.

To restore the CONNECT privilege:

1. Run the rstrconn.sql script, which is in the \$ORACLE HOME/rdbms/admin directory.

```
@$ORACLE HOME/rdbm admin/rstrconn.sql
```

2. Monitor the privileges that are used.

For example:

```
CREATE AUDIT POLICY connect_priv_pol
PRIVILEGES AUDIT CREATE TABLE, CREATE SEQUENCE, CREATE SYNONYM, CREATE DATABASE
LINK, CREATE CLUSTER, CREATE VIEW, ALTER SESSION;
AUDIT POLICY connect priv pol BY psmith;
```

3. Periodically, monitor database privilege usage.

For example:

ACME	ALTER SESSION
APPS	CREATE TABLE
8 rows selected	

A.12.4.3 Data Dictionary View to Show CONNECT Grantees

The DBA_CONNECT_ROLE_GRANTEES data dictionary view enables administrators who continue using the old CONNECT role to see which users have that role.

Table A-1 shows the columns in the DBA CONNECT ROLE GRANTEES view.

Table A-1 Columns and Contents for DBA_CONNECT_ROLE_GRANTEES

Column	Datatype	NULL	Description
GRANTEE	VARCHAR2(128)	NULL	User granted the CONNECT role
PATH_OF_CONNECT _ROLE_GRANT	VARCHAR2(4000	NULL	Role (or nested roles) by which the user is granted ${\tt CONNECT}$
ADMIN_OPT	VARCHAR2(3)	NULL	YES if user has the ADMIN option on CONNECT; otherwise, NO

A.12.4.4 Least Privilege Analysis Studies

Oracle partners and application providers should conduct a least privilege analysis so that they can deliver more secure products to their Oracle customers.

The principle of least privilege mitigates risk by limiting privileges to the minimum set required to perform a given function.

For each class of users that the analysis shows need the same set of privileges, create a role with only those privileges. Remove all other privileges from those users, and assign that role to those users. As needs change, you can grant additional privileges, either directly or through these new roles, or create new roles to meet new needs. This approach helps to ensure that inappropriate privileges have been limited, thereby reducing the risk of inadvertent or malicious harm.

You can create privilege analysis policies that show the use of privileges by database users. The policies capture this information and make it available in data dictionary views. Based on these reports, you can determine who should have access to your data.

Related Topics

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



B

Data Encryption and Integrity Parameters

The sqlnet.ora file has data encryption and integrity parameters.

- About Using sqlnet.ora for Data Encryption and Integrity
 You can use the default parameter settings as a guideline for configuring data encryption
 and integrity.
- Sample sqlnet.ora File
 The sample sqlnet.ora configuration file is based on a set of clients with similar characteristics and a set of servers with similar characteristics.
- Data Encryption and Integrity Parameters
 Oracle provides data and integrity parameters that you can set in the sqlnet.ora file.

B.1 About Using sqlnet.ora for Data Encryption and Integrity

You can use the default parameter settings as a guideline for configuring data encryption and integrity.

This sqlnet.ora file is generated when you perform the network configuration described in Configuring Oracle Database Native Network Encryption and Data Integrity and Configuring Transport Layer Security Authentication. Also provided are encryption and data integrity parameters.

B.2 Sample sqlnet.ora File

The sample sqlnet.ora configuration file is based on a set of clients with similar characteristics and a set of servers with similar characteristics.

The file includes examples of Oracle Database encryption and data integrity parameters.

By default, the sqlnet.ora file is located in the <code>ORACLE_HOME/network/admin</code> directory or in the location set by the <code>TNS_ADMIN</code> environment variable. Ensure that you have properly set the <code>TNS_ADMIN</code> variable to point to the correct sqlnet.ora file. See <code>SQL*Plus</code> User's <code>Guide</code> and <code>Reference</code> for more information and examples of setting the <code>TNS_ADMIN</code> variable.

Trace File Setup

```
#Trace file setup
trace_level_server=16
trace_level_client=16
trace_directory_server=/orant/network/trace
trace_directory_client=/orant/network/trace
trace_file_client=cli
trace_file_server=srv
trace_unique_client=true
```

Oracle Database Native Network Encryption

```
sqlnet.encryption_server=accepted
sqlnet.encryption client=requested
```

```
sqlnet.encryption_types_server=(RC4_40)
sqlnet.encryption types client=(RC4_40)
```



The RC4_40 algorithm is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

Oracle Database Network Data Integrity

```
#ASO Checksum
sqlnet.crypto_checksum_server=requested
sqlnet.crypto_checksum_client=requested
sqlnet.crypto_checksum_types_server = (SHA256)
sqlnet.crypto_checksum_types_client = (SHA256)
```

Transport Layer Security

Common

```
#Common
automatic_ipc = off
sqlnet.authentication_services = (beq)
names.directory path = (TNSNAMES)
```

Kerberos

```
#Kerberos
sqlnet.authentication_services = (beq, kerberos5)
sqlnet.authentication_kerberos5_service = oracle
sqlnet.kerberos5_conf= /krb5/krb.conf
sqlnet.kerberos5_keytab= /krb5/v5srvtab
sqlnet.kerberos5_realms= /krb5/krb.realm
sqlnet.kerberos5_cc_name = /krb5/krb5.cc
sqlnet.kerberos5_clockskew=900
sqlnet.kerberos5_conf_mit=false
```

RADIUS

```
#Radius
sqlnet.authentication_services = (beq, RADIUS )
sqlnet.radius_authentication_timeout = (10)
sqlnet.radius_authentication_retries = (2)
sqlnet.radius_authentication_port = (1645)
sqlnet.radius_send_accounting = OFF
sqlnet.radius_secret = /orant/network/admin/radius.key
sqlnet.radius_authentication = radius.us.example.com
sqlnet.radius_challenge_response = OFF
```



sqlnet.radius_challenge_keyword = challenge
sqlnet.radius_challenge_interface =
oracle/net/radius/DefaultRadiusInterface
sqlnet.radius classpath = /jre1.1/

B.3 Data Encryption and Integrity Parameters

Oracle provides data and integrity parameters that you can set in the sqlnet.ora file.

About the Data Encryption and Integrity Parameters

The data encryption and integrity parameters control the type of encryption algorithm you are using.

SQLNET.ENCRYPTION SERVER

The SQLNET.ENCRYPTION_SERVER parameter specifies the encryption behavior when a client or a server acting as a client connects to this server.

SQLNET.ENCRYPTION CLIENT

The SQLNET.ENCRYPTION_CLIENT parameter specifies the encryption behavior when this client or server acting as a client connects to a server.

SOLNET.CRYPTO CHECKSUM SERVER

The SQLNET.CRYPTO_CHECKSUM_SERVER parameter specifies the data integrity behavior when a client or another server acting as a client connects to this server.

SQLNET.CRYPTO CHECKSUM CLIENT

The SQLNET.CRYPTO_CHECKSUM_CLIENT parameter specifies the desired data integrity behavior when this client or server acting as a client connects to a server.

SQLNET.ENCRYPTION TYPES SERVER

The SQLNET.ENCRYPTION_TYPES_SERVER parameter specifies encryption algorithms this server uses in the order of the intended use.

SQLNET.ENCRYPTION TYPES CLIENT

The SQLNET.ENCRYPTION_TYPES_CLIENT parameter specifies encryption algorithms this client or the server acting as a client uses.

SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER

The SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER parameter specifies data integrity algorithms that this server or client to another server uses, in order of intended use.

SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT

The SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT parameter specifies a list of data integrity algorithms that this client or server acting as a client uses.

B.3.1 About the Data Encryption and Integrity Parameters

The data encryption and integrity parameters control the type of encryption algorithm you are using.

If you do not specify any values for Server Encryption, Client Encryption, Server Checksum, or Client Checksum, the corresponding configuration parameters do not appear in the sqlnet.ora file. However, the defaults are ACCEPTED.

For both data encryption and integrity algorithms, the server selects the first algorithm listed in its sqlnet.ora file that matches an algorithm listed in the client sqlnet.ora file, or in the client installed list if the client lists no algorithms in its sqlnet.ora file. If there are no entries in the server sqlnet.ora file, the server sequentially searches its installed list to match an item on the client side—either in the client sqlnet.ora file or in the client installed list. If no match can

be made and one side of the connection REQUIRED the algorithm type (data encryption or integrity), then the connection fails. Otherwise, the connection succeeds with the algorithm type inactive.

Data encryption and integrity algorithms are selected independently of each other. Encryption can be activated without integrity, and integrity can be activated without encryption, as shown by Table B-1:

Table B-1 Algorithm Type Selection

Encryption Selected?	Integrity Selected?
Yes	No
Yes	Yes
No	Yes
No	No

Related Topics

- Configuring Oracle Database Native Network Encryption and Data Integrity
 You can configure native Oracle Net Services data encryption and data integrity for both servers and clients.
- About Activating Encryption and Integrity
 In any network connection, both the client and server can support multiple encryption algorithms and integrity algorithms.

B.3.2 SQLNET.ENCRYPTION_SERVER

The SQLNET.ENCRYPTION_SERVER parameter specifies the encryption behavior when a client or a server acting as a client connects to this server.

The behavior of the server partially depends on the SQLNET.ENCRYPTION_CLIENT setting at the other end of the connection.

Table B-2 describes the SQLNET. ENCRYPTION SERVER parameter attributes.

Table B-2 SQLNET.ENCRYPTION_SERVER Parameter Attributes

Attribute	Description
Syntax	SQLNET.ENCRYPTION_SERVER = valid_value
Valid Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default Setting	ACCEPTED



Oracle Database Net Services Reference for more information about the SQLNET.ENCRYPTION SERVER parameter



B.3.3 SQLNET.ENCRYPTION_CLIENT

The SQLNET.ENCRYPTION_CLIENT parameter specifies the encryption behavior when this client or server acting as a client connects to a server.

The behavior of the client partially depends on the value set for <code>SQLNET.ENCRYPTION_SERVER</code> at the other end of the connection.

Table B-3 describes the SQLNET. ENCRYPTION CLIENT parameter attributes.

Table B-3 SQLNET.ENCRYPTION_CLIENT Parameter Attributes

Attribute	Description
Syntax	SQLNET.ENCRYPTION_CLIENT = valid_value
Valid Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default Setting	ACCEPTED



Oracle Database Net Services Reference for more information about the ${\tt SQLNET.ENCRYPTION_CLIENT}$ parameter

B.3.4 SQLNET.CRYPTO_CHECKSUM_SERVER

The SQLNET.CRYPTO_CHECKSUM_SERVER parameter specifies the data integrity behavior when a client or another server acting as a client connects to this server.

The behavior partially depends on the SQLNET.CRYPTO_CHECKSUM_CLIENT setting at the other end of the connection.

Table B-4 describes the SQLNET.CRYPTO_CHECKSUM_SERVER parameter attributes.

Table B-4 SQLNET.CRYPTO_CHECKSUM_SERVER Parameter Attributes

Attribute	Description
Syntax	SQLNET.CRYPTO_CHECKSUM_SERVER = valid_value
Valid Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default Setting	ACCEPTED



Oracle Database Net Services Reference for more information about the SQLNET.CRYPTO_CHECKSUM_SERVER parameter



B.3.5 SQLNET.CRYPTO_CHECKSUM_CLIENT

The SQLNET.CRYPTO_CHECKSUM_CLIENT parameter specifies the desired data integrity behavior when this client or server acting as a client connects to a server.

The behavior partially depends on the SQLNET.CRYPTO_CHECKSUM_SERVER setting at the other end of the connection.

Table B-5 describes the SQLNET.CRYPTO CHECKSUM CLIENT parameter attributes.

Table B-5 SQLNET.CRYPTO_CHECKSUM_CLIENT Parameter Attributes

Attribute	Description
Syntax	SQLNET.CRYPTO_CHECKSUM_CLIENT = valid_value
Valid Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default Setting	ACCEPTED

B.3.6 SQLNET.ENCRYPTION_TYPES_SERVER

The SQLNET.ENCRYPTION_TYPES_SERVER parameter specifies encryption algorithms this server uses in the order of the intended use.

This list is used to negotiate a mutually acceptable algorithm with the client end of the connection. Each algorithm is checked against the list of available client algorithm types until a match is found. If an algorithm that is not installed is specified on this side, the connection terminates with the error message ORA-12650: No common encryption or data integrity algorithm.

Table B-6 describes the SQLNET.ENCRYPTION_TYPES_SERVER parameter attributes.

Table B-6 SQLNET.ENCRYPTION_TYPES_SERVER Parameter Attributes

Attribute	Description	
Syntax	<pre>SQLNET.ENCRYPTION_TYPES_SERVER = (valid_encryption_algorithm [,valid_encryption_algorithm])</pre>	
Valid Values	 AES256: AES (256-bit key size) AES192: AES (192-bit key size) AES128: AES (128-bit key size) 	
Default Setting	If no algorithms are defined in the local sqlnet.ora file, then all installed algorithms are used in a negotiation in the preceding sequence.	
Usage Notes	You can specify multiple encryption algorithms. It can be either a single value or a list of algorithm names. For example, either of the following encryption parameters is acceptable:	
	SQLNET.ENCRYPTION_TYPES_SERVER=(AES256) SQLNET.ENCRYPTION_TYPES_SERVER=(AES256,AES192,AES128)	



See Also:

Oracle Database Net Services Reference for more information about the SQLNET.ENCRYPTION_TYPES_SERVER parameter

B.3.7 SQLNET.ENCRYPTION_TYPES_CLIENT

The SQLNET.ENCRYPTION_TYPES_CLIENT parameter specifies encryption algorithms this client or the server acting as a client uses.

This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. If an algorithm that is not installed is specified on this side, the connection terminates with the ORA-12650: No common encryption or data integrity algorithm error message.

Table B-7 describes the SQLNET. ENCRYPTION TYPES CLIENT parameter attributes.

Table B-7 SQLNET.ENCRYPTION TYPES CLIENT Parameter Attributes

Attribute	Description	
Syntax	<pre>SQLNET.ENCRYPTION_TYPES_CLIENT = (valid_encryption_algorithm [,valid_encryption_algorithm])</pre>	
Valid Values	 AES256: AES (256-bit key size). AES192: AES (192-bit key size). AES128: AES (128-bit key size). 	
Default Setting	If no algorithms are defined in the local sqlnet.ora file, all installed algorithms are used in a negotiation.	
Usage Notes	You can specify multiple encryption algorithms by separating each one with a comma. For example:	
	SQLNET.ENCRYPTION_TYPES_CLIENT=(AES256, AES192, AES128)	

See Also:

Oracle Database Net Services Reference for more information about the ${\tt SQLNET.ENCRYPTION_TYPES_CLIENT}$ parameter

B.3.8 SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER

The SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER parameter specifies data integrity algorithms that this server or client to another server uses. in order of intended use.

This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. Each algorithm is checked against the list of available client algorithm types until a match is found. If an algorithm is specified that is not installed on this side, the connection terminates with the ORA-12650: No common encryption or data integrity algorithm error error message.



Table B-8 describes the SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER parameter attributes.

Table B-8 SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER Parameter Attributes

Attribute	Description
Syntax	<pre>SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (valid_crypto_checksum_algorithm [,valid_crypto_checksum_algorithm])</pre>
Valid Values	 SHA512: SHA-2, produces a 512-bit hash. SHA384: SHA-2, produces a 384-bit hash. SHA256: SHA-2, produces a 256-bit hash. This is the default value. SHA1: Secure Hash Algorithm MD5: Message Digest 5
	Note: MD5 is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.
Default Setting	All available algorithms



Oracle Database Net Services Reference for more information about the ${\tt SQLNET.CRYPTO}$ CHECKSUM TYPES SERVER parameter

B.3.9 SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT

The SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT parameter specifies a list of data integrity algorithms that this client or server acting as a client uses.

This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. If an algorithm that is not installed on this side is specified, the connection terminates with the ORA-12650: No common encryption or data integrity algorithm error error message.

Table B-9 describes the SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT parameter attributes.

Table B-9 SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT Parameter Attributes

Attribute	Description
Syntax	SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (valid_crypto_checksum_algorithm
	[,valid_crypto_checksum_algorithm])

Table B-9 (Cont.) SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT Parameter Attributes

Attribute	Description
Valid Values	 SHA512: SHA-2, produces a 512-bit hash. SHA384: SHA-2, produces a 384-bit hash. SHA256: SHA-2, produces a 256-bit hash. This is the default value. SHA1: Secure Hash Algorithm MD5: Message Digest 5
	Note: MD5 is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.
Default Setting	If no algorithms are defined in the local sqlnet.ora file, all installed algorithms are used in a negotiation starting with SHA256.



Oracle Database Net Services Reference for more information about the ${\tt SQLNET.CRYPTO}$ CHECKSUM TYPES CLIENT parameter



C

Kerberos, TLS, and RADIUS Authentication Parameters

The sqlnet.ora and the database initialization files provide Kerberos, RADIUS, or TLS authentication parameters.

- Parameters for Clients and Servers Using Kerberos Authentication
 Oracle Database provides client and server parameters for using Kerberos authentication.
- Parameters for Clients and Servers Using Transport Layer Security
 Oracle provides parameters to control Transport Layer Security authentication.
- Parameters for Clients and Servers Using RADIUS Authentication Oracle provides parameters for RADIUS authentication.

C.1 Parameters for Clients and Servers Using Kerberos Authentication

Oracle Database provides client and server parameters for using Kerberos authentication.

Table C-1 lists parameters to insert into the configuration files for clients and servers using Kerberos.

Table C-1 Kerberos Authentication Parameters

File Name	Configuration Parameters
sqlnet.ora	SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5): Set on both client and server.
	SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=oracle: Set on both client and server.
	SQLNET.KERBEROS5_CC_NAME=/usr/tmp/DCE-CC: Not normally required on the server. If your client is on Microsoft Windows and is part of a domain, you may want to consider using the in-memory ticket cache and set this parameter to OSMSFT:// or MSLSA:.
	SQLNET.KERBEROS5_CLOCKSKEW=1200: Set on both client and server.
	SQLNET.KERBEROS5_CONF=/krb5/krb.conf: Set on both client and server. (Normally, this path in the client is different from the path in the server.)
	SQLNET.KERBEROS5_CONF_MIT=(TRUE) : Set this to TRUE on both the client and the server.
	SQLNET.KERBEROS5_REALMS=/krb5/krb.realms: This setting is not usually required for the client or the server.
	SQLNET.KERBEROS5_KEYTAB=/krb5/v5srvtab: Only set this parameter on the server, not the client.
	SQLNET.FALLBACK_AUTHENTICATION=FALSE: Set on both client and server.
initialization parameter file	OS_AUTHENT_PREFIX="": Set this parameter only on the server, not the client.

Related Topics

Step 6C: Set sqlnet.ora Parameters (Optional)
 You can set optional sqlnet.ora parameters, in addition to the required parameters, for better security.

C.2 Parameters for Clients and Servers Using Transport Layer Security

Oracle provides parameters to control Transport Layer Security authentication.

- Ways to Configure a Parameter for Transport Layer Security
 There are two ways to configure a parameter for Transport Layer Security (TLS).
- Transport Layer Security Authentication Parameters for Clients and Servers
 Oracle provides both static and dynamic Transport Layer Security (TLS) authentication
 parameters.
- Cipher Suite Parameters for Transport Layer Security
 You can configure cipher suite parameters for Transport Layer Security (TLS).
- Supported Transport Layer Security Cipher Suites
 Oracle Database supports a large number of cipher suites for Transport Layer Security (TLS).
- Transport Layer Security Version Parameters
 You can set a range of Transport Layer Security (TLS) parameters to configure the version of TLS to use.
- Transport Layer Security Client Authentication Parameters
 You can configure static and dynamic parameters for Secure Sockes Layer (TLS) on the client.
- Transport Layer Security X.509 Server Match Parameters
 The SSL_SERVER_DN_MATCH and SSL_SERVER_CERT_DN parameters validate the identity of the server to which a client connects.
- Oracle Wallet Location
 You must specify wallet location parameters for applications that must access an Oracle wallet for loading the security credentials into the process space.

C.2.1 Ways to Configure a Parameter for Transport Layer Security

There are two ways to configure a parameter for Transport Layer Security (TLS).

- Static: The name of the parameter that exists in the sqlnet.ora file. Parameters like SSL CIPHER SUITES and SSL VERSION can also be configured using the listener.ora file.
- Dynamic: The name of the parameter used in the security subsection of the Oracle Net address.

C.2.2 Transport Layer Security Authentication Parameters for Clients and Servers

Oracle provides both static and dynamic Transport Layer Security (TLS) authentication parameters.

Table C-2 describes the static and dynamic parameters for configuring TLS on the server.



Table C-2 TLS Authentication Parameters for Clients and Servers

Attribute	Description
Parameter Name (static)	SQLNET.AUTHENTICATION_SERVICES
Parameter Name (dynamic)	AUTHENTICATION
Parameter Type	String LIST
Parameter Class	Static
Permitted Values	Add TCPS to the list of available authentication services.
Default Value	No default value.
Description	To control which authentication services a user wants to use.
	Note: The dynamic version supports only the setting of one type.
Existing/New Parameter	Existing
Syntax (static)	SQLNET.AUTHENTICATION_SERVICES = (TCPS, selected_method_1, selected_method_2)
Example (static)	SQLNET.AUTHENTICATION_SERVICES = (TCPS, radius)
Syntax (dynamic)	AUTHENTICATION = string
Example (dynamic)	AUTHENTICATION = (TCPS)

C.2.3 Cipher Suite Parameters for Transport Layer Security

You can configure cipher suite parameters for Transport Layer Security (TLS).

Table C-3 describes the static and dynamic parameters for configuring cipher suites.

Table C-3 Cipher Suite Parameters for Transport Layer Security

Attribute	Description
Parameter Name (static)	SSL_CIPHER_SUITES
Parameter Name (dynamic)	SSL_CIPHER_SUITES
Parameter Type	String LIST
Parameter Class	Static
Permitted Values	Any known TLS cipher suite
Default Value	No default
Description	Controls the combination of encryption and data integrity used by TLS.
Existing/New Parameter	Existing
Syntax (static)	<pre>SSL_CIPHER_SUITES=(SSL_cipher_suite1[, SSL_cipher_suite2, SSL_cipher_suiteN])</pre>
Example (static)	SSL_CIPHER_SUITES=(SSL_DH_DSS_WITH_DES_CBC_SHA)
Syntax (dynamic)	SSL_CIPHER_SUITES=(SSL_cipher_suite1
	[, SSL_cipher_suite2,SSL_cipher_suiteN])
Example (dynamic)	SSL_CIPHER_SUITES=(SSL_DH_DSS_WITH_DES_CBC_SHA)



C.2.4 Supported Transport Layer Security Cipher Suites

Oracle Database supports a large number of cipher suites for Transport Layer Security (TLS).

The cipher suites are as follows:

- TLS ECDHE ECDSA WITH AES 128 GCM SHA256
- TLS ECDHE ECDSA WITH AES 128 CBC SHA
- TLS ECDHE ECDSA WITH AES 128 CBC SHA256
- TLS ECDHE ECDSA WITH AES 256 CBC SHA
- TLS ECDHE ECDSA WITH AES 256 CBC SHA384
- TLS ECDHE ECDSA WITH AES 256 GCM SHA384
- TLS RSA WITH AES 128 CBC SHA256
- TLS RSA WITH AES 128 GCM SHA256
- TLS RSA WITH AES 128 CBC SHA
- TLS RSA WITH AES 256 CBC SHA
- TLS RSA WITH AES 256 CBC SHA256

Related Topics

TLS Cipher Suite Authentication, Encryption, Integrity, and TLS Versions
 Oracle Database supports a set of cipher suites that are set by default when you install Oracle Database.

C.2.5 Transport Layer Security Version Parameters

You can set a range of Transport Layer Security (TLS) parameters to configure the version of TLS to use.

Table C-4 describes the SSL_VERSION static and dynamic parameters for configuring the version of TLS to be used.

Table C-4 Transport Layer Security Version Parameters

Description
SSL_VERSION
SSL_VERSION
string
Static



Table C-4 (Cont.) Transport Layer Security Version Parameters

Attribute	Description
Permitted Values	Any version which is valid to TLS. Values are as follows:
	undetermined 1.0 1.1 1.2 3.0
	If you want to specify one version or another version, then use "or". The following values are permitted:
	1.0 or 3.0 1.2 or 3.0 1.1 or 1.0 1.2 or 1.0 1.2 or 1.1 1.1 or 1.0 or 3.0 1.2 or 1.0 or 3.0 1.2 or 1.1 or 1.0 1.2 or 1.1 or
	3.0 1.2 or 1.1 or 1.0 or 3
Default Value	1.2, 1.1, or 1.0
	If you want to use a specific value (such as 1.2), then you must set it explicitly.
Description	To force the version of the TLS connection.
Existing/New Parameter	New
Syntax (static)	SSL_VERSION=version
Example (static)	SSL_VERSION=1.1
Syntax (dynamic)	SSL_VERSION=version
Example (dynamic)	SSL_VERSION=1.1 or 1.2



The $\mathtt{ADD_SSLv3_IMPLICITLY}$ initialization parameter has no effect on the $\mathtt{SSL_VERSION}$ parameter.

C.2.6 Transport Layer Security Client Authentication Parameters

You can configure static and dynamic parameters for Secure Sockes Layer (TLS) on the client.

Table C-5 describes the SSL_CLIENT_AUTHENTICATION parameters.

Table C-5 Transport Layer Security Client Authentication Parameters

Attribute	Description
Parameter Name (static)	SSL_CLIENT_AUTHENTICATION
Parameter Name (dynamic)	SSL_CLIENT_AUTHENTICATION
Parameter Type	Boolean
Parameter Class	Static
Permitted Values	TRUE or FALSE
Default Value	TRUE
Description	To control whether a client, in addition to the server, is authenticated using TLS.
Existing/New Parameter	New

Table C-5 (Cont.) Transport Layer Security Client Authentication Parameters

Attribute	Description
Syntax (static)	SSL_CLIENT_AUTHENTICATION={ TRUE FALSE}
Example (static)	SSL_CLIENT_AUTHENTICATION=FALSE
Syntax (dynamic)	<pre>SSL_CLIENT_AUTHENTICATION={TRUE FALSE}</pre>
Example (dynamic)	SSL_CLIENT_AUTHENTICATION=FALSE

C.2.7 Transport Layer Security X.509 Server Match Parameters

The SSL_SERVER_DN_MATCH and SSL_SERVER_CERT_DN parameters validate the identity of the server to which a client connects.

SSL_SERVER_DN_MATCH

The ${\tt SSL_SERVER_DN_MATCH}$ parameter forces the server's distinguished name (DN) to match the name of the service.

SSL_SERVER_CERT_DN
 The SSL_SERVER_CERT_DN specifies the distinguished name (DN) of a server.

C.2.7.1 SSL_SERVER_DN_MATCH

The ${\tt SSL_SERVER_DN_MATCH}$ parameter forces the server's distinguished name (DN) to match the name of the service.

Table C-6 describes the SSL SERVER DN MATCH parameter.

Table C-6 SSL_SERVER_DN_MATCH Parameter

Attribute	Description
Parameter Name	SSL_SERVER_DN_MATCH
Where Stored	sqlnet.ora
Purpose	Use this parameter to force the server's distinguished name (DN) to match its service name. If you force the match verifications, TLS ensures that the certificate is from the server. If you choose not to enforce the match verification, TLS performs the check but permits the connection, regardless of whether there is a match. Not forcing the match lets the server potentially fake its identity.
Values	yes on true. Specify to enforce a match. If the DN matches the service name, the connection succeeds; otherwise, the connection fails.
	no off false. Specify to not enforce a match. If the DN does not match the service name, the connection is successful, but an error is logged to the sqlnet.log file.
Default	Oracle8i, or later: false. TLS client (always) checks server DN. If it does not match the service name, the connection succeeds but an error is logged to sqlnet.log file.
Usage Notes	Additionally configure the tnsnames.ora parameter SSL_SERVER_CERT_DN to enable server DN matching.



C.2.7.2 SSL SERVER CERT DN

The SSL SERVER CERT DN specifies the distinguished name (DN) of a server.

Table C-7 describes the <code>SSL_SERVER_CERT_DN</code> parameter.

Table C-7 SSL_SERVER_CERT_DN Parameter

Attribute	Description
Parameter Name	SSL_SERVER_CERT_DN
Where Stored	tnsnames.ora. It can be stored on the client, for every server it connects to, or it can be stored in the LDAP directory, for every server it connects to, updated centrally.
Purpose	This parameter specifies the distinguished name (DN) of the server. The client uses this information to obtain the list of DNs it expects for each of the servers to force the server's DN to match its service name.
Values	Set equal to distinguished name (DN) of the server.
Default	N/A
Usage Notes	Additionally configure the sqlnet.ora parameter SSL_SERVER_DN_MATCH to enable server DN matching.
Example	<pre>dbalias=(description=address_list=(address=(protocol=tc ps) (host=hostname) (port=portnum))) (connect_data=(sid=Finance)) (security=(SSL_SERVER_CERT_DN="CN=Finance,CN=OracleCont ext,C=US,O=Acme"))</pre>

C.2.8 Oracle Wallet Location

You must specify wallet location parameters for applications that must access an Oracle wallet for loading the security credentials into the process space.

Table C-8 lists the configuration files in which you must specify the wallet locations.

- sqlnet.ora
- listener.ora

Table C-8 Wallet Location Parameters

Static Configuration	Dynamic Configuration
<pre>WALLET_LOCATION = (SOURCE= (METHOD=File) (METHOD_DATA= (DIRECTORY=your_wallet_dir))</pre>	<pre>MY_WALLET_DIRECTORY = your_wallet_dir</pre>
)	

The default wallet location is the ORACLE HOME directory.

C.3 Parameters for Clients and Servers Using RADIUS Authentication

Oracle provides parameters for RADIUS authentication.

sqlnet.ora File Parameters
 You can include RADIUS-specific parameters in the sqlnet.ora file.

Minimum RADIUS Parameters

At minimum, you should use the SQLNET.AUTHENTICATION_SERVICES and SQLNET.RADIUS.AUTHENTICATION parameters.

Initialization File Parameter for RADIUS

For RADIUS, you should set the OS AUTHENT PREFIX initialization parameter.

C.3.1 sqlnet.ora File Parameters

You can include RADIUS-specific parameters in the sqlnet.ora file.

SQLNET.AUTHENTICATION SERVICES

The SQLNET.AUTHENTICATION_SERVICES parameter configures the client or the server to use the RADIUS adapter.

SQLNET.RADIUS ALTERNATE

The SQLNET.RADIUS_ALTERNATE parameter sets the location of an alternate RADIUS server to be used if the primary server is unavailable for fault tolerance.

SQLNET.RADIUS ALTERNATE PORT

The SQLNET.RADIUS_ALTERNATE_PORT parameter sets the listening port for the alternate RADIUS server.

SQLNET.RADIUS_ALTERNATE_TIMEOUT

The SQLNET.RADIUS_ALTERNATE_TIMEOUT parameter sets the time for an alternate RADIUS server to wait for a response.

SQLNET.RADIUS_ALTERNATE_RETRIES

The SQLNET.RADIUS_ALTERNATE_RETRIES parameter sets the number of times that the alternate RADIUS server resends messages.

SQLNET.RADIUS_AUTHENTICATION

The SQLNET.RADIUS_AUTHENTICATION parameter sets the location of the primary RADIUS server, either host name or dotted decimal format.

SOLNET.RADIUS AUTHENTICATION INTERFACE

The SQLNET.RADIUS_AUTHENTICATION_INTERFACE parameter sets the name of the Java class that contains the GUI when RADIUS is in challenge-response (asynchronous) mode.

SOLNET.RADIUS AUTHENTICATION PORT

The SQLNET.RADIUS_AUTHENTICATION_PORT parameter sets the listening port of the primary RADIUS server.

SQLNET.RADIUS AUTHENTICATION TIMEOUT

The SQLNET.RADIUS_AUTHENTICATION_TIMEOUT parameter sets the time to wait for response.

SQLNET.RADIUS AUTHENTICATION RETRIES

The SQLNET.RADIUS_AUTHENTICATION_RETRIES parameter sets the number of times to resend authentication information.



SQLNET.RADIUS CHALLENGE RESPONSE

The SQLNET.RADIUS_CHALLENGE_RESPONSE parameter turns on or turns off the challenge-response or asynchronous mode support.

SQLNET.RADIUS CHALLENGE KEYWORD

The SQLNET.RADIUS_CHALLENGE_KEYWORD parameter sets the keyword to request a challenge from the RADIUS server.

SQLNET.RADIUS CLASSPATH

The SQLNET.RADIUS_CLASSPATH parameter sets the path for Java classes and the JDK Java libraries.

SOLNET.RADIUS SECRET

The SQLNET.RADIUS_SECRET parameter specifies the file name and location of the RADIUS secret key.

SQLNET.RADIUS SEND ACCOUNTING

The SQLNET.RADIUS SEND ACCOUNTING parameter turns accounting on or off.

C.3.1.1 SQLNET.AUTHENTICATION_SERVICES

The SQLNET.AUTHENTICATION_SERVICES parameter configures the client or the server to use the RADIUS adapter.

Table C-9 describes the SQLNET.AUTHENTICATION SERVICES parameter attributes.

Table C-9 SQLNET.AUTHENTICATION_SERVICES Parameter Attributes

Attribute	Description
Syntax	SQLNET.AUTHENTICATION_SERVICES=(radius)
Default setting	None

C.3.1.2 SQLNET.RADIUS ALTERNATE

The SQLNET.RADIUS_ALTERNATE parameter sets the location of an alternate RADIUS server to be used if the primary server is unavailable for fault tolerance.

Table C-10 describes the SQLNET.RADIUS_ALTERNATE parameter attributes.

Table C-10 SQLNET.RADIUS_ALTERNATE Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_ALTERNATE=alternate_RADIUS_server_hostname_or_ IP_address
Default setting	off

C.3.1.3 SQLNET.RADIUS_ALTERNATE_PORT

The SQLNET.RADIUS_ALTERNATE_PORT parameter sets the listening port for the alternate RADIUS server.

Table C-11 describes the SQLNET.RADIUS ALTERNATE PORT parameter attributes.



Table C-11 SQLNET.RADIUS_ALTERNATE_PORT Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_ALTERNATE_PORT=alternate_RADIUS_server_listening_port_number
Default setting	1645

C.3.1.4 SQLNET.RADIUS_ALTERNATE_TIMEOUT

The SQLNET.RADIUS_ALTERNATE_TIMEOUT parameter sets the time for an alternate RADIUS server to wait for a response.

Table C-12 describes the SQLNET.RADIUS ALTERNATE_TIMEOUT parameter attributes.

Table C-12 SQLNET.RADIUS_ALTERNATE_TIMEOUT Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_ALTERNATE_TIMEOUT=time_in_seconds
Default setting	5

C.3.1.5 SQLNET.RADIUS_ALTERNATE_RETRIES

The SQLNET.RADIUS_ALTERNATE_RETRIES parameter sets the number of times that the alternate RADIUS server resends messages.

Table C-13 describes the SQLNET.RADIUS_ALTERNATE_RETRIES parameter attributes.

Table C-13 SQLNET.RADIUS_ALTERNATE_RETRIES Parameter Attributes

Attribute	Description	
Syntax	SQLNET.RADIUS_ALTERNATE_RETRIES=n_times_to_resend	
Default setting	3	

C.3.1.6 SQLNET.RADIUS_AUTHENTICATION

The SQLNET.RADIUS_AUTHENTICATION parameter sets the location of the primary RADIUS server, either host name or dotted decimal format.

If the RADIUS server is on a different computer from the Oracle server, you must specify either the host name or the IP address of that computer.

Table C-14 describes the SQLNET.RADIUS_AUTHENTICATION parameter attributes.

Table C-14 SQLNET.RADIUS_AUTHENTICATION Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_AUTHENTICATION=RADIUS_server_IP_address
Default setting	localhost



C.3.1.7 SQLNET.RADIUS AUTHENTICATION INTERFACE

The SQLNET.RADIUS_AUTHENTICATION_INTERFACE parameter sets the name of the Java class that contains the GUI when RADIUS is in challenge-response (asynchronous) mode.

Table C-15 describes the SQLNET.RADIUS AUTHENTICATION INTERFACE parameter attributes.

Table C-15 SQLNET.RADIUS AUTHENTICATION INTERFACE Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_AUTHENTICATION_INTERFACE=Java_class_name
Default setting	<pre>DefaultRadiusInterface (oracle/net/radius/ DefaultRadiusInterface)</pre>

C.3.1.8 SQLNET.RADIUS_AUTHENTICATION_PORT

The SQLNET.RADIUS_AUTHENTICATION_PORT parameter sets the listening port of the primary RADIUS server.

Table C-16 describes the SQLNET.RADIUS AUTHENTICATION PORT parameter attributes.

Table C-16 SQLNET.RADIUS_AUTHENTICATION_PORT Parameter Attributes

Attribute	Description	
Syntax	SQLNET.RADIUS_AUTHENTICATION_PORT=port_number	
Default setting	1645	

C.3.1.9 SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

The SQLNET.RADIUS AUTHENTICATION TIMEOUT parameter sets the time to wait for response.

Table C-17 describes the SQLNET.RADIUS AUTHENTICATION TIMEOUT parameter attributes.

Table C-17 SQLNET.RADIUS_AUTHENTICATION_TIMEOUT Parameter Attributes

Attribute	Description	
Syntax	SQLNET.RADIUS_AUTHENTICATION_TIMEOUT=time_in_seconds	
Default setting	5	

C.3.1.10 SQLNET.RADIUS AUTHENTICATION RETRIES

The SQLNET.RADIUS_AUTHENTICATION_RETRIES parameter sets the number of times to resend authentication information.

Table C-18 describes the SQLNET.RADIUS_AUTHENTICATION_RETRIES parameter attributes.



Table C-18 SQLNET.RADIUS_AUTHENTICATION_RETRIES Parameter Attributes

Attribute	Description	
Syntax	SQLNET.RADIUS_AUTHENTICATION_RETRIES=n_times_to_resend	
Default setting	3	

C.3.1.11 SQLNET.RADIUS_CHALLENGE_RESPONSE

The SQLNET.RADIUS_CHALLENGE_RESPONSE parameter turns on or turns off the challenge-response or asynchronous mode support.

Table C-19 describes the SQLNET.RADIUS CHALLENGE RESPONSE parameter attributes.

Table C-19 SQLNET.RADIUS CHALLENGE RESPONSE Parameter Attributes

Attribute	Description	
Syntax	SQLNET.RADIUS_CHALLENGE_RESPONSE=on	
Default setting	off	

C.3.1.12 SQLNET.RADIUS CHALLENGE KEYWORD

The $SQLNET.RADIUS_CHALLENGE_KEYWORD$ parameter sets the keyword to request a challenge from the RADIUS server.

The user types no password on the client.

Table C-20 describes the SQLNET.RADIUS CHALLENGE KEYWORD parameter attributes.

Table C-20 SQLNET.RADIUS_CHALLENGE_KEYWORD Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_CHALLENGE_KEYWORD=keyword
Default setting	challenge

C.3.1.13 SQLNET.RADIUS CLASSPATH

The SQLNET.RADIUS_CLASSPATH parameter sets the path for Java classes and the JDK Java libraries.

If you decide to use the challenge-response authentication mode, then RADIUS presents the user with a Java-based graphical interface requesting first a password, then additional information, for example, a dynamic password that the user obtains from a token card.

Add the SQLNET.RADIUS_CLASSPATH parameter in the sqlnet.ora file to set the path for the Java classes for that graphical interface, and to set the path to the JDK Java libraries.

Table C-21 describes the SQLNET.RADIUS CLASSPATH parameter attributes.



Table C-21 SQLNET.RADIUS_CLASSPATH Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_CLASSPATH=path_to_GUI_Java_classes
Default setting	<pre>\$ORACLE_HOME/jlib/netradius.jar:\$ORACLE_HOME/JRE/lib/sparc/ native_threads</pre>

C.3.1.14 SQLNET.RADIUS SECRET

The SQLNET.RADIUS_SECRET parameter specifies the file name and location of the RADIUS secret key.

Table C-22 describes the SQLNET.RADIUS SECRET parameter attributes.

Table C-22 SQLNET.RADIUS_SECRET Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_SECRET=path_to_RADIUS_secret_key
Default setting	<pre>\$ORACLE_HOME/network/security/radius.key</pre>

C.3.1.15 SQLNET.RADIUS_SEND_ACCOUNTING

The SQLNET.RADIUS SEND ACCOUNTING parameter turns accounting on or off.

If you enable accounting, packets will be sent to the active RADIUS server at the listening port plus one. By default, packets are sent to port 1646. You need to turn this feature on only when your RADIUS server supports accounting and you want to keep track of the number of times the user is logging on to the system.

Table C-23 describes the SQLNET.RADIUS SEND ACCOUNTING parameter attributes.

Table C-23 SQLNET.RADIUS_SEND_ACCOUNTING Parameter Attributes

Attribute	Description
Syntax	SQLNET.RADIUS_SEND_ACCOUNTING=on
Default setting	off

C.3.2 Minimum RADIUS Parameters

At minimum, you should use the SQLNET.AUTHENTICATION_SERVICES and SQLNET.RADIUS.AUTHENTICATION parameters.

Use the following settings:

```
sqlnet.authentication_services = (radius)
sqlnet.radius.authentication = IP-address-of-RADIUS-server
```

C.3.3 Initialization File Parameter for RADIUS

For RADIUS, you should set the OS AUTHENT PREFIX initialization parameter.

For example:

OS_AUTHENT_PREFIX=""



D

Integrating Authentication Devices Using RADIUS

The RADIUS challenge-response user interface further enhances authentication in a RADIUS configuration.

- About the RADIUS Challenge-Response User Interface
 You can use third-party authentication vendors to customize the RADIUS challenge response user interface to fit a particular device.
- Customizing the RADIUS Challenge-Response User Interface
 You can customize OracleRadiusInterface interface by creating your own class.
- Example: Using the OracleRadiusInterface Interface
 You can use the OracleRadiusInterface interface to retrieve a user name and password.

D.1 About the RADIUS Challenge-Response User Interface

You can use third-party authentication vendors to customize the RADIUS challenge-response user interface to fit a particular device.

You can set up any authentication device that supports the RADIUS standard to authenticate Oracle users. When your authentication device uses the challenge-response mode, a graphical interface prompts the end user first for a password and then for additional information (for example, a dynamic password that the user obtains from a token card). This interface is Javabased to provide optimal platform independence.

Third-party vendors of authentication devices must customize this graphical user interface to fit their particular device. For example, a smart card vendor customizes the Oracle client to issue the challenge to the smart card reader. Then, when the smart card receives a challenge, it responds by prompting the user for more information, such as a PIN.

Related Topics

Configuring RADIUS Authentication
 RADIUS is a client/server security protocol widely used to enable remote authentication
 and access.

D.2 Customizing the RADIUS Challenge-Response User Interface

You can customize OracleRadiusInterface interface by creating your own class.

1. Open the sqlnet.ora file.

By default, the sqlnet.ora file is located in the $ORACLE_HOME/network/admin$ directory or in the location set by the <code>TNS_ADMIN</code> environment variable. Ensure that you have properly set the <code>TNS_ADMIN</code> variable to point to the correct sqlnet.ora file.

2. Locate the SQLNET.RADIUS_AUTHENTICATION_INTERFACE parameter, and replace the name of the class listed there (DefaultRadiusInterface), with the name of the new class that you have created.

When you make this change in the sqlnet.ora file, the class is loaded on the Oracle client in order to handle the authentication process.

3. Save and exit the sqlnet.ora file

The third party must implement the OracleRadiusInterface interface, which is located in the ORACLE.NET.RADIUS package.



SQL*Plus User's Guide and Reference for more information and examples of setting the TNS ADMIN variable

D.3 Example: Using the OracleRadiusInterface Interface

You can use the OracleRadiusInterface interface to retrieve a user name and password.

Example D-1 shows how to use the OracleRadiusInterface interface.

Example D-1 Using the OracleRadiusInterface Interface

```
public interface OracleRadiusInterface {
  public void radiusRequest();
  public void radiusChallenge(String challenge);
  public String getUserName();
  public String getPassword();
}
```

In this specification:

- radiusRequest prompts the end user for a user name and password, which will later be retrieved through getUserName and getPassword.
- getUserName extracts the user name the user enters. If this method returns an empty string, it is assumed that the user wants to cancel the operation. The user then receives a message indicating that the authentication attempt failed.
- getPassword extracts the password the user enters. If getUserName returns a valid string, but getPassword returns an empty string, the challenge keyword is replaced as the password by the database. If the user enters a valid password, a challenge may or may not be returned by the RADIUS server.
- radiusChallenge presents a request sent from the RADIUS server for the user to respond to the server's challenge.
- getResponse extracts the response the user enters. If this method returns a valid response, then that information populates the User-Password attribute in the new Access-Request packet. If an empty string is returned, the operation is aborted from both sides by returning the corresponding value.



Е

Oracle Database FIPS 140-2 Settings

Oracle supports the Federal Information Processing Standard (FIPS) standard for 140-2.

- About the Oracle Database FIPS 140-2 Settings
 Federal Information Processing Standards (FIPS) are standards and guidelines for federal computer systems that are developed by the U.S. National Institute of Standards and Technology (NIST).
- Configuring FIPS 140-2 for Transparent Data Encryption and DBMS_CRYPTO The DBFIPS 140 initialization parameter configures FIPS mode.
- Configuration of FIPS 140-2 for Transport Layer Security
 The SSLFIPS 140 parameter configures FIPS mode for Transport Layer Security (TLS).
- Configuration of FIPS 140-2 for Native Network Encryption
 You can configure FIPS 140-2 for native network encryption by setting a parameter in the sqlnet.ora file for both the server and the client.
- Postinstallation Checks for FIPS 140-2
 After you configure the FIPS 140-2 settings, you must verify permissions in the operating system.
- Verifying FIPS 140-2 Connections
 You can use trace files and other methods to verify the FIPS 140-2 connections.

E.1 About the Oracle Database FIPS 140-2 Settings

Federal Information Processing Standards (FIPS) are standards and guidelines for federal computer systems that are developed by the U.S. National Institute of Standards and Technology (NIST).

FIPS was developed in accordance with the Federal Information Security Management Act (FISMA). Although FIPS was developed for use by the federal government, many private sector entities voluntarily use these standards.

FIPS 140-2 specifies the security requirements that will be satisfied by a cryptographic module, providing four increasing, qualitative levels intended to cover a range of potential applications and environments. Security Level 1 conforms to the FIPS 140-2 algorithms, key sizes, integrity checks, and other requirements that are imposed by the regulations. FIPS 140-2 Security Level 1 requires no physical security mechanisms in the module beyond the requirement for production-grade equipment. As a result, this level allows software cryptographic functions to be performed in a general-purpose computer running on a specified operating environment.

When FIPS 140-2 settings are configured for the Oracle Database, the database uses FIPS 140-2 Level 1 validated cryptographic libraries to protect data at rest and in transit over the network. Oracle Database uses these cryptographic libraries for native network encryption, Transparent Data Encryption (TDE) of columns and tablespaces (including Oracle SecureFiles), Transport Layer Security (TLS), and the DBMS CRYPTO PL/SQL package.

Oracle Database currently uses Dell BSAFE, formerly known as RSA BSAFE, as the FIPS 140-2 level 1 validated cryptography library. To verify the current status of the FIPS certification, you can find information at the Computer Security Resource Center (CSRC) Web site address from the National Institute of Standards and Technology:

http://csrc.nist.gov/groups/STM/cmvp/validation.html

You can find information specific to FIPS by searching the validated cryptographic modules for vendor "RSA" and Module Name "BSAFE."

Note that Oracle Database FIPS settings enforce the use of FIPS-approved algorithms for the Oracle database only. Third-party vendor software used with Oracle Database running in FIPS mode must use only these FIPS-approved algorithms, or else the vendor software will encounter failures.

E.2 Configuring FIPS 140-2 for Transparent Data Encryption and DBMS CRYPTO

The DBFIPS 140 initialization parameter configures FIPS mode.

- 1. To configure Transparent Data Encryption and the DBMS_CRYPTO PL/SQL package program units to run in FIPS mode, set the DBFIPS 140 initialization parameter to TRUE.
 - The effect of this parameter depends on the platform.
- 2. Restart the database.

Table E-1 describes how the DBFIPS 140 parameter affects various platforms.

Table E-1 How the DBFIPS 140 Initialization Parameter Affects Platforms

Platform	Effect of Setting DBFIPS_140 to TRUE or FALSE
Linux or Windows on Intel x86_64	TRUE: TDE and DBMS_CRYPTO program units use Micro Edition Suite (MES) 4.6 FIPS mode, which uses RSA BSAFE Crypto-C Micro Edition (CCME) 4.1.5
	 FALSE: TDE and DBMS_CRYPTO program units use Intel Performance Primitives (IPP)
Solaris 11.1+ on either SPARC T-series or Intel x86_64	 TRUE: TDE and DBMS_CRYPTO program units use Micro Edition Suite (MES) 4.6 FIPS mode, which uses RSA BSAFE Crypto-C Micro Edition (CCME) 4.1.5 FALSE: TDE and DBMS_CRYPTO program units use Solaris
	Cryptographic Framework (SCF)/UCrypto (separately validated for FIPS 140)
Other operating systems or hardware	TRUE: TDE and DBMS_CRYPTO program units use MES 4.6 FIPS mode, which uses RSA BSAFE Crypto-C Micro Edition (CCME) 4.1.5 TRUE: TDE and RBMS_GRYPTO program units use MES 4.6 FIPS mode, which uses RSA BSAFE Crypto-C Micro Edition (CCME) 4.1.5
	FALSE: TDE and DBMS_CRYPTO program units use MES 4.6 non- FIPS mode

Be aware that setting <code>DBFIPS_140</code> to <code>TRUE</code> and thus using the underlying library in FIPS mode incurs a certain amount of overhead when the library is first loaded for each process. This is due to the verification of the signature and the execution of the self tests on the library. Once the library is loaded, then there is no other impact on performance.

Related Topics

Oracle Database Reference



E.3 Configuration of FIPS 140-2 for Transport Layer Security

The SSLFIPS 140 parameter configures FIPS mode for Transport Layer Security (TLS).

- Configuring the SSLFIPS_140 and SSLFIPS_LIB Parameters for Transport Layer Security
 To configure FIPS 140-2 for TLS, you must set the SSLFIPS_140 parameter. If you are
 using the Oracle Instant Client, then you must set the SSLFIPS_LIB parameter as well.
- Approved TLS Cipher Suites for FIPS 140-2
 A cipher suite is a set of authentication, encryption, and data integrity algorithms that exchange messages between network nodes.

E.3.1 Configuring the SSLFIPS_140 and SSLFIPS_LIB Parameters for Transport Layer Security

To configure FIPS 140-2 for TLS, you must set the <code>SSLFIPS_140</code> parameter. If you are using the Oracle Instant Client, then you must set the <code>SSLFIPS_LIB</code> parameter as well.

The SSLFIPS_140 parameter configures the Transport Layer Security (TLS) adapter to run in FIPS mode. SSLFIPS_LIB sets the location of the FIPS library.

- 1. Ensure that the fips.ora file is either located in the \$ORACLE_HOME/ldap/admin directory, or is in a location pointed to by the FIPS HOME environment variable.
- 2. In the fips.ora file, set the SSLFIPS 140 and SSLFIPS LIB parameters.
 - Set SSLFIPS_140 to TRUE so that the TLS adapter can run in FIPS mode. For example: SSLFIPS 140=TRUE

This parameter is FALSE by default.

• If you are using Oracle Instant Client, then set SSLFIPS_LIB to the location of the FIPS library. For example:

```
SSLFIPS LIB=$ORACLE HOME/lib
```

3. Repeat this procedure in any Oracle Database home for any database server or client.

When you set SSLFIPS_140 to TRUE, Transport Layer Security cryptographic operations take place in the embedded RSA/Micro Edition Suite (MES) library in FIPS mode. These cryptographic operations are accelerated by the CPU when hardware acceleration is available and properly configured in the host hardware and software.

If you set <code>SSLFIPS_140</code> to <code>FALSE</code>, then Transport Layer Security cryptographic operations take place in the embedded RSA/Micro Edition Suite (MES) library in *non-*FIPS mode, and as with the <code>TRUE</code> setting, the operations are accelerated if possible.



The SSLFIPS_140 parameter replaces the SQLNET.SSLFIPS_140 parameter used in Oracle Database 10g release 2 (10.2). You must set the parameter in the fips.ora file, and not the sqlnet.ora file.



E.3.2 Approved TLS Cipher Suites for FIPS 140-2

A cipher suite is a set of authentication, encryption, and data integrity algorithms that exchange messages between network nodes.

During a TLS handshake, for example, the two nodes negotiate to see as to which cipher suite they will use when transmitting messages back and forth.

Configuring Specific Cipher Suites

Oracle Database TLS cipher suites are automatically set to FIPS approved cipher suites. If you want to configure specific cipher suites, then you can do so by setting the <code>SSL_CIPHER_SUITES</code> parameter in the <code>sglnet.ora</code> or the <code>listener.ora</code> file.

```
SSL_CIPHER_SUITES=(SSL_cipher_suite1[,SSL_cipher_suite2[,..]])
```

You can also use Oracle Net Manager to set this parameter on the server and the client.

If a specific cipher suite is not specified, then Oracle Database will use the strongest cipher suite common to both the database server and client. The priority order of cipher suites to be selected are in order as they are listed in the preferred and less preferred cipher lists below. Oracle Database will not select 3DES cipher suites automatically due to their weakness; they must be configured explicitly.

Preferred Cipher Suites

The following cipher suites are approved for FIPS validation if you are using Transport Layer Security (TLS) version 1.2:

- TLS ECDHE ECDSA WITH AES 256 GCM SHA384
- TLS ECDHE ECDSA WITH AES 256 CBC SHA384
- TLS ECDHE_ECDSA_WITH_AES_128_GCM_SHA256
- TLS ECDHE ECDSA WITH AES 128 CBC SHA256
- TLS_ECDHE_RSA_WITH_AES_256_GCM_SHA384
- TLS ECDHE RSA WITH AES 256 CBC SHA384
- TLS_ECDHE_RSA_WITH_AES_128_GCM_SHA256
- TLS ECDHE RSA WITH AES 128 CBC SHA256

The following cipher suites are approved for FIPS validation if you are using Transport Layer Security (TLS) version 1, 1.1, or 1.2:

- TLS ECDHE ECDSA WITH AES 256 CBC SHA
- TLS ECDHE ECDSA WITH AES 128 CBC SHA
- TLS ECDHE RSA WITH AES 128 CBC SHA
- TLS ECDHE RSA WITH AES 256 CBC SHA

3DES-Based Cipher Suites

Oracle does not recommend 3DES-based cipher suites because of a weakness in their design. Oracle Database release 21c and later contains support for the following 3DES-based cipher suites. However, they are not enabled by default and must be explicitly configured through the SSL CIPHER SUITES parameter in the sqlnet.ora or the listener.ora file.



- TLS ECDHE ECDSA WITH 3DES EDE CBC SHA
- TLS_ECDHE_RSA_WITH_3DES_EDE_CBC_SHA
- TLS RSA WITH 3DES EDE CBC SHA

Related Topics

- Step 1C: Set the Transport Layer Security Cipher Suites on the Server (Optional) Optionally, you can set the Transport Layer Security cipher suites.
- Step 2D: Set the Client Transport Layer Security Cipher Suites (Optional)
 Optionally, you can set the Transport Layer Security cipher suites. Oracle Database provides default cipher suite settings.

E.4 Configuration of FIPS 140-2 for Native Network Encryption

You can configure FIPS 140-2 for native network encryption by setting a parameter in the sqlnet.ora file for both the server and the client.

- About Configuration of FIPS 140-2 for Native Network Encryption
 The configuration of FIPS 140-2 for native network encryption is similar to that of Transport Layer Security (TLS).
- Configuring the FIPS_140 Parameter for Native Network Encryption
 To configure FIPS 140-2 for native network encryption, you must set the FIPS_140 parameter in the sqlnet.ora file.

E.4.1 About Configuration of FIPS 140-2 for Native Network Encryption

The configuration of FIPS 140-2 for native network encryption is similar to that of Transport Layer Security (TLS).

For network native encryption, you enable FIPS mode by setting SSL_FIPS140 in the sqlnet.ora configuration file, instead of the SSL FIPS140 setting in fips.ora.

The algorithms that the FIPS library supports for native network encryption are as follows:

- Encryption: AES128, AES192, and AES256
- Checksumming: SHA1, SHA256, SHA384, and SHA512

E.4.2 Configuring the FIPS_140 Parameter for Native Network Encryption

To configure FIPS 140-2 for native network encryption, you must set the $FIPS_140$ parameter in the sqlnet.ora file.

The FIPS_140 parameter configures the native network encryption adapter to run in FIPS mode.

- 1. Locate the sqlnet.ora file that is used by the database client or database server
- 2. Add the following line to the sqlnet.ora file:

```
SQLNET.FIPS 140=TRUE
```

3. Repeat this procedure in any Oracle Database home for any database server or client.



When FIPS_140 is set to TRUE, native network encryption cryptographic operations take place in the embedded BSAFE Micro Edition Suite (MES) library in FIPS mode. These cryptographic operations are accelerated by the CPU when hardware acceleration is available and properly configured in the host hardware and software.

E.5 Postinstallation Checks for FIPS 140-2

After you configure the FIPS 140-2 settings, you must verify permissions in the operating system.

The permissions are as follows:

- Set execute permissions on all Oracle executable files to prevent the execution of Oracle Cryptographic Libraries by users who are unauthorized to do so, in accordance with the system security policy.
- Set read and write permissions on all Oracle executable files to prevent accidental or deliberate reading or modification of Oracle Cryptographic Libraries by any user.

To comply with FIPS 140-2 Level 2 requirements, in the security policy, include procedures to prevent unauthorized users from reading, modifying or executing Oracle Cryptographic Libraries processes and the memory they are using in the operating system.

E.6 Verifying FIPS 140-2 Connections

You can use trace files and other methods to verify the FIPS 140-2 connections.

- Verifying FIPS 140-2 Connections for Transport Layer Security
 You can use trace files to check the FIPS 140-2 connections for Transport Layer Security
 (TLS).
- Verifying FIPS 140-2 Connections for Network Native Encryption
 You can use trace files to check the FIPS 140-2 connections for network native encryption.
- Verifying FIPS 140-2 Connections for Transparent Data Encryption and DBMS_CRYPTO You can check if FIPS mode is enabled by using SQL*Plus.

E.6.1 Verifying FIPS 140-2 Connections for Transport Layer Security

You can use trace files to check the FIPS 140-2 connections for Transport Layer Security (TLS).

1. Add the following lines to sqlnet.ora to enable tracing:

```
trace_directory_server=trace_directory
trace_file_server=trace_file
trace_level_server=trace_level
```

For example:

```
trace_directory=/private/oracle/owm
trace_file_server=fips_trace.trc
trace level server=16
```

Trace level 16 is the minimum trace level required to check the results of the FIPS self-tests.

2. Check the trace files by searching for Provider Type: FIPS140.

E.6.2 Verifying FIPS 140-2 Connections for Network Native Encryption

You can use trace files to check the FIPS 140-2 connections for network native encryption.

1. Add the following lines to sqlnet.ora to enable tracing:

```
trace_directory_server=trace_directory
trace_file_server=trace_file
trace_level_server=trace_level
```

For example:

```
trace_directory=/private/oracle/owm
trace_file_server=fips_trace.trc
trace_level_server=16
```

Trace level 16 is the minimum trace level required to check the results of the FIPS self-tests.

2. Check the trace files by searching for FIPS DAC check succeeded.

E.6.3 Verifying FIPS 140-2 Connections for Transparent Data Encryption and DBMS_CRYPTO

You can check if FIPS mode is enabled by using SQL*Plus.

- Connect to the database instance by using SQL*Plus.
- Execute the following SHOW PARAMETER command:

```
SHOW PARAMETER DBFIPS 140
```

Output similar to the following should appear:

NAME	TYPE	VALUE
DBFIPS_140	boolean	TRUE



F

Managing Public Key Infrastructure (PKI) Elements

You can use the <code>orapki</code> command line utility and sqlnet.ora parameters to manage public key infrastructure (PKI) elements.

Uses of the orapki Utility

The orapki utility manages public key infrastructure (PKI) elements, such as wallets and certificate revocation lists, from the command line.

orapki Utility Syntax

The orapki utility syntax specifies an Oracle wallet, a certificate revocation list, or a PKI digital certificate.

Creating Signed Certificates for Testing Purposes

The orapki utility provides a convenient, lightweight way to create signed certificates for testing purposes.

Viewing a Certificate

After you create a certificate, you can use the orapki utility to view it.

 Importing a User-Supplied or Trusted Certificate into an Oracle Wallet You can add a user-supplied or trusted certificate to an Oracle wallet.

Controlling MD5 and SHA-1 Certificate Use

You can use the sqlnet.ora file to control whether MD5 and SHA-1 signed certificates are accepted.

Managing Oracle Wallets with orapki Utility

The orapki utility can create, view, modify wallets; it can add and export certificates and certificate requests.

Management of Certificate Revocation Lists (CRLs) with orapki Utility

You must manage certificate revocation lists (CRLs) with the orapki utility.

orapki Usage

Examples of orapki commands include creating wallets, user certificates, and wallets with self-signed certificates, and exporting certificates.

orapki Utility Commands Summary

The orapki commands perform a variety of wallet, certificate revocation lists (CRL), and certificate management tasks.

F.1 Uses of the orapki Utility

The <code>orapki</code> utility manages public key infrastructure (PKI) elements, such as wallets and certificate revocation lists, from the command line.

This way, you can automate these tasks by using scripts. Providing a way to incorporate the management of PKI elements into scripts makes it possible to automate many of the routine tasks of maintaining a PKI.

You can use the orapki command-line utility to perform the following tasks:

- Creating and viewing signed certificates for testing purposes
- Manage Oracle wallets (except for Transparent Data Encryption keystores):
 - Create and display Oracle wallets
 - Add and remove certificate requests
 - Add and remove certificates
 - Add and remove trusted certificates
- Manage certificate revocation lists (CRLs):
 - Renaming CRLs with a hash value for certificate validation
 - Uploading, listing, viewing, and deleting CRLs in Oracle Internet Directory



The use of PKI encryption with Transparent Data Encryption is deprecated. To configure Transparent Data Encryption, use the ADMINISTER KEY MANAGEMENT SQL statement. See *Oracle Database Advanced Security Guide* for more information.

F.2 orapki Utility Syntax

The orapki utility syntax specifies an Oracle wallet, a certificate revocation list, or a PKI digital certificate.

The syntax of the orapki command-line utility is as follows:

```
orapki module command -parameter value
```

In this specification, <code>module</code> can be <code>wallet</code> (Oracle wallet), <code>crl</code> (certificate revocation list), or <code>cert</code> (PKI digital certificate). The available commands depend on the <code>module</code> you are using.

For example, if you are working with a wallet, then you can add a certificate or a key to the wallet with the add command. The following example adds the user certificate located at / private/lhale/cert.txt to the wallet located at \$ORACLE HOME/wallet.p12:

orapki wallet add -wallet \$ORACLE_HOME/wallet/ewallet.p12 -user_cert -cert /private/ lhale/cert.txt

F.3 Creating Signed Certificates for Testing Purposes

The orapki utility provides a convenient, lightweight way to create signed certificates for testing purposes.

To create a signed certificate for testing purposes, use the following command:

```
orapki cert create [-wallet wallet_location] -request certificate_request_location -cert certificate_location -validity number_of_days [-summary]
```

This command creates a signed certificate from the certificate request. The <code>-wallet</code> parameter specifies the wallet containing the user certificate and private key that will be used to sign the certificate request. The <code>-validity</code> parameter specifies the number of days, starting from the current date, that this certificate will be valid. Specifying a certificate and certificate request is mandatory for this command.



F.4 Viewing a Certificate

After you create a certificate, you can use the \mathtt{orapki} utility to view it.

To view a certificate, use the following command:

```
orapki cert display -cert certificate location [-summary | -complete]
```

This command enables you to view a test certificate that you have created with <code>orapki</code>. You can choose either <code>-summary</code> or <code>-complete</code>, which determines how much detail the command will display. If you choose <code>-summary</code>, the command will display the certificate and its expiration date. If you choose <code>-complete</code>, it will display additional certificate information, including the serial number and public key.

F.5 Importing a User-Supplied or Trusted Certificate into an Oracle Wallet

You can add a user-supplied or trusted certificate to an Oracle wallet.

 To add a trusted certificate to an Oracle wallet, orapki wallet add with the trusted cert parameter.

```
orapki wallet add -wallet wallet_location [-pwd wallet_password] -trusted_cert -cert root and/or intermediate certificate file
```

 To add a user-created certificate to an Oracle wallet, use orapki wallet add with the user cert parameter.

```
orapki wallet add -wallet wallet_location [-pwd wallet_password] -user_cert -cert user certificate file
```

F.6 Controlling MD5 and SHA-1 Certificate Use

You can use the sqlnet.ora file to control whether MD5 and SHA-1 signed certificates are accepted.

To control whether the MD5 and SHA-1 signed certificates are accepted, you can edit the sqlnet.ora file to enable or disable their use.



MD5 is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

- 1. Log in to the server where the Oracle database resides.
- 2. Edit the sqlnet.ora file.

By default, the sqlnet.ora file is located in the <code>\$ORACLE_HOME/dbs</code> directory or in the location set by the <code>TNS ADMIN</code> environment variable.

3. Set the following parameters:



- ACCEPT_MD5_CERTS controls the use of MD5 certificates. The default is FALSE. This
 parameter replaces the ORACLE_SSL_ALLOW_MD5_CERT_SIGNATURES environment
 variable.
- ACCEPT SHA1 CERTS controls the use of SHA-1 certificates. The default is TRUE.

F.7 Managing Oracle Wallets with orapki Utility

The orapki utility can create, view, modify wallets; it can add and export certificates and certificate requests.

- About Managing Wallets with orapki
 You should understand the orapki command-line utility syntax used to create and manage
 Oracle wallets.
- Creating, Viewing, and Modifying Wallets with orapki You can use orapki to perform a range of management activities with Oracle wallets.
- Adding Certificates and Certificate Requests to Oracle Wallets with orapki
 You can use the orapki utility to perform a range of certificate-related tasks.
- Exporting Certificates and Certificate Requests from Oracle Wallets with orapki
 You can use the orapki utility to export certificates and certificate requests from Oracle wallets.

F.7.1 About Managing Wallets with orapki

You should understand the orapki command-line utility syntax used to create and manage Oracle wallets.

You can use the <code>orapki</code> utility <code>wallet</code> module commands in scripts to automate the wallet creation process. For example, you can create PKCS#12 wallets and auto-login wallets. You can create auto-login wallets that are associated with PKCS#12 wallets or auto-login wallets that are local to the computer on which they were created and the user who created them. You can view wallets, modify wallet passwords, and convert wallets to use the AES256 algorithm.



The -wallet parameter is mandatory for all wallet module commands.

F.7.2 Creating, Viewing, and Modifying Wallets with orapki

You can use orapki to perform a range of management activities with Oracle wallets.

- Creating a PKCS#12 Wallet
 You can use the orapki utility to create a PKCS#12 Oracle wallet.
- Using OpenSSL to Create a PKCS#12 Wallet That Has a Certificate Chain
 You can use the OpenSSL utility to create an PKCS#12 wallet so that it has a certificate
 chain.
- Creating an Auto-Login Wallet
 You can use the orapki utility to create an auto-login wallet.

- Creating an Auto-Login Wallet That Is Associated with a PKCS#12 Wallet You can create an auto-login wallet that is associated with a PKCS#12 wallet.
- Creating an Auto-Login Wallet That Is Local to the Computer and User Who Created It
 The orapki utility can create an auto-login wallet that is local to the computer of the user
 who created it.
- Viewing a Wallet

You can use the orapki utility to view a wallet.

- Modifying the Password for a Wallet
 You can use the orapki utility to modify the password of a wallet.
- Converting an Oracle Wallet to Use the AES256 Algorithm
 By default, an Oracle wallet with the ADMINISTER KEY MANAGEMENT OF ALTER SYSTEM statement is encrypted with 3DES.

F.7.2.1 Creating a PKCS#12 Wallet

You can use the orapki utility to create a PKCS#12 Oracle wallet.

 To create an Oracle PKCS#12 wallet (ewallet.p12), use the orapki wallet create command.

```
orapki wallet create -wallet wallet location [-pwd password]
```

This command prompts you to enter and reenter a wallet password, if no password has been specified on the command line. It creates a wallet in the location specified for -wallet.



The default algorithm is AES256. For security reasons, Oracle recommends that you do not specify the password at the command line. You should supply the password only when prompted to do so.

F.7.2.2 Using OpenSSL to Create a PKCS#12 Wallet That Has a Certificate Chain

You can use the OpenSSL utility to create an PKCS#12 wallet so that it has a certificate chain.

1. Concatenate the CA certificate chain.

For example:

```
cat root.cer sub.cer > ca.certs
```

In this example, root.cer is the root certificate.

2. Run openss1 to create the PKCS#12 wallet.

```
openssl pkcs12 -export -nodes -certfile ca.certs -inkey user.key -in user.cer
```

3. When prompted, provide a password for the wallet. Ensure that this password at at least 8 characters with an alphnumeric character mix.

For more information about how OpenSSL works with PKCS#12 wallets, visit https://www.openssl.org/ and search for "pkcs12".

F.7.2.3 Creating an Auto-Login Wallet

You can use the orapki utility to create an auto-login wallet.

• To create an auto-login wallet (cwallet.sso), which does not need a password to open the wallet, use the orapki wallet create command.

```
orapki wallet create -wallet wallet location -auto login only
```

You can modify or delete the wallet without using a password. File system permissions provide the necessary security for such auto-login wallets.

You cannot move local auto-login wallets to another computer. They must be used on the host on which they are created.

Even though a local auto-login wallet does not need a password to open, you must supply the password for the associated PKCS#12 wallet in order to modify or delete the wallet. Any update to the PKCS#12 wallet also updates the associated auto-login wallet.

F.7.2.4 Creating an Auto-Login Wallet That Is Associated with a PKCS#12 Wallet

You can create an auto-login wallet that is associated with a PKCS#12 wallet.

The auto-login wallet does not need a password to open.

However, you must supply the password for the associated PKCS#12 wallet in order to modify or delete the wallet. Any update to the PKCS#12 wallet also updates the associated auto-login wallet.

• To create an auto-login wallet (cwallet.sso) that is associated with a PKCS#12 wallet (ewallet.p12), use the orapki wallet create command.

```
orapki wallet create -wallet wallet location -auto login [-pwd wallet password]
```

This command creates a wallet with auto-login enabled (cwallet.sso) and associates it with a PKCS#12 wallet (ewallet.p12). The command prompts you to enter the password for the PKCS#12 wallet, if no password has been specified at the command line.

If the wallet_location already contains a PKCS#12 wallet, then auto-login is enabled for it. You must supply the password for the existing PKCS#12 wallet in order to enable auto-login for it.

If the wallet_location does not contain a PKCS#12 wallet, then a new PKCS#12 wallet is created. You must specify a password for the new PKCS#12 wallet.

If you want to turn the auto-login feature off for a PKCS#12 wallet, then use Oracle Wallet Manager.



Oracle Database Enterprise User Security Administrator's Guide for more information



F.7.2.5 Creating an Auto-Login Wallet That Is Local to the Computer and User Who Created It

The orapki utility can create an auto-login wallet that is local to the computer of the user who created it.

 To create a local auto-login wallet that is local to both the computer on which it is created and the user who created it, use the following command:

```
orapki wallet create -wallet wallet location -auto login local [-pwd wallet password]
```

This command creates an auto-login wallet (cwallet.sso). It associates it with a PKCS#12 wallet (ewallet.p12). The command prompts you to enter the password for the PKCS#12 wallet, if no password has been specified at the command line.

F.7.2.6 Viewing a Wallet

You can use the orapki utility to view a wallet.

To view an Oracle wallet, use the orapki wallet display command.

```
orapki wallet display -wallet wallet location
```

This command displays the certificate requests, user certificates, and trusted certificates contained in the wallet, which must be a binary PKCS12 file, with extension .p12. Other files will fail.

F.7.2.7 Modifying the Password for a Wallet

You can use the orapki utility to modify the password of a wallet.

• To change the wallet password, use the orapki wallet change pwd command.

```
orapki wallet change_pwd -wallet wallet_location [-oldpwd wallet_password] [-newpwd
wallet password]
```

This command changes the current wallet password to the new password. The command prompts you for the old and new passwords if no password is supplied at the command line.



For security reasons, Oracle recommends that you do not specify the password options at the command line. You should supply the password when prompted to do so.

F.7.2.8 Converting an Oracle Wallet to Use the AES256 Algorithm

By default, an Oracle wallet with the ADMINISTER KEY MANAGEMENT or ALTER SYSTEM statement is encrypted with 3DES.

Starting with Oracle Database release 19.22, the default is AES256, not 3DES.

For FIPS 140-2 compliance, Oracle recommends that you change the wallet encryption algorithm 3DES to AES256. You can use the <code>orapki convert</code> command to convert the wallet to use the AES256 algorithm, which is stronger than the 3DES algorithm. Note that if you had

created the wallet using ADMINISTER KEY MANAGEMENT or ALTER SYSTEM statement, then by default it uses the AES256 algorithm.



The 3DES112 and 3DES168 algorithms are deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

 To change the wallet algorithm from 3DES to AES256, use the orapki wallet convert command.

```
orapki wallet convert -wallet wallet location [-pwd wallet password] [-compat v12]
```

The compat v12 setting performs the conversion from 3DES to AES256.

F.7.3 Adding Certificates and Certificate Requests to Oracle Wallets with orapki

You can use the orapki utiltiy to perform a range of certificate-related tasks.

- Adding a Certificate Request to an Oracle Wallet
 You can use the orapki utility to add certificates and certificate requests to Oracle wallets.
- Adding a Trusted Certificate to an Oracle Wallet
 You can use the orapki utility to add trusted certificates to an Oracle wallet.
- Adding a Root Certificate to an Oracle Wallet
 You can use the orapki utility to add a root certificate to an Oracle wallet.
- Adding a User Certificate to an Oracle Wallet
 You can use the orapki utility to add a user certificate to an Oracle wallet.
- Verifying Credentials on the Hardware Device That Uses a PKCS#11 Wallet You can verify credentials on the hardware device using the PKCS#11 wallet.
- Adding PKCS#11 Information to an Oracle Wallet
 A wallet that contains PKCS#11 information can be used like any Oracle wallet.

F.7.3.1 Adding a Certificate Request to an Oracle Wallet

You can use the orapki utility to add certificates and certificate requests to Oracle wallets.

To add a certificate request to an Oracle wallet, use the orapki wallet add command.

```
orapki wallet add -wallet wallet location -dn user dn -keySize 512|1024|2048
```

This command adds a certificate request to a wallet for the user with the specified distinguished name (user_dn). The request also specifies the requested certificate's key size (512, 1024, or 2048 bits). To sign the request, export it with the export option.

Related Topics

Exporting Certificates and Certificate Requests from Oracle Wallets with orapki
 You can use the orapki utility to export certificates and certificate requests from Oracle wallets.



F.7.3.2 Adding a Trusted Certificate to an Oracle Wallet

You can use the orapki utility to add trusted certificates to an Oracle wallet.

• To add a trusted certificate to an Oracle wallet, use the orapki wallet add command.

```
orapki wallet add -wallet wallet location -trusted cert -cert certificate location
```

This command adds a trusted certificate, at the specified location (-cert certificate_location), to a wallet. You must add all trusted certificates in the certificate chain of a user certificate before adding a user certificate, or the command to add the user certificate will fail.

F.7.3.3 Adding a Root Certificate to an Oracle Wallet

You can use the orapki utility to add a root certificate to an Oracle wallet.

To add a root certificate to an Oracle wallet, use the orapki wallet add command.

```
orapki wallet add -wallet wallet\_location -dn certificate\_dn -keySize 512 \mid 1024 \mid 2048 -self signed -validity number of days
```

This command creates a new self-signed (root) certificate and adds it to the wallet. The -validity parameter (mandatory) specifies the number of days, starting from the current date, that this certificate will be valid. You can specify a key size for this root certificate (-keySize) of 512, 1024, or 2048 bits.

F.7.3.4 Adding a User Certificate to an Oracle Wallet

You can use the orapki utility to add a user certificate to an Oracle wallet.

• To add a user certificate to an Oracle wallet, use the orapki wallet add command.

```
orapki wallet add -wallet wallet_location -user_cert -cert certificate_location
```

This command adds the user certificate at the location specified with the <code>-cert</code> parameter to the Oracle wallet at the <code>wallet_location</code>. Before you add a user certificate to a wallet, you must add all the trusted certificates that make up the certificate chain. If all trusted certificates are not installed in the wallet before you add the user certificate, then adding the user certificate will fail.



For security reasons, Oracle recommends that you do not specify the password at the command line. You should supply the password when prompted to do so.

F.7.3.5 Verifying Credentials on the Hardware Device That Uses a PKCS#11 Wallet

You can verify credentials on the hardware device using the PKCS#11 wallet.

To verify the credential details, use the orapki wallet p11 verify command.

```
orapki wallet p11_verify -wallet wallet_location [-pwd wallet_password]
```



F.7.3.6 Adding PKCS#11 Information to an Oracle Wallet

A wallet that contains PKCS#11 information can be used like any Oracle wallet.

The private keys are stored on a hardware device. The cryptographic operations are also performed on the device.

To add PKCS#11 information to a wallet, use the orapki wallet p11 add command.

```
orapki wallet p11_add -wallet wallet_location -p11_lib pkcs11Lib
[-p11_tokenlabel tokenLabel] [-p11_tokenpw tokenPassphrase]
[-p11 certlabel certLabel] [-pwd wallet password]
```

In this specification:

- wallet specifies the wallet location.
- p11 lib specifies the path to the PKCS#11 library. This includes the library filename.
- p11_tokenlabel specifies the token or smart card used on the device. Use this when there are multiple tokens on the device. Token labels are set using vendor tools.
- p11_tokenpw specifies the password that is used to access the token. Token passwords
 are set using vendor tools.
- p11_certlabel is used to specify a certificate label on the token. Use this when a token contains multiple certificates. Certificate labels are set using vendor tools.
- pwd is used to specify the wallet password.

F.7.4 Exporting Certificates and Certificate Requests from Oracle Wallets with orapki

You can use the orapki utility to export certificates and certificate requests from Oracle wallets.

To export a certificate from an Oracle wallet, use the orapki wallet export command.

```
orapki wallet export -wallet wallet_location -dn certificate_dn -cert certificate_filename
```

This command exports a certificate with the subject's distinguished name (-dn) from a wallet to a file that is specified by -cert.

To export a certificate request from an Oracle wallet, use the following command:

```
orapki wallet export -wallet wallet\_location -dn certificate\_request\_dn -request certificate request filename
```

This command exports a certificate request with the subject's distinguished name (-dn) from a wallet to a file that is specified by -request.

F.8 Management of Certificate Revocation Lists (CRLs) with orapki Utility

You must manage certificate revocation lists (CRLs) with the orapki utility.



This utility creates a hashed value of the CRL issuer's name to identify the CRLs location in your system. If you do not use <code>orapki</code>, your Oracle server cannot locate CRLs to validate PKI digital certificates.

Related Topics

Certificate Revocation List Management

Certificate revocation list management entails ensuring that the CRLs are the correct format before you enable certificate revocation checking.

F.9 orapki Usage

Examples of orapki commands include creating wallets, user certificates, and wallets with self-signed certificates, and exporting certificates.

- Example: Wallet with a Self-Signed Certificate and Export of the Certificate
 The orapki wallet add command can create a wallet with a self-signed certificate; the orapki wallet export can export the certificate.
- Example: Creating a Wallet and a User Certificate
 The orapki utility can create wallets and user certificates.

F.9.1 Example: Wallet with a Self-Signed Certificate and Export of the Certificate

The orapki wallet add command can create a wallet with a self-signed certificate; the orapki wallet export can export the certificate.

Example F-1 illustrates the steps to create a wallet with a self-signed certificate, view the wallet, and then export the certificate to a file.

Example F-1 Creating a Wallet with a Self-Signed Certificate and Exporting the Certificate

Create a wallet.

For example:

```
orapki wallet create -wallet /private/user/orapki_use/root
```

The wallet is created at the location, /private/user/orapki use/root.

2. Add a self-signed certificate to the wallet.

```
orapki wallet add -wallet /private/user/orapki_use/root -dn 'CN=root test,C=US' -keysize 2048 -self signed -validity 3650
```

This creates a self-signed certificate with a validity of 3650 days. The distinguished name of the subject is <code>CN=root_test</code>, <code>C=US</code>. The key size for the certificate is 2048 bits.

3. View the wallet.

```
orapki wallet display -wallet /private/user/orapki_use/root
```

This is used to view the certificate contained in the wallet.

4. Export the certificate.

```
orapki wallet export -wallet /private/user/orapki_use/root -dn
'CN=root test,C=US' -cert /private/user/orapki use/root/b64certificate.txt
```



This exports the self-signed certificate to the file, b64certificate.txt. Note that the distinguished name used is the same as in step 2.

F.9.2 Example: Creating a Wallet and a User Certificate

The orapki utility can create wallets and user certificates.

Example F-2 illustrates miscellaneous tasks related to creating user certificates.

The following steps illustrate creating a wallet, creating a certificate request, exporting the certificate request, creating a signed certificate from the request for testing, viewing the certificate, adding a trusted certificate to the wallet and adding a user certificate to the wallet.

Example F-2 Creating a Wallet and a User Certificate

1. Create a wallet with auto-login enabled.

For exmaple:

```
orapki wallet create -wallet /private/user/orapki use/server -auto login
```

This creates a wallet at /private/user/orapki use/server with auto-login enabled.

2. Add a certificate request to the wallet.

```
orapki wallet add -wallet /private/user/orapki_use/server/ewallet.p12 -dn 'CN=server test,C=US' -keysize 2048
```

This adds a certificate request to the wallet that was created (ewallet.p12). The distinguished name of the subject is CN=server_test, C=US. The key size specified is 2048 bits.

Export the certificate request to a file.

```
orapki wallet export -wallet /private/user/orapki_use/server -dn
'CN=server_test,C=US' -request /private/user/orapki_use/server/creq.txt
```

This exports the certificate request to the specified file, which is creq.txt in this case.

Create a signed certificate from the request for test purposes.

```
orapki cert create -wallet /private/user/orapki_use/root -request /private/user/orapki_use/server/creq.txt -cert /private/user/orapki_use/server/cert.txt -validity 3650
```

This creates a certificate, cert.txt with a validity of 3650 days. The certificate is created from the certificate request generated in the preceding step.

View the certificate.

```
orapki cert display -cert /private/user/orapki use/server/cert.txt -complete
```

This displays the certificate generated in the preceding step. The -complete option enables you to display additional certificate information, including the serial number and public key.

6. Add a trusted certificate to the wallet.

```
orapki wallet add -wallet /private/user/orapki_use/server/ewallet.p12 -trusted_cert -cert /private/user/orapki_use/root/b64certificate.txt
```



This adds a trusted certificate, b64certificate.txt to the ewallet.p12 wallet. You must add all trusted certificates in the certificate chain of a user certificate before adding a user certificate.

Add a user certificate to the wallet.

orapki wallet add -wallet /private/user/orapki_use/server/ewallet.p12 -user_cert cert /private/user/orapki use/server/cert.txt

This command adds the user certificate, cert.txt to the ewallet.p12 wallet.

F.10 orapki Utility Commands Summary

The orapki commands perform a variety of wallet, certificate revocation lists (CRL), and certificate management tasks.

· orapki cert create

The orapki cert create command creates a signed certificate for testing purposes.

orapki cert display

The orapki cert display command displays details of a specific certificate.

orapki crl delete Command

The orapki crl delete command deletes a certificate revocation list (CRL) from Oracle Internet Directory.

orapki crl display

The orapki crl display command displays a specified certificate revocation list (CRL) that is stored in Oracle Internet Directory.

orapki crl hash

The orapki crl hash command generates a hash value of the certificate revocation list (CRL) issuer to identify the CRL file system location for certificate validation.

orapki crl list

The orapki crl list command displays a list of certificate revocation lists (CRLs) stored in Oracle Internet Directory.

orapki crl upload

The orapki crl upload command uploads a certificate revocation list (CRL) to the CRL subtree in Oracle Internet Directory.

orapki wallet add

The orapki wallet add command adds certificate requests and certificates to an Oracle wallet.

orapki wallet convert

The orapki wallet convert command changes the Oracle wallet's encryption algorithm from 3DES to AES256.

orapki wallet create

The orapki wallet create command creates an Oracle wallet or enables auto-login for an Oracle wallet.

orapki wallet display

The orapki wallet display command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

orapki wallet export

The orapki wallet export command exports certificate requests and certificates from an Oracle wallet.

F.10.1 orapki cert create

The orapki cert create command creates a signed certificate for testing purposes.

Syntax

orapki cert create [-wallet wallet_location] -request certificate_request_location -cert
certificate location -validity number of days [-summary]

- wallet specifies the wallet containing the user certificate and private key that will be used to sign the certificate request.
- request (mandatory) specifies the location of the certificate request for the certificate you are creating.
- cert (mandatory) specifies the directory location where the tool places the new signed certificate.
- validity (mandatory) specifies the number of days, starting from the current date, that this
 certificate will be valid.

F.10.2 orapki cert display

The orapki cert display command displays details of a specific certificate.

Syntax

orapki cert display -cert certificate_location [-summary|-complete]

- cert specifies the location of the certificate you want to display.
- You can use either the -summary or the -complete parameter to display the following information:
 - summary displays the certificate and its expiration date
 - complete displays additional certificate information, including the serial number and public key

F.10.3 orapki crl delete Command

The orapki crl delete command deletes a certificate revocation list (CRL) from Oracle Internet Directory.

The user who deletes the CRLs from the directory by using orapki must be a member of the CRLAdmins (cn=CRLAdmins, cn=groups, %s OracleContextDN%) directory group.

Prerequisites

None

Syntax

orapki crl delete -issuer issuer_name -ldap hostname:ssl_port -user username [-wallet wallet location] [-summary]

issuer specifies the name of the certificate authority (CA) who issued the CRL.

- ldap specifies the host name and SSL port for the directory where the CRLs are to be deleted. Note that this must be a directory SSL port with no authentication.
 - See also Uploading CRLs to Oracle Internet Directory for more information about this port.
- user specifies the user name of the directory user who has permission to delete CRLs from the CRL subtree in the directory.
- wallet (optional) specifies the location of the wallet that contains the certificate of the
 certificate authority (CA) who issued the CRL. Using it causes the tool to verify the validity
 of the CRL against the CA's certificate prior to deleting it from the directory.
- summary is optional. It displays the CRL LDAP entry that was deleted.

F.10.4 orapki crl display

The orapki crl display command displays a specified certificate revocation list (CRL) that is stored in Oracle Internet Directory.

Syntax

```
orapki crl display -crl crl location [-wallet wallet location] [-summary|-complete]
```

- crl parameter specifies the location of the CRL in the directory. It is convenient to paste the CRL location from the list that displays when you use the orapki crl list command. See orapki crl list.
- wallet (optional) specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to displaying it.
- summary and complete display the following information:
 - summary provides a listing that contains the CRL issuer's name and the CRL's validity period
 - complete provides a list of all revoked certificates that the CRL contains. Note that this
 option may take a long time to display, depending on the size of the CRL.

F.10.5 orapki crl hash

The orapki crl hash command generates a hash value of the certificate revocation list (CRL) issuer to identify the CRL file system location for certificate validation.

Syntax

```
orapki crl hash -crl crl_filename|URL [-wallet wallet_location] [-symlink|-copy]
crl_directory [-summary]
```

- crl specifies the filename that contains the CRL or the URL where it can be found.
- wallet (optional) specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to uploading it to the directory.
- Depending on the operating system, use either the -symlink or the -copy parameter:
 - (UNIX) symlink creates a symbolic link to the CRL at the crl directory location
 - (Windows) copy creates a copy of the CRL at the crl directory location
- summary (optional) displays the CRL issuer's name.



F.10.6 orapki crl list

The orapki crl list command displays a list of certificate revocation lists (CRLs) stored in Oracle Internet Directory.

Syntax

This is useful for browsing to locate a particular CRL to view or download to your local file system.

```
orapki crl list -ldap hostname:ssl port
```

ldap specifies the host name and SSL port for the directory server from where you want to list CRLs. Note that this must be a directory SSL port with no authentication.



Uploading CRLs to Oracle Internet Directory for more information about this port

F.10.7 orapki crl upload

The orapki crl upload command uploads a certificate revocation list (CRL) to the CRL subtree in Oracle Internet Directory.

Note that you must be a member of the directory administrative group CRLAdmins (cn=CRLAdmins, cn=groups, %s_OracleContextDN%) to upload CRLs to the directory.

Syntax

orapki crl upload -crl crl_location -ldap hostname:ssl_port -user username [-wallet wallet location] [-summary]

- crl specifies the directory location or the URL where the CRL is located that you are uploading to the directory.
- ldap specifies the host name and SSL port for the directory where you are uploading the CRLs. Note that this must be a directory SSL port with no authentication.

See also Uploading CRLs to Oracle Internet Directory for more information about this port.

- user specifies the user name of the directory user who has permission to add CRLs to the CRL subtree in the directory.
- wallet specifies the location of the wallet that contains the certificate of the certificate authority (CA) who issued the CRL. This is an optional parameter. Using it causes the tool to verify the validity of the CRL against the CA's certificate prior to uploading it to the directory.
- summary is optional. It displays the CRL issuer's name and the LDAP entry where the CRL is stored in the directory.



F.10.8 orapki wallet add

The orapki wallet add command adds certificate requests and certificates to an Oracle wallet.

Syntax

To add certificate requests:

orapki wallet add -wallet wallet location -dn user dn -keySize 512|1024|2048

- wallet specifies the location of the wallet to which you want to add a certificate request.
- dn specifies the distinguished name of the certificate owner.
- keySize specifies the key size for the certificate.
- To sign the request, export it with the export option. Refer to orapki wallet export

To add trusted certificates:

```
orapki wallet add -wallet wallet location -trusted cert -cert certificate location
```

 trusted_cert adds the trusted certificate, at the location specified with -cert, to the wallet.

To add root certificates:

```
orapki wallet add -wallet wallet\_location -dn certificate\_dn -keySize 512 \mid 1024 \mid 2048 -self signed -validity number of days
```

- self_signed creates a root certificate.
- validity is mandatory. Use it to specify the number of days, starting from the current date, that this root certificate will be valid.

To add user certificates:

```
orapki wallet add -wallet wallet_location -user_cert -cert certificate_location
```

user_cert adds the user certificate at the location specified with the -cert parameter to
the wallet. Before you add a user certificate to a wallet, you must add all the trusted
certificates that make up the certificate chain. If all trusted certificates are not installed in
the wallet before you add the user certificate, then adding the user certificate will fail.

F.10.9 orapki wallet convert

The orapki wallet convert command changes the Oracle wallet's encryption algorithm from 3DES to AES256.

Syntax

```
orapki wallet convert -wallet wallet location [-pwd wallet password] [-compat v12]
```

- wallet specifies a location for the new wallet or the location of the wallet for which you want to turn on auto-login.
- pwd is the wallet password.
- compat v12 performs the conversion from 3DES to AES256.



F.10.10 orapki wallet create

The orapki wallet create command creates an Oracle wallet or enables auto-login for an Oracle wallet.

Syntax

```
orapki wallet create -wallet wallet location [-auto login|-auto login local]
```

The default algorithm is AES256.

- wallet specifies a location for the new wallet or the location of the wallet for which you want to turn on auto-login.
- auto_login creates an auto-login wallet, or it turns on automatic login for the wallet specified with the -wallet option.

See also *Oracle Database Enterprise User Security Administrator's Guide* for details about auto-login wallet.

• auto_login_local creates a local auto-login wallet, or it turns on local automatic login for the wallet specified with the -wallet option.

F.10.11 orapki wallet display

The orapki wallet display command displays the certificate requests, user certificates, and trusted certificates in an Oracle wallet.

The orapki wallet display command is a superset of the information that is shown in the orapki secretstore list_entries command. orapki wallet display shows everything, including the secret store entries' thumbprint. It inloudes both the SHA-1 and SHA-256 thumbprint information for a private key. These thumbprints select a particular certificate from the wallet and are displayed when you run the orapki wallet display command. You can specify an alias when you store a private key. The alias and thumbprint enable you to specify the exact private key to use with the connect string.

Syntax

```
orapki wallet display [-wallet [wallet_file_directory]] [-summary | [-complete | -
complete -details]]
[-pwd wallet_password] [-ssvs]
```

- wallet specifies a location for the wallet you want to open if it is not located in the current working directory.
- summary displays a summary of the wallet information; complete displays more details.
- ssvs displays the version of the wallet.
- details displays additional attributes such as version, signature algorithm, subject public key information, and extensions, as follows:
 - summary is the subject name.
 - complete Contains Alias, Subject, Issuer, Not Before, Not After, Serial Number,
 Key Length, MD5 digest, SHA-256 digest, SHA-1 digest, Thumbprint
 - details contains Alias, Subject, Version, Subject, Issuer, Serial Number, Not
 Before, Not After, Fingerprint, Signature Algorithm, MD5 digest, SHA-256 digest



(thumbprint), SHA-1 digest (thumbprint), Subject Public Key Information (which includes Key Algorithm, Key Length, and Key Data), and, if any, Certificate Extensions.

Example

orapki wallet display -wallet \$ORACLE_HOME/admin/db_unique_name/wallet

Related Topics

F.10.12 orapki wallet export

The orapki wallet export command exports certificate requests and certificates from an Oracle wallet.

Syntax

To export a certificate from an Oracle wallet:

```
orapki wallet export -wallet wallet\_location -dn certificate\_dn -cert certificate filename
```

- wallet specifies the location of the wallet from which you want to export the certificate.
- dn specifies the distinguished name of the certificate.
- cert specifies the name of the file that contains the exported certificate.

To export a certificate request from an Oracle wallet:

```
orapki wallet export -wallet wallet_location -dn certificate_request_dn -request certificate_request_filename
```

request specifies the name of the file that contains the exported certificate request.



G

How the Unified Auditing Migration Affects Individual Audit Features

Most of the pre-Oracle Database 12c release 1 (12.1) auditing features can be used before a unified auditing migration.

Table G-1 describes how the pre-Oracle Database 12c audit features change in the migration.

Table G-1 Availability of Unified Auditing Features Before and After Migration

Feature	Availability in Pre-Migrated Environment	Availability in Post-Migrated Environment
General Auditing Features	-	-
Operating system audit trail	Yes	No
XML file audit trail	Yes	No
Network auditing	Yes	No
The ability of users to audit and to removing auditing from their own schema objects	Yes	No
Mandatory auditing of audit administrative actions	No	Yes
Auditing Roles	-	-
AUDIT_ADMIN	Yes, but not needed for users who want to audit their own objects, nor for users who already have the ALTER SYSTEM privilege and want to change the auditing initialization parameters	Yes
AUDIT_VIEWER	Yes	Yes
System Tables	-	-
SYS.AUD\$	Yes	Yes, but will only have pre-unified audit records
SYS.FGA_LOG\$	Yes	Yes, but will only have pre-unified audit records
Initialization Parameters	-	-
AUDIT_TRAIL	Yes	Yes, but will not have any effect
AUDIT_FILE_DEST	Yes	Yes, but will not have any effect
AUDIT_SYS_OPERATIONS	Yes	Yes, but will not have any effect
AUDIT_SYSLOG_LEVEL	Yes	Yes, but will not have any effect
UNIFIED_AUDIT_SGA_QUEUE_SIZE	Yes, but note that this parameter has been deprecated, but is currently retained for backward compatibility.	Yes, but note that this parameter has been deprecated, but is currently retained for backward compatibility.
Data Dictionary Views ¹	-	-

Table G-1 (Cont.) Availability of Unified Auditing Features Before and After Migration

Feature	Availability in Pre-Migrated Environment	Availability in Post-Migrated Environment
ALL_AUDIT_POLICIES	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
DBA_AUDIT_POLICIES	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
DBA_AUDIT_POLICY_COLUMNS	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
DBA_COMMON_AUDIT_TRAIL	Yes	Yes, but will only have pre-unified audit records
DBA_AUDIT_EXISTS	Yes	Yes
DBA_AUDIT_OBJECT	Yes	Yes
DBA_AUDIT_POLICIES	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
DBA_AUDIT_POLICY_COLUMNS	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
DBA_AUDIT_SESSION	Yes	Yes, but will only have pre-unified audit records
DBA_AUDIT_STATEMENT	Yes	Yes, but will only have pre-unified audit records
DBA_AUDIT_TRAIL	Yes	Yes, but will only have pre-unified audit records. The RLS_INFO column captures audited Oracle VPD predicates.
DBA_FGA_AUDIT_TRAIL	Yes	Yes, but will only have pre-unified audit records. The RLS_INFO column captures audited Oracle VPD predicates.
DBA_OBJ_AUDIT_OPTS	Yes	Yes
DBA_PRIV_AUDIT_OPTS	Yes	Yes
DBA_STMT_AUDIT_OPTS	Yes	Yes
UNIFIED_AUDIT_TRAIL	Yes, but does not collect any audit records	Yes, and collects audit records
USER_AUDIT_OBJECT	Yes	Yes
USER_AUDIT_POLICY_COLUMN	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
USER_AUDIT_POLICIES	Yes	Yes, but only if fine-grained audit policies are created using the DBMS_FGA PL/SQL package
USER_AUDIT_SESSION	Yes	Yes
USER_AUDIT_STATEMENT	Yes	Yes



Table G-1 (Cont.) Availability of Unified Auditing Features Before and After Migration

Feature	Availability in Pre-Migrated Environment	Availability in Post-Migrated Environment
USER_AUDIT_TRAIL	Yes	Yes, but will only have pre-unified audit records
USER_OBJ_AUDIT_OPTS	Yes	Yes
V\$XML_AUDIT_TRAIL	Yes	Yes, but will only have pre-unified audit records. The RLS_INFO column captures audited Oracle VPD predicates.
CREATE AUDIT POLICY, ALTER AUDIT POLICY, and DROP AUDIT POLICY Statements	The statements are available, but the audit policies will not write to the old audit trails. When a policy is enabled, its audit records are written to the unified audit trail.	Yes, but writes the audit record to the unified audit trail only
AUDIT and NOAUDIT Statements	-	-
AUDIT	Yes, and can be used in a multitenant environment	Yes, but enhanced to enable audit policies; create application context audit settings; create audit records or success, failure, or both; and use in a multitenant environment
NOAUDIT	Yes, and can be used in a multitenant environment	Yes, but changed to disable audit policies, disable application context audit settings, and use in a multitenant environment
DBMS_FGA.ADD_POLICY Procedure Parameters	-	-
audit_trail	Yes, and is used as in previous releases	Yes, but when unified auditing is enabled, you can omit this parameter because all records will be written to the unified audit trail.
DBMS_AUDIT_MGMT Package AUDIT_TRAIL_TYPE Property Options	-	-
DBMS_AUDIT_MGMT.AUDIT_TRAIL_AUD_STD	Yes	Yes, but only pre-unified audit records
DBMS_AUDIT_MGMT.AUDIT_TRAIL_FGA_STD	Yes	Yes, but only pre-unified audit records
DBMS_AUDIT_MGMT.AUDIT_TRAIL_DB_ STD	Yes	Yes, but only pre-unified audit records
DBMS_AUDIT_MGMT.AUDIT_TRAIL_OS	Yes	Yes, but only pre-unified audit records
DBMS_AUDIT_MGMT.AUDIT_TRAIL_XML	Yes	Yes, but only pre-unified audit records
DBMS_AUDIT_MGMT.AUDIT_TRAIL_FIL ES	Yes	Yes, but only pre-unified audit records
DBMS_AUDIT_MGMT.AUDIT_TRAIL_ALL	Yes	Yes, but only pre-unified audit records
Oracle Database Vault Features	-	-



Table G-1 (Cont.) Availability of Unified Auditing Features Before and After Migration

Feature	Availability in Pre-Migrated Environment	Availability in Post-Migrated Environment
DVSYS.AUDIT_TRAIL\$ system table	Yes	Is renamed to DVSYS.OLD_AUDIT_TRAIL\$ and retains the old audit records. The previous DVSYS.AUDIT_TRAIL\$ table is made into a view named DVSYS.AUDIT_TRAIL\$. No new audit records are added.
Oracle Label Security Features	-	-
SA_AUDIT_ADMIN PL/SQL package	Yes	No

These data dictionary views will continue to show audit data from audit records that are still in the SYS.AUD\$ and SYS.FGA_LOG\$ system tables. Unified audit trail records are shown only in the unified audit trail-specific views. You must be granted the AUDIT_ADMIN or AUDIT_VIEWER role to query any views that are not prefaced with USER_.



Glossary

access control

The ability of a system to grant or limit access to specific data for specific clients or groups of clients.

Access Control Lists (ACLs)

The group of access directives that you define. The directives grant levels of access to specific data for specific clients, or groups of clients, or both.

Advanced Encryption Standard

Advanced Encryption Standard (AES) is a new cryptographic algorithm that has been approved by the National Institute of Standards and Technology as a replacement for DES. (DES is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.) The AES standard is available in Federal Information Processing Standards Publication 197. The AES algorithm is a symmetric block cipher that can process data blocks of 128 bits, using cipher keys with lengths of 128, 192, and 256 bits.

AES

See Advanced Encryption Standard

application context

A name-value pair that enables an application to access session information about a user, such as the user ID or other user-specific information, and then securely pass this data to the database.

See also global application context.

attribute

An item of information that describes some aspect of an entry in an LDAP directory. An entry comprises a set of attributes, each of which belongs to an object class. Moreover, each attribute has both a *type*, which describes the kind of information in the attribute, and a *value*, which contains the actual data.



application role

A database role that is granted to application users and that is secured by embedding passwords inside the application.

See also secure application role.

authentication

The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to granting access to resources in a system. A recipient of an authenticated message can be certain of the message's origin (its sender). Authentication is presumed to preclude the possibility that another party has impersonated the sender.

authentication method

A security method that verifies a user's, client's, or server's identity in distributed environments. Network authentication methods can also provide the benefit of single sign-on (SSO) for users. The following authentication methods are supported:

- Kerberos
- RADIUS
- Transport Layer Security (TLS)
- Windows native authentication

authorization

Permission given to a user, program, or process to access an object or set of objects. In Oracle, authorization is done through the role mechanism. A single person or a group of people can be granted a role or a group of roles. A role, in turn, can be granted other roles. The set of privileges available to an authenticated entity.

auto-login wallet

Password-based access to services without providing credentials at the time of access. This auto-login access stays in effect until the auto-login feature is disabled for that wallet. File system permissions provide the necessary security for auto-login wallet. When auto-login is enabled for a wallet, it is only available to the operating system user who created that wallet. Sometimes these are called "SSO wallets" because they provide single sign-on capability.

CDB

Multitenant container database. An Oracle Database installation that contains one root zero or more pluggable databases (PDBs). Every Oracle database is either a CDB or a non-CDB.



base

The root of a subtree search in an LDAP-compliant directory.

CA

See certificate authority

certificate

An ITU x.509 v3 standard data structure that securely binds an identify to a public key.

A certificate is created when an entity's public key is signed by a trusted identity, a certificate authority. The certificate ensures that the entity's information is correct, and that the public key belongs to that entity.

A certificate contains the entity's name, identifying information, and public key. It is also likely to contain a serial number, expiration date, and information about the rights, uses, and privileges associated with the certificate. Finally, it contains information about the certificate authority that issued it.

certificate authority

A trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are. When it certifies a user, the certificate authority first seeks verification that the user is not on the certificate revocation list (CRL), then verifies the user's identity and grants a certificate, signing it with the certificate authority's private key. The certificate authority has its own certificate and public key which it publishes. Servers and clients use these to verify signatures the certificate authority has made. A certificate authority might be an external company that offers certificate services, or an internal organization such as a corporate MIS department.

certificate chain

An ordered list of certificates containing an end-user or subscriber certificate and its certificate authority certificates.

certificate request

A certificate request, which consists of three parts: certification request information, a signature algorithm identifier, and a digital signature on the certification request information. The certification request information consists of the subject's distinguished name, public key, and an optional set of attributes. The attributes may provide additional information about the subject identity, such as postal address, or a challenge password by which the subject entity may later request certificate revocation. See PKCS #10.



certificate revocation list (CRL)

(CRLs) Signed data structures that contain a list of revoked certificate **s**. The authenticity and integrity of the CRL is provided by a digital signature appended to it. Usually, the CRL signer is the same entity that signed the issued certificate.

checksumming

A mechanism that computes a value for a message packet, based on the data it contains, and passes it along with the data to authenticate that the data has not been tampered with. The recipient of the data recomputes the cryptographic checksum and compares it with the cryptographic checksum passed with the data; if they match, it is "probabilistic" proof the data was not tampered with during transmission.

cleartext

Unencrypted plain text.

Cipher Block Chaining (CBC)

An encryption method that protects against block replay attacks by making the encryption of a cipher block dependent on all blocks that precede it; it is designed to make unauthorized decryption incrementally more difficult. Oracle Database employs *outer* cipher block chaining because it is more secure than *inner* cipher block chaining, with no material performance penalty.

CIDR

The standard notation used for IP addresses. In CIDR notation, an IPv6 subnet is denoted by the subnet prefix and the size in bits of the prefix (in decimal), separated by the slash (/) character. For example, fe80:0000:0217:f2ff::/64 denotes a subnet with addresses fe80:0000:0217:f2ff:0000:0000:0000:0000 through fe80:0000:0217:f2ff:ffff:ffff:ffff. The CIDR notation includes support for IPv4 addresses. For example, 192.0.2.1/24 denotes the subnet with addresses 192.0.2.1 through 192.0.2.255.

cipher suite

A set of authentication, encryption, and data integrity algorithms used for exchanging messages between network nodes. During a TLS handshake, for example, the two nodes negotiate to see which cipher suite they will use when transmitting messages back and forth.

cipher suite name

Cipher suites describe the kind of cryptographics protection that is used by connections in a particular session.

ciphertext

Message text that has been encrypted.

Classless Inter-Domain Routing

See CIDR.

client

A client relies on a service. A client can sometimes be a user, sometimes a process acting on behalf of the user during a database link (sometimes called a proxy).

common privilege grant

A privilege that a common user grants to another common user or to a common role. Common privilege grants can be either system privileges or object privileges, and they apply across all PDBs in a CDB.

See also local privilege grant.

common role

A role that exists in all containers in a CDB.

common user

In a CDB, a database user that exists with the same identity in every existing and future PDB.

confidentiality

A function of cryptography. Confidentiality guarantees that only the intended recipient(s) of a message can view the message (decrypt the ciphertext).

connect descriptor

A specially formatted description of the destination for a network connection. A connect descriptor contains destination service and network route information. The destination service is indicated by using its service name for Oracle9*i* or Oracle8*i* databases or its Oracle system identifier (SID) for Oracle databases version 8.0. The network route provides, at a minimum, the location of the listener through use of a network address. See connect identifier

connect identifier

A name, net service name, or service name that resolves to a connect descriptor. Users initiate a connect request by passing a user name and password along with a connect identifier in a connect string for the service to which they want to connect.

For example:

CONNECT username@connect_identifier Enter password: password

connect string

Information the user passes to a service to connect, such as user name, password and net service name. For example:

CONNECT username@net_service_name
Enter password: password

container

In a CDB either, a root or a PDB.

container data object

In a CDB, a table or view containing data pertaining to multiple containers and possibly the CDB as a whole, along with mechanisms to restrict data visible to specific common users through such objects to one or more containers. Examples of container data objects are Oracle-supplied views whose names begin with V\$ and CDB.

credentials

A user name, password, or certificate used to gain access to the database.

CRL

See certificate revocation list (CRL)

CRL Distribution Point

(CRL DP) An optional extension specified by the X.509 version 3 certificate standard, which indicates the location of the Partitioned CRL where revocation information for a certificate is stored. Typically, the value in this extension is in the form of a URL. CRL DPs allow revocation information within a single certificate authority domain to be posted in multiple CRLs. CRL DPs subdivide revocation information into more manageable pieces to avoid proliferating voluminous CRLs, thereby providing performance benefits. For example, a CRL DP is specified in the certificate and can point to a file on a Web server from which that certificate's revocation information can be downloaded.

CRL DP

See CRL Distribution Point

cryptography

The practice of encoding and decoding data, resulting in secure messages.

data dictionary

A set of read-only tables that provide information about a database.

Data Encryption Standard (DES)

An older Federal Information Processing Standards encryption algorithm superseded by the Advanced Encryption Standard (AES). The DES, DES40, 3DES112, and 3DES168 algorithms are deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

database administrator

(1) A person responsible for operating and maintaining an Oracle Server or a database application. (2) An Oracle user name that has been given DBA privileges and can perform database administration functions. Usually the two meanings coincide. Many sites have multiple DBAs.

database alias

See net service name

Database Installation Administrator

Also called a database creator. This administrator is in charge of creating new databases. This includes registering each database in the directory using the Database Configuration Assistant. This administrator has create and modify access to database service objects and attributes. This administrator can also modify the Default domain.

database link

A network object stored in the local database or in the network definition that identifies a remote database, a communication path to that database, and optionally, a user name and password. Once defined, the database link is used to access the remote database.

A public or private database link from one database to another is created on the local database by a DBA or user.

A global database link is created automatically from each database to every other database in a network with Oracle Names. Global database links are stored in the network definition.

database password version

An irreversible value that is derived from the user's database password. It is also called a password verifier. This value is used during password authentication to the database to prove the identity of the connecting user.

Database Security Administrator

The highest level administrator for database enterprise user security. This administrator has permissions on all of the enterprise domains and is responsible for:



Administering the Oracle DBSecurityAdmins and OracleDBCreators groups.

Creating new enterprise domains.

Moving databases from one domain to another within the enterprise.

decryption

The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).

definer's rights procedure

A procedure (or program unit) that executes with the privileges of its owner, not its current user. Definer's rights subprograms are bound to the schema in which they are located.

For example, assume that user blake and user scott each have a table called dept in their respective user schemas. If user blake calls a definer's rights procedure, which is owned by user scott, to update the dept table, then this procedure will update the dept table in the scott schema. This is because the procedure executes with the privileges of the user who owns (defined) the procedure (that is, scott).

See also invoker's rights procedure.

denial-of-service (DoS) attack

An attack that renders a Web site inaccessible or unusable. The denial-of-service attack can occur in many different ways but frequently includes attacks that cause the site to crash, reject connections, or perform too slowly to be usable. DoS attacks come in two forms:

- Basic denial-of-service attacks, which require only one or a few computers
- Distributed DoS attacks, which require many computers to execute

DES

See Data Encryption Standard (DES)

dictionary attack

A common attack on passwords. The attacker creates a list of many common passwords and encrypts them. Then the attacker steals a file containing encrypted passwords and compares it to his list of encrypted common passwords. If any of the encrypted password values (called verifiers) match, then the attacker can steal the corresponding password. Dictionary attacks can be avoided by using "salt" on the password before encryption. See salt.

Diffie-Hellman key negotiation algorithm

This is a method that lets two parties communicating over an insecure channel to agree upon a random number known only to them. Though the parties exchange information over the insecure channel during execution of the Diffie-Hellman key negotiation algorithm, it is

computationally infeasible for an attacker to deduce the random number they agree upon by analyzing their network communications. Oracle Database uses the Diffie-Hellman key negotiation algorithm to generate session keys.

digital signature

A digital signature is created when a public key algorithm is used to sign the sender's message with the sender's private key. The digital signature assures that the document is authentic, has not been forged by another entity, has not been altered, and cannot be repudiated by the sender.

directory information tree (DIT)

A hierarchical tree-like structure consisting of the DNs of the entries in an LDAP directory. See distinguished name (DN)

directory naming

A naming method that resolves a database service, net service name, or net service alias to a connect descriptor stored in a central directory server. A

directory naming context

A subtree which is of significance within a directory server. It is usually the top of some organizational subtree. Some directories only permit one such context which is fixed; others permit none to many to be configured by the directory administrator.

distinguished name (DN)

The unique name of a directory entry. It is comprised of all of the individual names of the parent entries back to the root entry of the directory information tree. See directory information tree (DIT)

domain

Any tree or subtree within the Domain Name System (DNS) namespace. Domain most commonly refers to a group of computers whose host names share a common suffix, the domain name.

Domain Name System (DNS)

A system for naming computers and network services that is organized into a hierarchy of domains. DNS is used in TCP/IP networks to locate computers through user-friendly names. DNS resolves a friendly name into an IP address, which is understood by computers.

In Oracle Net Services, DNS translates the host name in a TCP/IP address into an IP address.

directly granted role

A role that has been granted directly to the user, as opposed to an indirectly granted role.

encrypted text

Text that has been encrypted, using an encryption algorithm; the output stream of an encryption process. On its face, it is not readable or decipherable, without first being subject to decryption. Also called ciphertext. Encrypted text ultimately originates as plaintext.

encryption

Disguising a message, rendering it unreadable to all but the intended recipient.

enterprise domain

A directory construct that consists of a group of databases and enterprise roles. A database should only exist in one enterprise domain at any time. Enterprise domains are different from Windows 2000 domains, which are collections of computers that share a common directory database.

Enterprise Domain Administrator

User authorized to manage a specific enterprise domain, including the authority to add new enterprise domain administrators.

enterprise role

Access privileges assigned to enterprise users. A set of Oracle role-based authorizations across one or more databases in an enterprise domain. Enterprise roles are stored in the directory and contain one or more global roles.

enterprise user

A user defined and managed in a directory. Each enterprise user has a unique identify across an enterprise.

entry

The building block of a directory, it contains information about an object of interest to directory users.

external authentication

Verification of a user identity by a third party authentication service, such as Kerberos or RADIUS.

Federal Information Processing Standard (FIPS)

A U.S. government standard that defines security requirements for cryptographic modules employed within a security system protecting unclassified information within computer and



telecommunication systems. Published by the National Institute of Standards and Technology (NIST).

FIPS

See Federal Information Processing Standard (FIPS).

forced cleanup

The ability to forcibly cleanup (that is, remove) all audit records from the database. To accomplish this, you set the <code>USE_LAST_ARCH_TIMESTAMP</code> argument of the <code>DBMS_AUDIT_MGMT.CLEAN_AUDIT_TRAIL_procedure</code> to <code>FALSE</code>.

See also purge job.

forest

A group of one or more Active Directory trees that trust each other. All trees in a forest share a common schema, configuration, and global catalog. When a forest contains multiple trees, the trees do not form a contiguous namespace. All trees in a given forest trust each other through transitive bidirectional trust relationships.

Forwardable Ticket Granting Ticket

A special Kerberos ticket that can be forwarded to proxies, permitting the proxy to obtain additional Kerberos tickets on behalf of the client for proxy authentication.

See also Kerberos ticket.

global role

A role managed in a directory, but its privileges are contained within a single database. A global role is created in a database by using the following syntax:

```
CREATE ROLE role name IDENTIFIED GLOBALLY;
```

global application context

A name-value pair that enables application context values to be accessible across database sessions.

See also application context.

grid computing

A computing architecture that coordinates large numbers of servers and storage to act as a single large computer. Oracle Grid Computing creates a flexible, on-demand computing resource for all enterprise computing needs. Applications running on the Oracle Database grid computing infrastructure can take advantage of common infrastructure services for failover,

software provisioning, and management. Oracle Grid Computing analyzes demand for resources and adjusts supply accordingly.

HTTP

Hypertext Transfer Protocol: The set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. Relative to the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), HTTP is an application protocol.

HTTPS

The use of Transport Layer Security (TLS) as a sublayer under the regular HTTP application layer.

indirectly granted role

A role granted to a user through another role that has already been granted to this user. Then you grant the role2 and role3 roles to the role1 role. Roles role2 and role3 are now under role1. This means psmith has been indirectly granted the roles role2 and role3, in addition to the direct grant of role1. Enabling the direct role1 for psmith enables the indirect role2 and role3 for this user as well.

identity

The combination of the public key and any other public information for an entity. The public information may include user identification data such as, for example, an e-mail address. A user certified as being the entity it claims to be.

identity management

The creation, management, and use of online, or digital, entities. Identity management involves securely managing the full life cycle of a digital identity from creation (provisioning of digital identities) to maintenance (enforcing organizational policies regarding access to electronic resources), and, finally, to termination.

identity management realm

A subtree in Oracle Internet Directory, including not only an Oracle Context, but also additional subtrees for users and groups, each of which are protected with access control lists.

initial ticket

In Kerberos authentication, an initial ticket or ticket granting ticket (TGT) identifies the user as having the right to ask for additional service tickets. No tickets can be obtained without an initial ticket. An initial ticket is retrieved by running the <code>okinit</code> program and providing a password.



instance

Every running Oracle database is associated with an Oracle instance. When a database is started on a database server (regardless of the type of computer), Oracle allocates a memory area called the System Global Area (SGA) and starts an Oracle process. This combination of the SGA and an Oracle process is called an instance. The memory and the process of an instance manage the associated database's data efficiently and serve the one or more users of the database.

integrity

A guarantee that the contents of a message received were not altered from the contents of the original message sent.

invoker's rights procedure

A procedure (or program unit) that executes with the privileges of the current user, that is, the user who invokes the procedure. These procedures are not bound to a particular schema. They can be run by a variety of users and allow multiple users to manage their own data by using centralized application logic. Invoker's rights procedures are created with the AUTHID clause in the declaration section of the procedure code.

For example, assume that user blake and user scott each have a table called dept in their respective user schemas. If user blake calls an invoker's rights procedure, which is owned by user scott, to update the dept table, then this procedure will update the dept table in the blake schema. This is because the procedure executes with the privileges of the user who invoked the procedure (that is, blake.).

See also definer's rights procedure.

java code obfuscation

Java code obfuscation is used to protect Java programs from reverse engineering. A special program (an obfuscator) is used to scramble Java symbols found in the code. The process leaves the original program structure intact, letting the program run correctly while changing the names of the classes, methods, and variables in order to hide the intended behavior. Although it is possible to decompile and read non-obfuscated Java code, the obfuscated Java code is sufficiently difficult to decompile to satisfy U.S. government export controls.

Java Database Connectivity (JDBC)

An industry-standard Java interface for connecting to a relational database from a Java program, defined by Sun Microsystems.

JDBC

See Java Database Connectivity (JDBC)



KDC

See Key Distribution Center (KDC).

Kerberos

A network authentication service developed under Massachusetts Institute of Technology's Project Athena that strengthens security in distributed environments. Kerberos is a trusted third-party authentication system that relies on shared secrets and assumes that the third party is secure. It provides single sign-on capabilities and database link authentication (MIT Kerberos only) for users, provides centralized password storage, and enhances PC security.

Kerberos ticket

A temporary set of electronic credentials that verify the identity of a client for a particular service. Also referred to as a service ticket.

Key Distribution Center (KDC)

In Kerberos authentication, the KDC maintains a list of user principals and is contacted through the kinit (okinit is the Oracle version) program for the user's initial ticket. Frequently, the KDC and the Ticket Granting Service are combined into the same entity and are simply referred to as the KDC. The Ticket Granting Service maintains a list of service principals and is contacted when a user wants to authenticate to a server providing such a service. The KDC is a trusted third party that must run on a secure host. It creates ticket-granting tickets and service tickets.

See also Kerberos ticket.

key pair

A public key and its associated private key. See public and private key pair.

keytab file

A Kerberos key table file containing one or more service keys. Hosts or services use *keytab* files in the same way as users use their passwords.

kinstance

An instantiation or location of a Kerberos authenticated service. This is an arbitrary string, but the host Computer name for a service is typically specified.

kservice

An arbitrary name of a Kerberos service object.

last archive timestamp

A timestamp that indicates the timestamp of the last archived audit record. For the database audit trail, this timestamp indicates the last audit record archived. For operating system audit

files, it indicates the highest last modified timestamp property of the audit file that was archived. To set this timestamp, you use the <code>DBMS_AUDIT_MGMT.SET_LAST_ARCHIVE_TIMESTAMP</code> PL/SQL procedure.

See also purge job.

LDAP

See Lightweight Directory Access Protocol (LDAP)

Idap.ora file

A file created by Oracle Net Configuration Assistant that contains the following directory server access information:

- Type of directory server
- Location of the directory server
- Default identity management realm or Oracle Context (including ports) that the client or server will use

Lightweight Directory Access Protocol (LDAP)

A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. The framework of design conventions supporting industry-standard directory products, such as the Oracle Internet Directory.

listener

A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.

Every time a client requests a network session with a server, a listener receives the actual request. If the client information matches the listener information, then the listener grants a connection to the server.

listener.ora file

A configuration file for the listener that identifies the:

- Listener name
- · Protocol addresses that it is accepting connection requests on
- Services it is listening for

The listener.ora file typically resides in <code>\$ORACLE_HOME/network/admin</code> on UNIX platforms and <code>ORACLE_BASE/ORACLE_HOME/network/admin</code> on Windows.

lightweight user session

A user session that contains only information pertinent to the application that the user is logging onto. The lightweight user session does not hold its own database resources, such as

transactions and cursors; hence it is considered "lightweight." Lightweight user sessions consume far less system resources than traditional database session. Because lightweight user sessions consume much fewer server resources, a lightweight user session can be dedicated to each end user and can persist for as long as the application deems necessary.

local privilege grant

A privilege that applies only to the PDB in which it was granted.

See also common privilege grant.

local role

In a CDB, a role that exists only in a single PDB, just as a role in a non-CDB exists only in the non-CDB. Unlike a common role, a local role may only contain roles and privileges that apply within the container in which the role exists.

local user

In a CDB, any user that is not a common user.

mandatory auditing

Activities that are audited by default. Examples are modifications to unified audit trail policies (such as ALTER AUDIT POLICY statements) and top level statements by the administrative users SYS, SYSDBA, SYSOPER, SYSASM, SYSBACKUP, SYSDG, and SYSKM, until the database opens. See "Activities That Are Mandatorily Audited" for more information.

MD5

Message Digest 5. An algorithm that assures data integrity by generating a 128-bit cryptographic message digest value from given data. If as little as a single bit value in the data is modified, the MD5 checksum for the data changes. Forgery of data in a way that will cause MD5 to generate the same result as that for the original data is considered computationally infeasible.

MD5 is deprecated in this release. To transition your Oracle Database environment to use stronger algorithms, download and install the patch described in My Oracle Support note 2118136.2.

message authentication code

Also known as data authentication code (DAC). A checksumming with the addition of a secret key. Only someone with the key can verify the cryptographic checksum.

message digest

See checksumming

CDB

See CDB.

namespace

In Oracle Database security, the name of an application context. You create this name in a CREATE CONTEXT statement.

naming method

The resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service.

National Institute of Standards and Technology (NIST)

An agency within the U.S. Department of Commerce responsible for the development of security standards related to the design, acquisition, and implementation of cryptographic-based security systems within computer and telecommunication systems, operated by a Federal agency or by a contractor of a Federal agency or other organization that processes information on behalf of the Federal Government to accomplish a Federal function.

net service alias

An alternative name for a directory naming object in a directory server. A directory server stores net service aliases for any defined net service name or database service. A net service alias entry does not have connect descriptor information. Instead, it only references the location of the object for which it is an alias. When a client requests a directory lookup of a net service alias, the directory determines that the entry is a net service alias and completes the lookup as if it was actually the entry it is referencing.

net service name

A simple name for a service that resolves to a connect descriptor. Users initiate a connect request by passing a user name and password along with a net service name in a connect string for the service to which they want to connect:

```
CONNECT username@net_service_name
Enter password: password
```

Depending on your needs, net service names can be stored in a variety of places, including:

- Local configuration file, tnsnames.ora, on each client
- Directory server
- External naming service, such as NIS

network authentication service

A means for authenticating clients to servers, servers to servers, and users to both clients and servers in distributed environments. A network authentication service is a repository for storing

information about users and the services on different servers to which they have access, as well as information about clients and servers on the network. An authentication server can be a physically separate computer, or it can be a facility co-located on another server within the system. To ensure availability, some authentication services may be replicated to avoid a single point of failure.

network listener

A listener on a server that listens for connection requests for one or more databases on one or more protocols. See listener.

NIST

See National Institute of Standards and Technology (NIST).

non-CDB

An Oracle database that is not a CDB.

non-repudiation

Incontestable proof of the origin, delivery, submission, or transmission of a message.

obfuscation

A process by which information is scrambled into a non-readable form, such that it is extremely difficult to de-scramble if the algorithm used for scrambling is not known.

obfuscator

A special program used to obfuscate Java source code. See obfuscation.

object class

A named group of attributes. When you want to assign attributes to an entry, you do so by assigning to that entry the object classes that hold those attributes. All objects associated with the same object class share the same attributes.

Oracle Context

1. An entry in an LDAP-compliant internet directory called <code>cn=OracleContext</code>, under which all Oracle software relevant information is kept, including entries for Oracle Net Services directory naming and checksumming security.

There can be one or more Oracle Contexts in a directory. An Oracle Context is usually located in an identity management realm.

Oracle Virtual Private Database

A set of features that enables you to create security policies to control database access at the row and column level. Essentially, Oracle Virtual Private Database adds a dynamic WHERE clause to a SQL statement that is issued against the table, view, or synonym to which an Oracle Virtual Private Database security policy was applied.

Oracle Net Services

An Oracle product that enables two or more computers that run the Oracle server or Oracle tools such as Designer/2000 to exchange data through a third-party network. Oracle Net Services support distributed processing and distributed database capability. Oracle Net Services is an open system because it is independent of the communication protocol, and users can interface Oracle Net to many network environments.

Oracle PKI certificate usages

Defines Oracle application types that a certificate supports.

Password-Accessible Domains List

A group of enterprise domains configured to accept connections from password-authenticated users.

PCMCIA cards

Small credit card-sized computing devices that comply with the Personal Computer Memory Card International Association (PCMCIA) standard. These devices, also called PC cards, are used for adding memory, modems, or as hardware security modules. PCMCIA cards that are used as hardware security modules securely store the private key component of a public and private key pair and some also perform the cryptographic operations as well.

PDB

An individual database that is part of a CDB.

See also root.

peer identity

SSL connect sessions are between a particular client and a particular server. The identity of the peer may have been established as part of session setup. Peers are identified by X.509 certificate chains.

PEM

The Internet Privacy-Enhanced Mail protocols standard, adopted by the Internet Architecture Board to provide secure electronic mail over the Internet. The PEM protocols provide for encryption, authentication, message integrity, and key management. PEM is an inclusive standard, intended to be compatible with a wide range of key-management approaches,

including both symmetric and public-key schemes to encrypt data-encrypting keys. The specifications for PEM come from four Internet Engineering Task Force (IETF) documents: RFCs 1421, 1422, 1423, and 1424.

PKCS #10

An RSA Security, Inc., Public-Key Cryptography Standards (PKCS) specification that describes a syntax for certification requests. A certification request consists of a distinguished name, a public key, and optionally a set of attributes, collectively signed by the entity requesting certification. Certification requests are referred to as certificate requests in this manual. See certificate request

PKCS #11

An RSA Security, Inc., Public-Key Cryptography Standards (PKCS) specification that defines an application programming interface (API), called Cryptoki, to devices which hold cryptographic information and perform cryptographic operations. See PCMCIA cards

PKCS #12

An RSA Security, Inc., Public-Key Cryptography Standards (PKCS) specification that describes a transfer syntax for storing and transferring personal authentication credentials—typically in a format called a wallet.

PKI

See public key infrastructure (PKI)

plaintext

Message text that has not been encrypted.

pluggable database

See PDB.

principal

A string that uniquely identifies a client or server to which a set of Kerberos credentials is assigned. It generally has three parts: kservice/kinstance@REALM. In the case of a user, kservice is the user name. See also kservice, kinstance, and realm

private key

In public-key cryptography, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures. See public and private key pair.

proxy authentication

A process typically employed in an environment with a middle tier such as a firewall, wherein the end user authenticates to the middle tier, which thence authenticates to the directory on the user's behalf—as its *proxy*. The middle tier logs into the directory as a *proxy user*. A proxy user can switch identities and, once logged into the directory, switch to the end user's identity. It can perform operations on the end user's behalf, using the authorization appropriate to that particular end user.

public key

In public-key cryptography, this key is made public to all. It is primarily used for encryption but can be used for verifying signatures. See public and private key pair.

public and private key pair

A set of two numbers used for encryption and decryption, where one is called the private key and the other is called the public key. Public keys are typically made widely available, while private keys are held by their respective owners. Though mathematically related, it is generally viewed as computationally infeasible to derive the private key from the public key. Public and private keys are used only with asymmetric encryption algorithms, also called public-key encryption algorithms, or public-key cryptosystems. Data encrypted with either a public key or a private key from a key pair can be decrypted with its associated key from the key-pair. However, data encrypted with a public key cannot be decrypted with the same public key, and data enwrapped with a private key cannot be decrypted with the same private key.

public key infrastructure (PKI)

Information security technology utilizing the principles of public key cryptography. Public key cryptography involves encrypting and decrypting information using a shared public and private key pair. Provides for secure, private communications within a public network.

PUBLIC role

A special role that every database account automatically has. By default, it has no privileges assigned to it, but it does have grants to many Java objects. You cannot drop the PUBLIC role, and a manual grant or revoke of this role has no meaning, because the user account will always assume this role. Because all database user accounts assume the PUBLIC role, it does not appear in the DBA ROLES and SESSION ROLES data dictionary views.

purge job

A database job created by the <code>DBMS_AUDIT_MGMT.CREATE_PURGE_JOB</code> procedure, which manages the deletion of the audit trail. A database administrator schedules, enables, and disables the purge job. When the purge job becomes active, it deletes audit records from the database audit tables, or it deletes Oracle Database operating system audit files.

See also forced cleanup, last archive timestamp.



RADIUS

Remote Authentication Dial-In User Service (RADIUS) is a client/server protocol and software that enables remote access servers to communicate with a central server to authenticate dialin users and authorize their access to the requested system or service.

realm

1. Short for identity management realm. 2. A Kerberos object. A set of clients and servers operating under a single key distribution center/ticket-granting service (KDC/TGS). Services (see kservice) in different realms that share the same name are unique.

realm Oracle Context

An Oracle Context that is part of an identity management realm in Oracle Internet Directory.

registry

A Windows repository that stores configuration information for a computer.

remote computer

A computer on a network other than the local computer.

role

A named group of related privileges that you grant as a group to users or other roles.

See also indirectly granted role.

root

In a multitenant environment, a collection of Oracle-supplied and user-created schemas to which all PDBs belong. The container database has only one root. Each PDB is considered to be a child of this root. Root has an entry in its data dictionary that indicates the existence of each PDB.

See also container, CDB, PDB.

root key certificate

See trusted certificate

salt

In cryptography, a way to strengthen the security of encrypted data. Salt is a random string that is added to the data before it is encrypted, making it more difficult for attackers to steal the data by matching patterns of ciphertext to known ciphertext samples. Salt is often also added to passwords, before the passwords are encrypted, to avoid dictionary attacks, a method that unethical hackers (attackers) use to steal passwords. The encrypted salted values make it

difficult for attackers to match the hash value of encrypted passwords (sometimes called verifiers) with their dictionary lists of common password hash values.

schema

1. Database schema: A named collection of objects, such as tables, views, clusters, procedures, packages, attributes, object classes, and their corresponding matching rules, which are associated with a particular user. 2. LDAP directory schema: The collection of attributes, object classes, and their corresponding matching rules.

schema mapping

See user-schema mapping

secure application role

A database role that is granted to application users, but secured by using an invoker's right stored procedure to retrieve the role password from a database table. A secure application role password is not embedded in the application.

See also application role.

Secure Hash Algorithm (SHA)

An algorithm that assures data integrity by generating a 160-bit cryptographic message digest value from given data. If as little as a single bit in the data is modified, the Secure Hash Algorithm checksum for the data changes. Forgery of a given data set in a way that will cause the Secure Hash Algorithm to generate the same result as that for the original data is considered computationally infeasible.

An algorithm that takes a message of less than 264 bits in length and produces a 160-bit message digest. The algorithm is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision and inversion attacks.

Secure Sockets Layer (SSL)

An industry standard protocol designed by Netscape Communications Corporation for securing network connections. SSL provides authentication, encryption, and data integrity using public key infrastructure (PKI).

The Transport Layer Security (TLS) protocol is the successor to the SSL protocol.

separation of duty

Restricting activities only to those users who must perform them. For example, you should not grant the SYSDBA administrative privilege to any user. Only grant this privilege to administrative users. Separation of duty is required by many compliance policies. See "Guidelines for

Securing User Accounts and Privileges" for guidelines on granting privileges to the correct users.

server

A provider of a service.

service

- 1. A network resource used by clients; for example, an Oracle database server.
- An executable process installed in the Windows registry and administered by Windows.Once a service is created and started, it can run even when no user is logged on to the computer.

service name

For Kerberos-based authentication, the kservice portion of a service principal.

service principal

See principal

service key table

In Kerberos authentication, a service key table is a list of service principals that exist on a kinstance. This information must be extracted from Kerberos and copied to the Oracle server computer before Kerberos can be used by Oracle.

service ticket

A service ticket is trusted information used to authenticate the client, to a specific service or server, for a predetermined period of time. It is obtained from the KDC using the initial ticket. See also Kerberos ticket.

session key

A key shared by at least two parties (usually a client and a server) that is used for data encryption for the duration of a single communication session. Session keys are typically used to encrypt network traffic; a client and a server can negotiate a session key at the beginning of a session, and that key is used to encrypt all network traffic between the parties for that session. If the client and server communicate again in a new session, they negotiate a new session key.

session layer

A network layer that provides the services needed by the presentation layer entities that enable them to organize and synchronize their dialogue and manage their data exchange. This layer establishes, manages, and terminates network sessions between the client and server. An example of a session layer is Network Session.

SHA

See Secure Hash Algorithm (SHA).

shared schema

A database or application schema that can be used by multiple enterprise users. Oracle Database supports the mapping of multiple enterprise users to the same shared schema on a database, which lets an administrator avoid creating an account for each user in every database. Instead, the administrator can create a user in one location, the enterprise directory, and map the user to a shared schema that other enterprise users can also map to. Sometimes called user/schema separation.

single key-pair wallet

A PKCS #12-format wallet that contains a single user certificate and its associated private key. The public key is imbedded in the certificate.

single password authentication

The ability of a user to authenticate with multiple databases by using a single password. In the Oracle Database implementation, the password is stored in an LDAP-compliant directory and protected with encryption and Access Control Lists.

single sign-on (SSO)

The ability of a user to *authenticate once*, combined with strong authentication occurring transparently in subsequent connections to other databases or applications. Single sign-on lets a user access multiple accounts and applications with a single password, entered during a single connection. *Single password, single authentication*. Oracle Database supports Kerberos and SSL-based single sign-on.

smart card

A plastic card (like a credit card) with an embedded integrated circuit for storing information, including such information as user names and passwords, and also for performing computations associated with authentication exchanges. A smart card is read by a hardware device at any client or server.

A smartcard can generate random numbers which can be used as one-time use passwords. In this case, smartcards are synchronized with a service on the server so that the server expects the same password generated by the smart card.



sniffer

Device used to surreptitiously listen to or capture private data traffic from a network.

SSO

See single sign-on (SSO)

System Global Area (SGA)

A group of shared memory structures that contain data and control information for an Oracle instance.

system identifier (SID)

A unique name for an Oracle instance. To switch between Oracle databases, users must specify the desired SID. The SID is included in the CONNECT DATA parts of the connect descriptor in a thinames.ora file, and in the definition of the network listener in a listener.ora file.

third-party attack

A security attack characterized by the third-party, surreptitious interception of a message, wherein the third-party decrypts the message, re-encrypts it (with or without alteration of the original message), and re-transmits it to the originally-intended recipient—all without the knowledge of the legitimate sender and receiver. This type of security attack works only in the absence of authentication. Formerly called man-in-the-middle attack.

ticket

A piece of information that helps identify who the owner is. See initial ticket and service ticket.

tnsnames.ora

A file that contains connect descriptors; each connect descriptor is mapped to a net service name. The file may be maintained centrally or locally, for use by all or individual clients. This file typically resides in the following locations depending on your platform:

- (UNIX) ORACLE_HOME/network/admin
- (Windows) ORACLE BASE\ORACLE_HOME\network\admin

token card

A device for providing improved ease-of-use for users through several different mechanisms. Some token cards offer one-time passwords that are synchronized with an authentication service. The server can verify the password provided by the token card at any given time by contacting the authentication service. Other token cards operate on a challenge-response basis. In this case, the server offers a challenge (a number) which the user types into the token

card. The token card then provides another number (cryptographically-derived from the challenge), which the user then offers to the server.

transport layer

A networking layer that maintains end-to-end reliability through data flow control and error recovery methods. Oracle Net Services uses *Oracle protocol supports* for the transport layer.

Transport Layer Security (TLS)

An industry standard protocol for securing network connections. The TLS protocol is a successor to the SSL protocol. It provides authentication, encryption, and data integrity using public key infrastructure (PKI). The TLS protocol is developed by the Internet Engineering Task Force (IETF).

trusted certificate

A trusted certificate, sometimes called a root key certificate, is a third party identity that is qualified with a level of trust. The trusted certificate is used when an identity is being validated as the entity it claims to be. Typically, the certificate authorities you trust are called trusted certificates. If there are several levels of trusted certificates, a trusted certificate at a lower level in the certificate chain does not need to have all its higher level certificates reverified.

trusted certificate authority

See certificate authority.

trust point

See trusted certificate.

user name

A name that can connect to and access objects in a database.

user-schema mapping

An LDAP directory entry that contains a pair of values: the base in the directory at which users exist, and the name of the database schema to which they are mapped. The users referenced in the mapping are connected to the specified schema when they connect to the database. User-schema mapping entries can apply only to one database or they can apply to all databases in a domain. See shared schema.

user/schema separation

See shared schema.



user search base

The node in the LDAP directory under which the user resides.

views

Selective presentations of one or more tables (or other views), showing both their structure and their data.

wallet

A data structure used to store and manage security credentials for an individual entity.

Windows native authentication

An authentication method that enables a client single login access to a Windows server and a database running on that server.

X.509

An industry-standard specification for digital certificate s.



Index

Symbols	access control list (ACL) (continued)
"all nerminational" A 2	external network services (continued)
"all permissions", A-3	port ranges, 10-18
Numerics	privilege assignments, about, 10-19 privilege assignments, database administrators checking, 10-20
12C password hash version	privilege assignments, users checking,
about, <i>3-37</i>	10-21
12C password version	revoking privileges, 10-6
recommended by Oracle, 3-37	wallet access
	about, <i>10-3</i>
A	advantages, 10-3
	client certificate credentials, using, 10-8
about, 6-2, 9-22	finding information about, 10-22
about connection, 6-6	non-shared wallets, 10-8
ACCEPT_MD5_CERTS sqlnet.ora parameter, <i>F-3</i>	password credentials, 10-8
ACCEPT_SHA1_CERTS sqlnet.ora parameter,	password credentials, using, 10-8
F-3	revoking, 10-12
access configuration, DBCA, 6-19	revoking access, 10-12
access configuration, silent mode, 6-20	shared database session, 10-8 wallets with sensitive information, 10-8
access configuration, system parameters, 6-18	wallets without sensitive information, 10-8
access control	ACCHK_READ role, 4-29
encryption, problems not solved by, 17-1	account locking
enforcing, A-13	example, 3-10
object privileges, 4-51	explicit, 3-11
password encryption, 3-3	password management, 3-10
access control list (ACL), 10-2, 10-4	PASSWORD_LOCK_TIME profile parameter,
examples	3-10
external network connection for email	accounting, RADIUS, 24-18
alert, 27-105	activating checksumming and encryption, 18-8
external network connections, 10-13 wallet access, 10-13	ad hoc tools
external network services	database access, security problems of, 4-42
about, 10-2	adapters, 20-6
advantages, 10-2	ADD_SSLV3_TO_DEFAULT sqlnet.ora
affect of upgrade from earlier release,	parameter, 23-22
10-3	ADG_ACCOUNT_INFO_TRACKING initialization
email alert for audit violation tutorial,	parameter
27-105	guideline for securing, A-13
finding information about, 10-22	ADM_PARALLEL_EXECUTE_TASK role
network hosts, using wildcards to specify,	about, <i>4-29</i>
10-17	ADMIN OPTION
ORA-06512 error, 10-21	about, 4-71
ORA-24247 error, 10-21	revoking privileges, 4-76
ORA-24247 errors, 10-3	revoking roles, 4-76
order of precedence hosts 10-17	roles, <i>4-41</i>

ADMIN OPTION (continued)	ALTER SESSION statement
system privileges, 4-12	schema, setting current, 12-23
administrative accounts	ALTER USER privilege, 2-18
about, 2-32	ALTER USER statement
predefined, listed, 2-32	changing SYS password with, 2-21
administrative privileges	default roles, 4-85
about, 4-5	explicit account unlocking, 3-11
granting to users, 4-5	profiles, changing, 3-13
SYSBACKUP privilege, 4-6	REVOKE CONNECT THROUGH clause, 3-82
SYSDBA privilege, 4-5	altering users, 2-18
SYSDG privilege, 4-7	ANO encryption
SYSKM privilege, 4-8	configuring with SSL authentication, 18-15
SYSOPER privilege, 4-5	anonymous, 23-16
SYSRAC privilege, 4-9	ANONYMOUS user account, 2-32
administrative user passwords	ANSI operations
default, importance of changing, A-7	Oracle Virtual Private Database affect on,
administrative users	14-45
auditing, <i>27-12</i>	ANY system privilege
last successful login time, 3-51	guidelines for security, A-10
locked or expired accounts, 3-50	application common users
mandatorily audited, 28-2	about, 2-3
password complexity verification functions,	application containers
3-53	application contexts, 13-3
password files, managing, 3-51	Transport Layer Security, 23-2
password files, multitenant environment, 3-52	Virtual Private Database policies, 14-5
password management, 3-50	application contexts, 13-6, 13-28, 13-52
password profile limits, 3-51	about, <i>13-2</i>
administrator privileges	application containers, 13-3
access, A-14	as secure data cache, 13-3
operating system authentication, 3-58	benefits of using, 13-3
passwords, 3-59, A-7	bind variables, 14-4
SYSDBA and SYSOPER access, centrally	components, 13-2
controlling, 3-54	creating session based, 13-8
write, on listener.ora file, A-14	DBMS_SESSION.SET_CONTEXT
Advanced Encryption Standard (AES)	procedure, 13-13
about, <i>18-2</i>	driving context, 13-55
Advanced Networking Option (ANO) (Oracle	editions, affect on, 13-3
native encryption), 18-15	finding errors by checking trace files, 13-55
AES256 algorithm	finding information about, 13-55
converting to in Oracle wallets, <i>F-7</i>	global application contexts
alerts, used in fine-grained audit policy, 27-105	authenticating user for multiple
ALTER ANY LIBRARY statement	applications, 13-35
security guidelines, A-3	creating, 13-31
ALTER DATABASE DICTIONARY DELETE	logon trigger, creating, 13-15
CREDENTIALS statement, 12-17	Oracle Virtual Private Database, used with,
ALTER DATABASE DICTIONARY ENCRYPT	14-4
CREDENTIALS statement, 12-17	performance, 14-36
ALTER DATABASE DICTIONARY REKEY	policy groups, used in, 14-16
CREDENTIALS statement, 12-17	returning predicate, 14-4
ALTER PROCEDURE statement	session information, retrieving, 13-11
used for compiling procedures, 4-63	support for database links, 13-21
ALTER PROFILE statement	types, <i>13-4</i>
password management, 3-5	users, nondatabase connections, 13-29,
ALTER RESOURCE COST statement, 2-28, 2-29	13-37
ALTER ROLE statement	where values are stored, 13-2
changing authorization method, 4-37	

application contexts (continued)	audit files (continued)
See also client session-based application	operating system file
contexts, database session-based application	archiving, 28-11
contexts, global application contexts	standard audit trail
application developers	archiving, setting timestamp, 28-15
CONNECT role change, A-25	records, archiving, 28-12
application security	audit policies, 26-2
finding privilege use by users, 5-2	See also unified audit policies
restricting wallet access to current application,	audit policies, application contexts
10-8	about, <i>27-30</i>
revoking access control privileges from Oracle	appearance in audit trail, 27-32
wallets, <i>10-12</i>	configuring, 27-31
sharing wallet with other applications, 10-8	disabling, 27-31
specifying attributes, 13-9	examples, <i>27-32</i>
application users who are database users	audit records
Oracle Virtual Private Database, how it works	when written to OS files, 28-7
with, <i>14-49</i>	audit trail
applications	archiving, 28-12
about security policies for, 12-1	capturing syslog records, 28-6
database users, 12-2	capturing Windows Event Viewer records,
enhancing security with, 4-25	28-6
object privileges, 12-24	finding information about audit management,
object privileges permitting SQL statements,	28-25
12-25	finding information about usage, 27-111
One Big Application User authentication	SYSLOG records, 28-5
security considerations, 12-3	unified
security risks of, 12-2	archiving, 28-12
Oracle Virtual Private Database, how it works	AUDIT_ADMIN role, 4-29
with, <i>14-45</i>	AUDIT_VIEWER role, 4-29
password handling, guidelines, 12-4	auditing, <u>27-1</u>
password protection strategies, 12-3	administrators, Database Vault, 27-43
privileges, managing, <i>12-18</i>	audit options, 27-1
roles	audit trail, sensitive data in, A-19
multiple, 4-26	CDBs, 26-9
privileges, associating with database	committed data, A-20
roles, <i>12-22</i>	cursors, affect on auditing, 28-4
security, 4-42, 12-3	database user names, 3-64
security considerations for use, 12-2	Database Vault administrators, 27-43
security limitations, 14-45	databases, when unavailable, 28-8
security policies, 14-17	distributed databases and, 26-10
validating with security policies, 14-18	DV_ADMIN role user, 27-43
APPQOSSYS user account, 2-32	DV_OWNER role user, 27-43
architecture, 6-3	finding information about audit management,
archiving	28-25
operating system audit files, 28-11	finding information about usage, 27-111
standard audit trail, 28-12	fine-grained
timestamping audit trail, 28-15	See fine-grained auditing, 27-93
ASMSNMP user account, 2-32	functions, 27-17
asymmetric key operations, 17-12	functions, Oracle Virtual Private Database,
asynchronous authentication mode in RADIUS,	27-19
24-5	general steps
attacks	commonly used security-relevant
See security attacks	activities, 27-2
audit files	specific fine-grained activities, 27-2
operating system audit trail	SQL statements and other general
archiving, setting timestamp, 28-15	activities 27-1

auditing (continued)	auditing, purging records (continued)
general steps for, 27-1	creating audit trail
guidelines for security, A-19	purge job, 28-14
historical information, A-20	creating the purge job, 28-17
INHERIT PRIVILEGE privilege, 9-7	DBMS_SCHEDULER package, 28-14
keeping information manageable, A-19	deleting a purge job, 28-23
loading audit records to unified audit trail,	disabling purge jobs, 28-21
28-8	enabling purge jobs, 28-21
mandatory auditing, 28-2	general steps for, 28-13
multitier environments	purging audit trail manually, 28-19
See standard auditing, 27-23	roadmap, <i>28-13</i>
One Big Application User authentication,	scheduling the purge job, 28-17
compromised by, 12-2	setting archive timestamp, 28-15
operating-system user names, 3-64	time interval for named purge job, 28-22
Oracle Virtual Private Database policy	AUDSYS user account, 2-32
functions, 27-19	AUTHENTICATEDUSER role, 4-29
packages, 27-17	authentication, 3-3, 20-6
performance, 26-3	about, 3-2
PL/SQL packages, 27-17	administrators
predefined policies	operating system, 3-58
general steps for using, 27-2	passwords, 3-59
privileges required, 26-9	SYSDBA and SYSOPER access,
procedures, 27-17	centrally controlling, 3-54
purging records	by database, 3-60
example, 28-24	by SSL, <i>3-71</i>
general steps for manual purges, 28-14	client, <i>A-13</i>
general steps for scheduled purges,	client-to-middle tier process, 3-83
28-13	configuring multiple methods, 25-4
range of focus, 27-1	database administrators, 3-54
READ object privileges in policies, <i>27-21</i>	databases, using
READ privileges	about, <i>3-61</i>
about, 27-20	advantages, 3-62
how recorded in audit trail, 27-21	procedure, 3-62
recommended settings, <i>A-21</i>	directory service, 3-71
Sarbanes-Oxley Act	directory-based services, 3-67
auditing, meeting compliance through,	external authentication
26-2	about, 3-73
SELECT privileges	advantages, 3-73
about, 27-20	operating system authentication, 3-75
how recorded in audit trail, 27-21	user creation, 3-74
sensitive data, A-21	global authentication
suspicious activity, A-21	about, 3-70
traditional, 27-72	advantages, 3-72
triggers, 27-17	user creation for private schemas, 3-71
unified audit trail	user creation for shared schemas, 3-71
about, 26-4	methods, 20-3
VPD predicates	middle-tier authentication
fine-grained audit policies, 27-95	proxies, example, 3-85
unified audit policies, 27-18	modes in RADIUS, 24-3
when audit options take effect, 28-2	multitier, 3-75
when records are created, 28-2	network authentication
See also unified audit policies	third-party services, 3-66
auditing, purging records	Transport Layer Security, 3-66
about, 28-13	One Big Application User, compromised by,
cancelling archive timestamp, 28-23	12-2

authentication (continued)	С
operating system authentication, 3-68	
about, 3-64	CAPTURE_ADMIN role, 4-29
advantages, 3-64	cascading revokes, 4-79
disadvantages, 3-64	catpvf.sql script (password complexity functions),
operating system user in PDBs, 3-68	3-26
ORA-28040 errors, <i>3-40</i>	CDB common users
PDBs, 3-68	about, 2-3
proxy user authentication	plug-in operations, 2-4
about, 3-79	CDB_DBA role, 4-29
expired passwords, 3-82	CDBs, 2-3
public key infrastructure, 3-67	auditing
RADIUS, 3-67	how affects, 26-9
remote, A-13	traditional, 27-72
schema-only accounts, 3-63	CBAC role grants with DELEGATE option,
about, 3-63	9-15
altering, 3-64	common privilege grants, 4-14
creating users, 3-64	granting privileges and roles, 4-16
schema-only accounts, users created with,	local privilege grants, 4-14
3-63	object privileges, 4-15
specifying when creating a user, 2-9	PDB lockdown profiles, 4-45, 4-46
strong, A-7	privilege management, 4-13
SYSDBA on Windows systems, 3-58	privilege profiles, 5-4
Windows native authentication, 3-58	revoking privileges, <i>4-16</i> roles
See also passwords, proxy authentication AUTHENTICATION parameter, C-2	
authentication types, 6-4	altering, 4-37
AUTHID DEFINER clause	creating common, 4-21
used with Oracle Virtual Private Database	creating local, 4-21
functions, 14-4	granting common, 4-22
authorization	how common roles work, 4-19
about, 4-1	managing, 4-18
changing for roles, 4-37	privileges required to manage, 4-20 rules for creating common, 4-20
global	SYSLOG capture of unified audit records,
about, 3-70	28-6
advantages, 3-72	system privileges, 4-14
multitier, 3-75	transparent sensitive data protection, 15-4
omitting for roles, 4-34	user accounts
operating system, 4-39	creating, 2-14
roles, about, 4-37	local, 2-5
automatic reparse	user privileges, how affects, 4-3
Oracle Virtual Private Database, how it works	users
with, 14-46	CDB common, 2-3
Will, 17 70	common, 2-3
B	viewing information about, <i>4-16</i>
В	Virtual Private Database
banners	policies, 14-5
auditing user actions, configuring, 12-29	Center for Internet Security (CIS), 27-90
unauthorized access, configuring, 12-29	centrally managed users
BFILEs	Oracle Autonomous Database, 6-38
guidelines for security, <i>A-10</i>	certificate, 23-6
bind variables	certificate authority, 23-6
application contexts, used with, 14-4	certificate key algorithm
sensitive columns, 15-18	Transport Layer Security, A-17
BLOBS	certificate revocation list (CRL)
encrypting, 17-8	deleting, <i>F-14</i>

certificate revocation list (CRL) (continued)	client identifiers (continued)
displaying, <i>F-15</i>	DBMS_APPLICATION_INFO.SET_CLIENT_INFO,
displaying list of, F-16	3-91
hash value generation, F-15	global application context, independent of, 3-89
uploading, <i>F-16</i>	setting with DBMS_SESSION.SET_IDENTIFIER
certificate revocation lists, 23-6	procedure, 13-29
manipulating with orapki tool, 23-56	See also nondatabase users
uploading to LDAP directory, 23-56	client session-based application contexts, 13-52
where to store them, 23-52	about, 13-52
certificate revocation status checking	CLIENTCONTEXT namespace, clearing
disabling on server, 23-53, 23-55	value from, 13-54
certificate validation error message	CLIENTCONTEXT namespace, setting value
CRL could not be found, 23-61	in, 13-53
CRL date verification failed with RSA status,	retrieving CLIENTCONTEXT namespace, 13-53
23-61	See also application contexts
CRL signature verification failed with RSA	CLIENT_IDENTIFIER USERENV attribute, 3-89
status, 23-61	setting and clearing with DBMS_SESSION
Fetch CRL from CRL DP	package, 3-91
No CRLs found, 23-61	setting with OCI user session handle attribute,
OID hostname or port number not set, 23-61	3-90
certificates, 6-15	See also USERENV namespace
adding to wallet using orapki, <i>F-3</i>	CLIENTID_OVERWRITE event, 3-91
creating signed with orapki, <i>F-2</i>	code based access control (CBAC)
Oracle Real Application Clusters components	about, 9-11
that need certificates, 23-37 challenge-response authentication in RADIUS,	granting and revoking roles to program unit,
24-5	9-16
change_on_install default password, A-7	how works with definers rights, 9-14
character sets	how works with invoker's rights, 9-12
role names, multibyte characters in, 4-34	privileges, 9-12
role passwords, multibyte characters in, 4-38	tutorial, 9-17
Cipher Block Chaining (CBC) mode, defined, 18-2	column masking behavior, 14-14
cipher suites	column specification, 14-15
about, 23-16	restrictions, 14-15
authentication methods, 23-17	columns
data integrity, 23-17	granting privileges for selected, 4-75
encryption algorithms used by, 23-17	granting privileges on, 4-75
procedure for specifying for server, 23-18	INSERT privilege and, 4-75
TLS compatibility, 23-17	listing users granted to, 4-89
Transport Layer Security, A-17	privileges, 4-75
Transport Layer Security (TLS), C-4	pseudo columns
Cipher Suites	USER, 4-61
FIPS 140-2 settings, <i>E-4</i>	revoking privileges on, 4-78
client authentication in TLS, 23-20	command line recall attacks, 12-4, 12-6
client connections	committed data
guidelines for security, A-13	auditing, <i>A-20</i>
secure external password store, 3-46	common privilege grants
securing, A-13	about, <i>4-14</i>
client identifier	granting, 4-16
setting for applications that use JDBC, 3-90	revoking, 4-16
client identifiers, 13-29	with object privileges, 4-15
about, 3-88	with system privileges, 4-14
auditing users, 27-23	common roles
consistency between DBMS_SESSION.SET_IDENTIFIE	about, 4-19
and	auditing, 27-7
	creating, 4-21

common roles (continued)	CONTAINER_DATA objects
granting, 4-22	viewing information about, 4-16
how they work, 4-19	context profiles
privileges required to manage, 4-20	privilege analysis, 5-3
rules for creating, 4-20	controlled step-in procedures, 9-3
common user accounts	CPU time limit, 2-24
creating, 2-15	CREATE ANY LIBRARY statement
enabling access to other PDBs, 4-16	security guidelines, A-3
granting privileges to, 4-13	CREATE ANY PROCEDURE system privilege,
common users	4-62
accessing data in PDBs, 4-17	CREATE CONTEXT statement
altering, 2-18	example, <i>13-8</i>
configuration	CREATE LOCKDOWN PROFILE statement, 4-47
guidelines for security, A-12	CREATE PROCEDURE system privilege, 4-62
configuration files	CREATE PROFILE statement
Kerberos, <i>C-1</i>	password aging and expiration, 3-12
listener.ora, A-14	password management, 3-5
sample listener.ora file, A-14	passwords, example, 3-13
server.key encryption file, A-17	CREATE ROLE statement
tsnames.ora, A-17	IDENTIFIED EXTERNALLY option, 4-39
typical directory, A-17	CREATE SCHEMA statement
configuring	securing, 12-23
Kerberos authentication service parameters,	CREATE SESSION statement
22-5	CONNECT role privilege, A-7
RADIUS authentication, 24-8	securing, 12-23
thin JDBC support, 19-1	CREATE USER statement
TLS, 23-13	explicit account locking, 3-11
	•
on the client, 23-22	IDENTIFIED BY option, 2-9
on the server, 23-14	IDENTIFIED EXTERNALLY option, 2-9
CONNECT role	creating Oracle service directory user account,
about, A-23	6-7
applications	CRL, 23-6
account provisioning, A-24	CRLAdmins directory administrative group, <i>F-16</i>
affects of, A-24	CRLs
database upgrades, A-24	disabling on server, 23-53, 23-55
installation of, A-24	where to store them, 23-52
script to create, 4-29	cryptographic hardware devices, 23-8
users	cryptographic libraries
application developers, impact, A-25	FIPS 140-2, <i>E-1</i>
client-server applications, impact, A-25	CSW_USR_ROLE role, 4-29
general users, impact, A-25	CTXAPP role, 4-29
how affects, A-25	CTXSYS user account, 2-32
why changed, A-23	cursors
connecting	affect on auditing, 28-4
with username and password, 25-1	reparsing, for application contexts, 13-15
connection pooling	shared, used with Virtual Private Database,
about, <i>3-75</i>	14-4
finding unnecessarily granted privileges, 5-2	CWM_USER role, 4-29
global application contexts, 13-29	
nondatabase users, 13-37	D
proxy authentication, 3-83	D
container data objects	data definition language (DDL)
about, <i>4-16</i>	roles and privileges, 4-28
container database (CDB)	data dictionary
See CDBs	about, 16-1
	data dictionary views, 16-6
	data dictionally views, 10-0

data dictionary (continued)	database links (continued)
deleting, 16-4	sensitive credential data (continued)
encrypting sensitive information in, 16-1–16-6	restoring functioning of after lost keystore,
multitenant environment, 16-2	16-5
procedure, 16-2	session-based application contexts,
protecting, A-10	accessing, 13-13
rekeying, 16-3	database session-based application contexts,
restoring lost keystore, 16-5	13-6
data encryption and integrity parameters	about, <i>13-6</i>
about, B-3	cleaning up after user exits, 13-6
SQLNET.CRYPTO_CHECKSUM_CLIENT, B-6	components, 13-7
SQLNET.CRYPTO_CHECKSUM_SERVER, B-5	database links, 13-13
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT,	dynamic SQL, 13-12
B-8	externalized, using, 13-27
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER,	how to use, 13-5
B-7	
	initializing externally, 13-21
SQLNET ENCRYPTION_CLIENT, B-5	initializing globally, 13-23
SQLNET.ENCRYPTION_SERVER, B-4	ownership, 13-8
SQLNET.ENCRYPTION_TYPES_CLIENT, B-7	parallel queries, 13-12
SQLNET.ENCRYPTION_TYPES_SERVER, B-6	PL/SQL package creation, 13-9
Data Encryption Standard (DES)	session information, setting, 13-13
DES40 encryption algorithm, 18-2	SYS_CONTEXT function, 13-11
Triple-DES encryption algorithm, 18-2	trusted procedure, 13-2
data files, A-10	tutorial, 13-17
guidelines for security, A-10	See also application contexts
data manipulation language (DML)	database upgrades and CONNECT role, A-24
privileges controlling, 4-59	databases
data security	access control
encryption, problems not solved by, 17-3	password encryption, 3-3
database administrators (DBAs)	additional security products, 1-2
access, controlling, 17-2	authentication, 3-60
authentication, 3-54	database user and application user, 12-2
malicious, encryption not solved by, 17-2	default password security settings, 3-8
Database Configuration Assistant (DBCA)	DBCA-created databases, 3-8
default passwords, changing, A-7	manually-created databases, 3-8
user accounts, automatically locking and	default security features, summary, 1-1
expiring, A-3	granting privileges, 4-70
database links, 6-5	granting roles, 4-70
application context support, 13-21	limitations on usage, 2-23
application contexts, 13-13	schema-only accounts, 3-63
authenticating with Kerberos, 3-66	security and schemas, 12-23
authenticating with third-party services, 3-66	security embedded, advantages of, 12-3
definer's rights procedures, 9-22	security policies based on, 14-3
global user authentication, 3-72	DATAPUMP_EXP_FULL_DATABASE role, 4-29
object privileges, 4-51	DATAPUMP IMP FULL DATABASE role, 4-29
operating system accounts, care needed,	DBA role
3-64	about, <i>4-29</i>
Oracle DBaaS-to-IAM connections, 7-36	DBA_CONTAINER_DATA data dictionary view,
RADIUS not supported, 24-1	4-16
sensitive credential data	DBA ROLE PRIVS view
about, 16-1	application privileges, finding, 12-18
data dictionary views, 16-6	DBA_ROLES data dictionary view
deleting, 16-4	PUBLIC role, 4-13
encrypting, 16-2	DBFS_ROLE role, 4-29
multitenant environment, 16-2	DBMS_CREDENTIAL.CREATE_CREDENTIAL
rekeying, 16-3	procedure, 12-14
roncynig, 10 0	p. 000 0 0 1 2 2 7

DBMS_CRYPTO package	default profiles
asymmetric key operations, 17-12	about, 3-6
data encryption storage, 17-9	default realms
examples, 17-15	ORA_DV_AUDPOL2 predefined audit policy
supported cryptographic algorithms, 17-9	for, 27-92
DBMS_CRYPTO PL/SQL package	default roles
enabling for FIPS 140-2, <i>E-2</i>	setting for user, 2-17
DBMS_FGA package	specifying, 4-85
about, 27-98	default users
ADD_POLICY procedure, 27-99	accounts, A-3
DISABLE_POLICY procedure, 27-103	Enterprise Manager accounts, A-3
DROP_POLICY procedure, 27-104	passwords, A-7
editions, 27-98	defaults
ENABLE_POLICY procedure, 27-103	tablespace quota, 2-11
PDBs, <u>27-99</u>	user tablespaces, 2-9
DBMS_NETWORK_ACL_ADMIN.REMOVE_HOS	definer's rights
T_ACE procedure, 10-6	about, 9-2
DBMS_PRIVILEGE_CAPTURE PL/SQL package,	code based access control
5-5	about, 9-11
DBMS_RLS.ADD_POLICY	granting and revoking roles to program
sec_relevant_cols parameter, 14-12	unit, 9-16
sec_relevant_cols_opt parameter, 14-15	how code based access control works,
DBMS_RLS.ADD_POLICY procedure	9-14
transparent sensitive data protection polices,	compared with invoker's rights, 9-1
15-23	example of when to use, 9-2
DBMS_SESSION package	procedure privileges, used with, 9-2
client identifiers, using, 3-91	procedure security, 9-2
global application context, used in, 13-32	schema privileges for, 9-2
SET_CONTEXT procedure	secure application roles, 12-20
about, <i>13-13</i>	used with Oracle Virtual Private Database
DBMS_SESSION.SET_CONTEXT procedure	functions, 14-4
about, <i>13-13</i>	views, 9-8
syntax, <i>13-13</i>	definer's rights, database links
username and client_id settings, 13-33	revokes of INHERIT [ANY] REMOTE
DBMS_SESSION.SET_IDENTIFIER procedure	PRIVILEGES, 9-24
client session ID, setting, 13-29	grants of INHERIT ANY REMOTE
DBMS_APPLICATION.SET_CLIENT_INFO	PRIVILEGES, 9-24
value, overwritten by, 3-91	grants of INHERIT ANY REMOTE
DBSFWUSER user account, 2-32	PRIVILEGES on connected user to
DBSNMP user account	current user, example, 9-23
about, 2-32	grants of INHERIT REMOTE PRIVILEGES to
password usage, A-7	other users, 9-23
DDL	revoking INHERIT REMOTE PRIVILEGES
See data definition language	from PUBLIC, example, 9-25
debugging	revoking INHERIT REMOTE PRIVILEGES on
Java stored procedures, 10-21	connecting user from procedure
PL/SQL stored procedures, 10-21	owner, example, 9-25
default command rules	tutorial, 9-26
ORA_DV_AUDPOL2 predefined audit policy	definers's rights, database links
for, 27-92	about, 9-22
default passwords, A-7	ORA-25433 error, 9-22
change_on_install or manager passwords,	denial of service (DoS) attacks
A-7	about, 8
changing, importance of, 3-6	denial-of-service (DoS) attacks
finding, 3-6	bad packets, preventing, 12-26
default permissions, A-10	networks, securing, A-14

denial-of-service (DoS) attacks (continued) password concurrent guesses, 3-3	<u>E</u>
Department of Defense Database Security	ECB ciphertext encryption mode, 17-12
Technical Implementation Guide, 3-27	editions
dictionary tables	application contexts, how affects, 13-3
auditing, 27-15	fine-grained auditing packages, results in,
Diffie-Hellman, 23-16	13-32
Diffie-Hellman key negotiation algorithm, 18-7 DIP user account, 2-35	global application contexts, how affects, 13-32 Oracle Virtual Private Database packages,
direct path load	results in, 13-32
fine-grained auditing effects on, 27-93	EJBCLIENT role, 4-29
directories	EM_EXPRESS_ALL role, 4-29
auditing, <i>27-14</i>	EM_EXPRESS_BASIC role, 4-29
directory authentication, configuring for SYSDBA	email alert example, 27-105
or SYSOPER access, 3-54	encrypting information in, 16-1
directory objects	encryption
granting EXECUTE privilege on, 4-71	access control, 17-1
directory-based services authentication, 3-67	BLOBS, 17-8
disabling unnecessary services	challenges, 17-4
FTP, TFTP, TELNET, A-14	data security, problems not solved by, 17-3
dispatcher processes (Dnnn)	data transfer, A-14
limiting SGA space for each session, 2-25	deleted encrypted data, A-10
distributed databases	examples, 17-15
auditing and, 26-10	finding information about, 17-20
DML	indexed data, 17-4
See data manipulation language	key generation, 17-5
driving context, 13-55	key storage, 17-6
DROP PROFILE statement	key transmission, 17-5
example, <i>2-29</i>	keys, changing, 17-8
DROP ROLE statement	malicious database administrators, 17-2
example, <i>4-42</i>	network encryption, 18-7
security domain, affected, 4-42	network traffic, A-14
DROP USER statement	problems not solved by, 17-1
about, <i>2-30</i>	Transparent Data Encryption, 17-8
schema objects of dropped user, 2-31	transparent tablespace encryption, 17-8
dsi.ora file	encryption and checksumming
about, <i>6-11</i>	activating, 18-8
changing contents of, 6-11	applying patch <number>, 18-5</number>
compared with Idap.ora, 6-10	negotiating, 18-9
multitenant environment, 6-11	parameter settings, 18-11
placement of, 6-11	patching to use improved algorithms, 18-4
search order for, 6-11	encryption of data dictionary sensitive data, 16-1
WALLET_LOCATION parameter and, 6-11	ENFORCE_CREDENTIAL configuration
when to use, 6-11	parameter
DVF schema	security guideline, A-18
ORA_DV_AUDPOL predefined audit policy	enterprise directory service, 4-40
for, 27-92	enterprise roles, 3-70, 4-40
DVSYS schema	enterprise user management, 12-2
ORA_DV_AUDPOL predefined audit policy	Enterprise User Security
for, 27-92	application context, globally initialized, 13-25
dynamic Oracle Virtual Private Database policy	proxy authentication
types, <i>14-20</i>	Oracle Virtual Private Database, how it
DYNAMIC policy type, 14-20	works with, <i>14-49</i>
	enterprise users
	centralized management, 3-70
	global role, creating, 4-40

enterprise users (continued)	examples (continued)
One Big Application User authentication,	roles
compromised by, 12-2	altering for external authorization, 4-37
proxy authentication, 3-79	creating for application authorization, 4-38
shared schemas, protecting users, 12-24	creating for external authorization, 4-39
error messages	creating for password authorization, 4-35,
ORA-12650, <i>18-8</i> , <i>18-10</i> , <i>B-6–B-8</i>	4-36
ORA-25433, 9-22	default, setting, 4-85
errors	external, 4-36
ORA-00036, <i>27-99</i>	global, <i>4-36</i>
ORA-01720, 4-60	using SET ROLE for password-
ORA-01994, <i>2-21</i>	authenticated roles, 4-38
ORA-06512, <i>10-21</i> , <i>27-109</i>	views, 4-86
ORA-06598, 9-6	secure external password store, 3-45
ORA-1000, <i>27-99</i>	session ID of user
ORA-1536, <i>2-12</i>	finding, 2-30
ORA-24247, 10-3, 10-21, 27-109	system privilege and role, granting, 4-71
ORA-28017, <i>2-21</i>	tablespaces
ORA-28040, <i>3-40</i> , <i>3-61</i>	assigning default to user, 2-10
ORA-28046, <i>2-21</i>	quota, assigning to user, 2-11
ORA-28144, 27-99	temporary, 2-13
ORA-28575, <i>12-14</i>	type creation, 4-67
ORA-45622, <i>15-12</i>	users
example, basic, 27-64	account creation, 2-6
example, comparison, 27-64	creating with GRANT statement, 4-71
examples, 13-17, 14-26	dropping, 2-31
access control lists	middle-tier server proxying a client, 3-81
external network connections, 10-13	object privileges granted to, 4-72
wallet access, 10-13	proxy user, connecting as, 3-81
account locking, 3-10	See also tutorials
audit trail, purging unified trail, 28-24	exceptions
auditing GRANT operations, 27-15	WHEN NO DATA FOUND, used in application
auditing REVOKE operations, 27-15	context package, 13-18
auditing user SYS, 27-10	WHEN OTHERS, used in triggers
data encryption	development environment (debugging)
encrypting and decrypting BLOB data,	example, 13-16
17-17	production environment example, 13-16
encrypting and decrypting procedure with	Exclusive Mode
AES 256-Bit, 17-16	SHA-2 password hashing algorithm, enabling,
directory objects, granting EXECUTE privilege	3-38
on, 4-71	EXECUTE ANY LIBRARY statement
encrypting procedure, 17-15	security guidelines, A-3
Java code to read passwords, 12-8	EXECUTE_CATALOG_ROLE role
locking an account with CREATE PROFILE,	SYS schema objects, enabling access to,
3-10	4-11
login attempt grace period, 3-13	EXEMPT ACCESS POLICY privilege
nondatabase user authentication, 13-37	Oracle Virtual Private Database
passwords	enforcements, exemption, 14-47
aging and expiration, 3-13	EXP_FULL_DATABASE role
changing, 2-20	about, 4 -29
creating for user, 2-9	expiring a password
privileges	explicitly, 3-13
granting ADMIN OPTION, 4-71	exporting data
views, 4-86	direct path export impact on Oracle Virtual
procedure privileges affecting packages, 4-64	Private Database, 14-47
profiles, assigning to user, <i>2-13</i>	policy enforcement, 14-47
r. 555, 4.55.g /g to 4.55/, = ±0	

extended data objects	files (continued)
views and Virtual Private Database, 14-10	server.key encryption file, A-17
external authentication	symbolic links, restricting, A-10
about, 3-73	tnsnames.ora, A-17
advantages, 3-73	fine-grained access control
network, 3-75	See Oracle Virtual Private Database (VPD)
operating system, 3-75	fine-grained auditing
user creation, 3-74	about, 27-93
external network services	alerts, adding to policy, 27-105
enabling listener for, 10-6	archiving audit trail, 28-12
external network services, fine-grained access to	columns, specific, 27-101
See access control list (ACL)	DBMS_FGA package, 27-98
external network services, syntax for, 10-4	direct loads of data, 27-93
external procedures	edition-based redefinitions, 27-97
configuring extproc process for, 12-14	editions, results in, 13-32
credentials, 12-12	
	finding errors by checking trace files, 27-111
DBMS_CREDENTIAL.CREATE_CREDENTIAL	how audit records are generated, 27-95
procedure, 12-14	how to use, 27-93
legacy applications, 12-16	policies
security guideline, A-18	adding, 27-99
external roles, 4-36	disabling, 27-103
external tables, A-10	dropping, 27-104
extproc process	enabling, <i>27-103</i>
about, <i>12-12</i>	modifying, 27-99
configuring credential for, 12-14	policy creation syntax, 27-99
legacy applications, 12-16	privileges required, 27-95
	records
F	archiving, 28-12
I	transparent sensitive data protection policy
failed login attempts	settings, 15-31
account locking, 3-10	TSDP policies and, 15-31
password management, 3-10	VPD predicates, 27-95
resetting, 3-10	FIPS 140-2 cryptographic libraries
fallback authentication, Kerberos, 22-21	about, <i>E-1</i>
	native network encryption, <i>E-5</i>
Federal Information Processing Standard (FIPS)	FIPS parameter
DBMS_CRYPTO package, <i>E-2</i> FIPS 140-2	configuring, <i>E-3</i>
	fips.ora file, <i>E-3</i> , <i>E-5</i>
Cipher Suites, <i>E-4</i>	firewalls
postinstallation checks, <i>E-6</i>	advice about using, A-14
SQLNET.FIPS_140, <i>E-5</i>	database server location, A-14
SSLFIPS_140, E-3	
SSLFIPS_LIB, E-3, E-5	ports, A-17
verifying connections for	supported types, A-14
DBMS_CRYPTO, E-7	flashback query
verifying connections for network native	Oracle Virtual Private Database, how it works
encryption, <i>E-7</i>	with, 14-46
verifying connections for TLS, <i>E-6</i>	foreign keys
Transparent Data Encryption, <i>E-2</i>	privilege to use parent key, 4-59
files	FTP service, A-14
BFILEs	functions
operating system access, restricting, A-10	auditing, 27-14, 27-17
BLOB, 17-8	granting roles to, 4-41
keys, 17-7	Oracle Virtual Private Database
listener.ora file	components of, 14-6
guidelines for security, A-14, A-17	privileges used to run, 14-4
restrict listener access, A-14	privileges for, 4-62
. 550100 11000101 400000, 71 17	,

functions (continued)	gradual database password rollover (continued)
roles, 4-27	changing password during rollover period, 3-21
G	changing password to begin rollover period, 3-20
GATHER_SYSTEM_STATISTICS role, 4-29	enabling, 3-19
global application contexts, 13-28	finding users who use old passwords, 3-24
about, 13-28	manually ending the password before rollover
authenticating nondatabase users, 13-37	period, 3-22
checking values set globally for all users,	Oracle Data Guard, 3-24
13-34	Oracle Data Pump exports, 3-24
clearing values set globally for all users,	password change life cycle, 3-18
13-34	passwords, compromised, 3-23
components, 13-29	server behavior after rollover ends, 3-23
editions, affect on, 13-32	GRANT ALL PRIVILEGES statement
example of authenticating nondatabase users,	SELECT ANY DICTIONARY privilege,
13-37	exclusion of, A-10
example of authenticating user moving to	GRANT ANY PRIVILEGE system privilege, 4-12
different application, 13-35	GRANT CONNECT THROUGH clause
example of setting values for all users, 13-34	consideration when setting
Oracle RAC environment, 13-30	FAILED_LOGIN_ATTEMPTS
Oracle RAC instances, 13-28	parameter, 3-6
ownership, 13-30	for proxy authorization, 3-81
PL/SQL package creation, 13-32	GRANT statement, 4-70
process, lightweight users, 13-49	ADMIN OPTION, 4-71
process, standard, 13-48	creating a new user, 4-71
sharing values globally for all users, 13-34	object privileges, 4-72, 12-24
system global area, 13-28	system privileges and roles, 4-70
tutorial for client session IDs, 13-44	when takes effect, 4-84
used for One Big Application User scenarios,	WITH GRANT OPTION, 4-73
14-49	granting privileges and roles
uses for, 14-49	about, <i>4-12</i>
See also application contexts	specifying ALL, 4-53
global authentication	GSMROOTUSER user account, 2-32
about, 3-70	guidelines
advantages, 3-72	handling compromised passwords, 3-23
user creation for private schemas, 3-71	guidelines for security
user creation for shared schemas, 3-71	auditing, A-19
global authorization	custom installation, A-12
about, 3-70	data files and directories, A-10
advantages, 3-72	encrypting sensitive data, A-10
role creation, 4-40	guidelines for security
roles, 3-70	custom installation, A-12
global roles, 4-36	installation and configuration, A-12
about, 4-40	networking security, A-13
global users, 3-70	operating system accounts, limiting privileges,
GLOBAL_AQ_USER_ROLE role, 4-29	A-10
GLOBAL_EXTPROC_CREDENTIAL	operating system users, limiting number of,
configuration parameter	A-10
security guideline, 12-16	Oracle home default permissions, disallowing
grace period for login attempts	modification, A-10
example, 3-13	ORACLE_DATAPUMP access driver, A-11
grace period for password expiration, 3-13	passwords, A-7
gradual database password rollover	products and options
about, 3-17	install only as necessary, A-12
actions permitted during, 3-22	sample schemas, A-12

guidelines for security (continued)	initialization parameter file (continued)
Sample Schemas	parameters for clients and servers using
remove or relock for production, A-12	RADIUS, C-8
test database, A-12	parameters for clients and servers using TLS,
symbolic links, restricting, A-10	C-2
Transport Layer Security	initialization parameters
mode, <i>A-17</i>	application protection, 12-26
TCPS protocol, A-17	MAX_ENABLED_ROLES, 4-85
user accounts and privileges, A-3	OS_AUTHENT_PREFIX, 3-74
•	OS_ROLES, <u>4-39</u>
Н	SEC_MAX_FAILED_LOGIN_ATTEMPTS, 12-28
<u></u>	SEC_RETURN_SERVER_RELEASE_BANNER,
hackers	12-28
See security attacks	SEC_USER_AUDIT_ACTION_BANNER, 12-29
handshake	SEC_USER_UNAUTHORIZED_ACCESS_BANNER
TLS, 23-4	12-29
how it works, 6-3	INSERT privilege
HR user account, 2-36	granting, 4-75
HS_ADMIN_EXECUTE_ROLE role	revoking, 4-78
about, 4-29	installation
HS_ADMIN_ROLE role	guidelines for security, A-12
about, <u>4-29</u>	intruders
HS_ADMIN_SELECT_ROLE role	See security attacks
about, 4-29	invoker's rights
HTTP authentication	about, 9-3
See access control lists (ACL), wallet access	code based access control
HTTP verifier removal, A-7	about, 9-11
HTTPS	granting and revoking roles to program
port, correct running on, A-17	unit, 9-16
	how code based access control works,
I	9-12
<u> </u>	tutorial, 9-17
IMP_FULL_DATABASE role	compared with definer's rights, 9-1
about, 4-29	controlled step-in, 9-3
inactive user accounts, locking automatically, 3-9	procedure privileges, used with, 9-2
INACTIVE ACCOUNT TIME profile parameter,	procedure security, 9-3
3-9	secure application roles, 12-20
indexed data	secure application roles, requirement for
encryption, 17-4	enabling, <i>12-20</i>
indirectly granted roles, 4-24	security risk, 9-5
INHERIT ANY PRIVILEGES privilege	views
about, 9-6	about, 9-8
managing, 9-7	finding user who invoked invoker's right
revoking from powerful users, 9-7	view, 9-10
when it should be granted, 9-7	IP addresses
INHERIT ANY REMOTE PRIVILEGES, 9-22	falsifying, <i>A-14</i>
INHERIT PRIVILEGES privilege	IX user account, 2-36
about, 9-6	
auditing, 9-7	J
managing, 9-7	<u> </u>
when it should be granted, 9-6	Java Byte Code Obfuscation, 19-3
INHERIT REMOTE PRIVILEGES	Java Database Connectivity (JDBC)
about, 9-22	configuration parameters, 19-4
initialization parameter file	Oracle extensions, 19-1
parameters for clients and servers using	thin driver features, 19-2
Kerheros C-1	

Java Debug Wire Protocol (JDWP)	key transmission
network access for debugging operations,	encryption, 17-5
10-21	kinstance (Kerberos), 22-3
Java schema objects	kservice (Kerberos), 22-3
auditing, <i>27-14</i>	
Java stored procedures	1
network access for debugging operations,	L
10-21	large objects (LOBs)
JAVA_ADMIN role, 4-29	about securing, 12-16
JAVA_RESTRICT initialization parameter	encryption management, 12-17
security guideline, <i>A-10</i>	LBAC DBA role, 4-29
JAVADEBUGPRIV role, 4-29	LBACSYS schema
JAVAIDPRIV role, 4-29	ORA_DV_AUDPOL predefined audit policy
JAVASYSPRIV role, 4-29	for, 27-92
JAVAUSERPRIV role, 4-29	LBACSYS user account, 2-32
JDBC	LBACSYS.ORA_GET_AUDITED_LABEL function
See Java Database Connectivity	about, 27-56
JDBC connections	
JDBC Thin Driver proxy authentication	Idap.ora
configuring, 3-79	which directory SSL port to use for no
with real user, 3-83	authentication, 23-58
JDBC/OCI proxy authentication, 3-79	Idap.ora file
multiple user sessions, 3-83	about, 6-14
Oracle Virtual Private Database, 14-49	benefit of, 6-14
JDeveloper	changing contents of, 6-14
debugging using Java Debug Wire Protocol,	compared with dsi.ora, 6-10
10-21	creating for Microsoft Active Directory
JMXSERVER role, 4-29	services, 6-13, 6-15
SWINGERVER FOIC, 4 25	placement of, 6-14
	search order for, 6-14
K	least privilege principle, A-3
Kerberos, <i>20-4</i>	about, A-3
authentication adapter utilities, 22-11	granting user privileges, A-3
authentication fallback behavior, 22-21	middle-tier privileges, 3-84
configuring authentication, 22-1, 22-5	libraries
configuring address configuring for database server, 22-3	auditing, 27-14
configuring for Windows 2008 Domain	lightweight users
•	example using a global application context,
Controller KDC, 22-16	13-44
connecting to database, 22-15 interoperability with Windows Server Domain	Lightweight Directory Access Protocol
Controller KDC, 22-16	(LDAP), <i>14-36</i>
kinstance, 22-3	listener
	endpoint
kservice, 22-3	TLS configuration, 23-22
realm, 22-3	not an Oracle owner, <i>A-14</i>
sqlnet.ora file sample, <i>B-1</i>	preventing online administration, A-14
system requirements, 20-7	restrict privileges, A-14
Kerberos authentication, 3-66	secure administration, A-14
configuring for SYSDBA or SYSOPER	listener.ora file
access, 3-55	administering remotely, A-14
password management, A-7	default location, A-17
Kerberos Key Distribution Center (KDC), 22-16	FIPS 140-2 Cipher Suite settings, <i>E-4</i>
key generation	online administration, preventing, <i>A-14</i>
encryption, 17-5	Oracle wallet setting, C-7
key storage	TCPS, securing, A-17
encryption, 17-6	lists data dictionary
	data dictionary views

lists data dictionary <i>(continued)</i>	memory
See views	users, viewing, 2-40
granting privileges and roles	MERGE INTO statement, affected by
finding information about, 4-86	DBMS_RLS.ADD_POLICY
privileges, 4-10	statement_types parameter, 14-11
finding information about, 4-86	metadata links
roles, 12-20	privilege management, 4-56
finding information about, 4-86	methods
views, 4-86	privileges on, 4-65
privileges, 4-60, 4-86	Microsoft Active Directory services, 6-3-6-7, 6-15
roles, <i>4-86</i>	6-18, 6-19
LOB_SIGNATURE_ENABLE initialization	about configuring connection, 6-18
parameter, 12-16	about password authentication, 6-23
LOBs	access configuration, Oracle wallet
about securing, 12-16	verification, 6-21
encryption management, 12-17	access configuration, testing integration, 6-22
local privilege grants	access, Kerberos authentication, 6-26
about, 4-14	access, PKI authentication, 6-27
granting, 4-16	account policies, 6-38
revoking, 4-16	administrative user configuration, exclusive
local roles	mapping, 6-31
about, 4-19	administrative user configuration, shared
creating, 4-21	access accounts, 6-31
rules for creating, 4-21	dsi.ora file, about, 6-11
local user accounts	dsi.ora file, compared with Idap.ora, 6-10
creating, 2-16	extending Active Directory schema, 6-8
local users	Idap.ora file, about, 6-14
about, 2-5	Idap.ora file, compared with dsi.ora, 6-10
lock and expire	Idap.ora file, creating, 6-13, 6-15
default accounts, A-3	logon user name with password
predefined user accounts, A-3	authentication, 6-25
lockdown profiles, PDB, 4-45	user authorization, about, 6-28
locking inactive user accounts automatically, 3-9	user authorization, mapping Directory user
log files	group to global role, 6-30
owned by trusted user, A-10	user authorization, verifying, 6-32
logical reads limit, 2-24	user management, altering mapping
logon triggers	definition, 6-30
externally initialized application contexts,	user management, exclusively mapping
13-15	Directory user to database global
for application context packages, 13-15	user, 6-30
running database session application context	
package, 13-15	user management, mapping group to shared
secure application roles, 4-44	global user, 6-29 user management, migrating mapping
LOGSTDBY ADMINISTRATOR role, 4-29	definition, 6-30
20031DB1_ADMINISTRATOR Tole, 4 23	
	Microsoft Active Directory services integration,
M	6-2, 6-3, 6-5
	Microsoft Azure AD token
malicious database administrators, 17-2	checking version of, 8-37
See also security attacks	Microsoft Directory Access services, 6-20
manager default password, A-7	Microsoft Windows
managing roles with RADIUS server, 24-20	Kerberos
materialized views	configuring for Windows 2008 Domain
auditing, 27-14	Controller KDC, 22-16
MD5 message digest algorithm, 18-6	middle-tier systems
MDDATA user account, 2-35	client identifiers, 3-88
MDSYS user account, 2-32	enterprise user connections, 3-87

middle-tier systems (continued)	network IP addresses
password-based proxy authentication, 3-86	guidelines for security, A-14
privileges, limiting, 3-84	network traffic encryption, A-14
proxies authenticating users, 3-85	nondatabase users, 13-28, 13-29
proxying but not authenticating users, 3-85	about, 13-29
reauthenticating user to database, 3-86	auditing, <i>27-84</i>
USERENV namespace attributes, accessing,	clearing session data, 13-39
13-23	creating session data, 19-99 creating client session-based application
	contexts, 13-52
mining models	
auditing, 27-14	global application contexts
mixed mode auditing capabilities, 26-8	package example, 13-37
monitoring user actions, 26-2	reason for using, 13-29
See also auditing, standard auditing, fine-	setting, 13-37
grained auditing multiplex multiple-client network sessions, <i>A-14</i>	tutorial, <i>13-44</i>
	One Big Application User authentication
multitenant container database (CDB)	about, <i>14-49</i>
See CDBs	features compromised by, 12-2
multitenant option, 6-5	security risks, 12-2
My Oracle Support, A-2	Oracle Virtual Private Database
security patches, downloading, A-2	how it works with, 14-49
user account for logging service requests,	tutorial for creating a policy group, 14-38
2-35	See also application contexts, client identifiers
N	0
native network encryption	obfuscation, 19-3
compared with Transport Layer Security, 18-3	object privileges, 4-51, A-3
FIPS 140-2, <i>E-5</i>	about, 4-51
FIPS library location setting (SSLFIPS_LIB),	granting on behalf of the owner, 4-73
E-5	managing, 12-24
FIPS mode setting (FIPS_140), <i>E-5</i>	revoking, 4-76
native network enryption	
disabling, 25-2	revoking on behalf of owner, 4-77
nCipher hardware security module	schema object privileges, <i>4-51</i>
using Oracle Net tracing to troubleshoot,	synonyms, 4-55
23-66	with common privilege grants, 4-15
	See also schema object privileges
Net8 See Oracle Net	object types
Netscape Communications Corporation, 23-2	auditing, 27-14
network authentication	objects
external authentication, 3-75	applications, managing privileges in, 12-24
	granting privileges, 12-25
guidelines for securing, A-7	privileges
roles, granting using, 4-81	applications, 12-24
smart cards, A-7	managing, 4-65
third-party services, 3-66	protecting in shared schemas, 12-24
token cards, A-7	protecting in unique schemas, 12-23
Transport Layer Security, 3-66	SYS schema, access to, 4-11
X.509 certificates, A-7	OE user account, 2-36
network connections	OEM_ADVISOR role, 4-29
denial-of-service (DoS) attacks, addressing,	OEM_MONITOR role, 4-29
A-14	OFB ciphertext encryption mode, 17-12
guidelines for security, A-13, A-14	OJVMSYS user account, 2-32
securing, A-14	okcreate
network encryption	Kerberos adapter utility, 22-11
about, 18-7	okcreate options, 22-14
configuring, 18-7	·

okdstry	ORA-01017 errors in Oracle DBaaS-IAM
Kerberos adapter utility, 22-11	integration
okdstry options, 22-14	client-side, 7-37
okinit	IAM administrator actions to remedy, 7-42
Kerberos adapter utility, 22-11	IAM user configurations, 7-41
okinit utility options, 22-12	ORA-01720 error, 4-60
oklist	ORA-01994, <i>2-21</i>
Kerberos adapter utility, 22-11	ORA-03114 error, 7-42, 8-37
OLAP_DBA role, 4-29	ORA-06512 error, 10-21, 27-109
OLAP_USER role, 4-29	ORA-06598 error, 9-6
OLAP_XS_ADMIN role, 4-29	ORA-12599 error, 7-42, 8-37
OLAPSYS user account, 2-32	ORA-12650 error, <i>B-7</i>
One Big Application User authentication	ORA-1536 error, 2-12
See nondatabase users	ORA-24247 error, 10-3, 10-21, 27-109
openssl utility	ORA-28017 error, 2-21
creating PKCS#12 wallet with, F-5	ORA-28040 error, 3-40, 3-61
operating system	ORA-28046 error, 2-21
audit files written to, 28-7	ORA-28575 error, 12-14
operating system users	ORA-29024 error, 10-13
configuring for PDBs, 3-69	ORA-40300 error, 23-67
operating systems, 3-68	ORA-40301 error, 23-67
accounts, 4-82	ORA-40302 error, 23-67
authentication	ORA-45622 errors, 15-12
about, <i>3-64</i>	ORA-64219: invalid LOB locator encountered,
advantages, 3-64	12-16
disadvantages, 3-64	ORA\$DEPENDENCY profile, 5-4
external, 3-75	Oracle Advanced Security
operating system user for PDB, 3-68	checksum sample for sqlnet.ora file, B-1
roles, using, 4-81	configuration parameters, 19-4
default permissions, A-10	encryption sample for sqlnet.ora file, <i>B-1</i>
enabling and disabling roles, 4-83	Java implementation, 19-3
operating system account privileges, limiting,	network authentication services, A-7
A-10	TLS features, 23-3
role identification, 4-82	user access to application schemas, 12-24
roles and, 4-29	Oracle Audit Vault and Database Firewall
roles, granting using, 4-81	schema-only accounts, 3-63
users, limiting number of, A-10	Oracle Autonomous Database
OPTIMIZER_PROCESSING_RATE role, 4-29	centrally managed users, 6-38
ORA_ACCOUNT_MGMT predefined unified audit	Oracle Call Interface (OCI)
policy, <i>27-90</i>	application contexts, client session-based,
ORA_CIS_RECOMMENDATIONS predefined	13-52
unified audit policy, 27-90	proxy authentication, 3-79
ORA_DATABASE_PARAMETER predefined	Oracle Virtual Private Database, how it
unified audit policy, 27-89	works with, 14-49
ORA_DV_AUDPOL predefined unified audit	proxy authentication with real user, 3-83
policy, 27-92	security-related initialization parameters,
ORA_DV_AUDPOL2 predefined unified audit	12-26
policy, 27-92	Oracle Connection Manager
ORA_LOGON_FAILURES predefined unified	securing client networks with, A-14
audit policy, 27-88	Oracle Data Guard
ORA_SECURECONFIG predefined unified audit	gradual database password rollover, 3-24
policy, 27-88	SYSDG administrative privilege, 4-7
ORA_STIG_PROFILE profile, 3-27	Oracle Data Mining
ORA-01017 errors in Oracle Cloud Infrastructure-	audit events, 27-57
IAM integration, 7-40	Oracle Data Pump
	audit events, 27-60

Oracle Data Pump (continued)	Oracle Database-to-Microsoft Azure Active
exported data from VPD policies, 14-49	directory client connections
exports during gradual database password	direct token retrievals, 8-32
rollover, 3-24	Oracle Database-to-Microsoft Azure Active
unified audit trail, 28-9	Directory client connections
Oracle Database Enterprise User Security	about, 8-19
password security threats, 3-37	confidential client registration, 8-21
Oracle Database Real Application Clusters	creating a client app registration, 8-21
archive timestamp for audit records, 28-15	curl used to get Azure AD token, 8-25
global contexts, 13-28	MSAL in Python, 8-25
Oracle Database Real Application Security	network proxies, 8-28
ALL audit events, 27-38	network proxy for default database, 8-30
auditing, 27-32	network proxy for Oracle Real Application
security class and ACL audit events, 27-36	Clusters, 8-30
session audit events, 27-37	operational flow, 8-20
user, privilege, and role audit events, 27-34	public client registration, 8-21
Oracle Database Vault	retrieving token using Azure CLI, 8-26
auditing, 27-41	ROPC used to get Azure AD token, 8-23
command rules, audit events, 27-45	SQL*Plus using Azure AD token, 8-26
Data Pump, audit events, 27-48	supported drivers, 8-19
enable and disable, audit events, 27-48	testing Azure endpoint accessibility, 8-28
	Oracle Database-to-Microsoft Azure AD client
factors, audit events, 27-46	
OLS, audit events, 27-47	connections
realms, audit events, 27-44	net naming for IAM, 7-22, 8-19
rule sets and rules, audit events, 27-44	network proxy for Windows, 8-31
secure application roles, audit events, 27-47	secrets for IAM, 7-22, 8-19
Oracle Database-to-Azure AD authorizations	Oracle DBaaS client connections
disabling, 8-16	supported drivers, 7-22
enabling, 8-15	Oracle DBaaS-to-Azure AD proxy authentication
Oracle Database-to-IAM	about, 8-34
trace files for client side, 8-36	configuring, 8-34
Oracle Database-to-Microsoft Azure Active	validating, 8-34
Directory	Oracle DBaaS-to-IAM
about, <i>8-2</i>	about, 7-2, 7-22
architecture, 8-3	about token requests using passwords or
assigning app role to service principal, 8-14	SEPS, 7-23
assigning users and groups to Azure AD app	architecture, 7-4
roles, 8-14	cross-tenancy access examples, 7-34
Azure AD token, checking version of, 8-37	cross-tenancy, about, 7-31
configuring v2 tokens, 8-12	database clients for cross-tenancy access,
creating Azure AD app roles, 8-13	7-36
exclusive mapping between database schema	parameters for setting password or SEPS
and Azure AD user, 8-17	token requests, 7-24
mapping Oracle roles with Azure AD roles,	requesting cross-tenancy tokens, 7-36
8-18	trace files for client side, 7-39
on-premises requirements, 8-8	troubleshooting client side, 7-39
Oracle schema-to-Azure AD application role	Oracle DBaaS-to-IAM authorizations
mapping, 8-17	about, 7-10
registering database instance to Microsoft	altering, 7-13
Azure tenancy, 8-9	creating IAM database password, 7-20
trace files for client, levels, 8-35	creating policies for authenticating users, 7-19
trace files for client, setting, 8-36	enabling, 7-8
use cases, 8-6	
	IAM user to database global user, 7-12
user and group mappings, 8-5, 8-7	IAM user to database global user, 7-12
	instance principals, 7-13

Oracle DBaaS-to-IAM authorizations (continued)	Oracle Label Security
mapping schemas and roles to users and	audit events, 27-52
groups in another tenancy, 7-35	auditing, 27-51
migrating, 7-13	auditing internal predicates in policies, 27-18
resource principals, 7-13	user session label audit events, 27-54
shared database global user, 7-11	Oracle Label Security (OLS)
source user tenancy, 7-33	Oracle Virtual Private Database, using with,
target database resource tenancy, 7-33	14-47
token requested by IAM user name and	Oracle native encryption
password, 7-27	configured with SSL authentication, 18-15
token requested by IAM user name and	Oracle Net, A-14
secure external password store	firewall support, A-14
(SEPS), 7-26	Oracle parameters
user authorization, verifying, 7-14	authentication, 25-5
Oracle DBaaS-to-IAM client connections	Oracle Password Protocol, 19-3
IAM token, 7-29	Oracle RAC
password verifier, 7-23	Transport Layer Security, 23-34
SQL*Plus using an IAM database password,	Oracle Real Application Clusters
7-28	components that need certificates, 23-37
token, 7-27	global application contexts, 13-30
Oracle DBaaS-to-IAM connections	SYSRAC administrative privilege, 4-9
about, 7-8	Oracle Real Application Security
connection pools using instance or resource	auditing internal predicates in policies, 27-18
principals, 7-20	Oracle Recovery Manager
database links, 7-36	audit events, 27-40
walletless connections, 7-28	auditing, 27-39
Oracle DBaaS-to-IAM proxy authentication	SYSBACKUP administrative privilege, 4-6
about, 7-17	Oracle Scheduler
configuring, 7-18	sensitive credential data
validating, 7-18	about, <i>16-1</i>
Oracle Developer Tools For Visual Studio (ODT)	data dictionary views, 16-6
debugging using Java Debug Wire Protocol,	deleting, 16-4
10-21	encrypting, 16-2
Oracle E-Business Suite	multitenant environment, 16-2
schema-only accounts, 3-63	rekeying, 16-3
Oracle Enterprise Manager	restoring functioning of lost keystore, 16-5
PDBs, 11-1	Oracle SQL*Loader
statistics monitor, 2-25	Direct Load Path audit events, 27-62
Oracle Enterprise Security Manager	Oracle Technology Network
role management with, 3-67	security alerts, A-2
Oracle home	Oracle Virtual Private Database, 14-2
default permissions, disallowing modification,	exporting data using Data Pump Export,
A-10	14-49
Oracle Internet Directory	Oracle Virtual Private Database (VPD), 14-3
Diffie-Hellman TLS port, 23-58	about, <i>14-2</i>
Oracle Internet Directory (OID)	ANSI operations, 14-45
authenticating with directory-based service,	application containers, 14-5
3-67	application contexts
SYSDBA and SYSOPER access, controlling,	tutorial, 14-31
3-54	used with, 14-4
Oracle Java Virtual Machine	applications
JAVA_RESTRICT initialization parameter	how it works with, 14-45
security guideline, A-10	users who are database users, how it
Oracle Java Virtual Machine (OJVM)	works with, 14-49
permissions, restricting, A-3	applications using for security, 12-3
pormissions, restricting, 71 o	automatic reparsing, how it works with, 14-46
	AUTOHIATO LODALOHIA, HOW IT MOTES MILL TA-AD

Oracle Virtual Private Database (VPD) (continued)	Oracle Virtual Private Database (VPD) (continued)
benefits, 14-3	policy types (continued)
CDBs, 14-5	context sensitive, restricting evaluation,
column level, 14-12	14-23
column masking behavior	context sensitive, when to use, 14-25
enabling, <i>14-14</i>	context-sensitive, audited, 27-19
restrictions, 14-15	DYNAMIC, 14-20
column-level display, 14-12	dynamic, audited, 27-19
components, 14-6	shared context sensitive, about, 14-25
configuring, 14-8	shared context sensitive, when to use,
cursors, shared, 14-4	14-25
edition-based redefinitions, 14-44	shared static, about, 14-22
editions, results in, 13-32	shared static, when to use, 14-22
Enterprise User Security proxy authentication,	static, about, 14-21
how it works with, 14-49	static, audited, 27-19
exporting data, 14-47	static, when to use, 14-22
extended data objects in views, 14-10	summary of features, 14-26
finding information about, 14-50	privileges required to create policies, 14-4
flashback query, how it works with, 14-46	SELECT FOR UPDATE statements in
function	policies, 14-45
components, 14-6	tutorial, simple, 14-27
how it is executed, 14-4	user models, 14-49
JDBC proxy authentication, how it works with,	Web-based applications, how it works with,
14-49	14-49
nondatabase user applications, how works	Oracle Virtual Private Datebase (VPD)
with, 14-49	predicates
OCI proxy authentication, how it works with,	audited in fine-grained audit policies,
14-49	27-95
Oracle Label Security	audited in unified audit policies, 27-18
exceptions in behavior, 14-47	Oracle Wallet Manager, 3-67
using with, 14-47	X.509 Version 3 certificates, 3-67
outer join operations, 14-45	Oracle wallets
performance benefit, 14-3	authentication method, 3-67
policies, Oracle Virtual Private Database	search order for TLS, 23-33
about, 14-9	setting location, 23-15
applications, validating, 14-18	sglnet.listener.ora setting, C-7
attaching to database object, 14-10	sqlnet.ora location setting, C-7
column display, 14-12	ORACLE_DATAPUMP access driver
column-level display, default, 14-13	guidelines for security, A-11
dynamic, 14-20	ORACLE_OCM user account, 2-35
multiple, 14-17	OracleMetaLink
optimizing performance, 14-19	See My Oracle Support
privileges used to run, 14-4	orapki utility
SQL statements, specifying, 14-11	about, <i>F-1</i>
policy groups	adding a certificate request to a wallet with,
about, <i>14-16</i>	F-8
	adding a root certificate to a wallet with, <i>F-9</i>
benefits, 14-16 creating, 14-16	adding a trusted certificate to a wallet with,
o ,	F-9
default, 14-17	adding certificate to wallet, <i>F-3</i>
tutorial, implementation, 14-38	
policy types	adding user certificates to a wallet with, <i>F-9</i>
context sensitive, about, 14-23	adding user-supplied certificate to wallet, <i>F-3</i>
context sensitive, altering existing policy,	cert create command, <i>F-14</i>
14-24	cert display command, <i>F-14</i>
context sensitive, creating, 14-24	certificate revocation lists, 23-56
context sensitive, refreshing, 14-24	changing the wallet password with, F-7

orapki utility (continued)	parallel query, and SYS_CONTEXT, 13-12
converting wallet to use AES256 algorithm,	parameters
F-7	authentication
creating a local auto-login wallet with, F-7	Kerberos, C-1
creating a wallet with, F-5	RADIUS, C-8
creating an auto-login wallet with, F-6	Transport Layer Security (TLS), C-2
creating signed certificates for testing, <i>F-2</i>	configuration for JDBC, 19-4
crl delete command, <i>F-14</i>	encryption and checksumming, 18-11
crl display command, <i>F-15</i>	pass phrase
crl hash command, <i>F-15</i>	read and parse server.key file, A-17
crl list command, <i>F-16</i>	PASSWORD command
crl upload command, <i>F-16</i>	about, 2-20
·	
examples, <i>F-11</i>	changing SYS password with, 2-21
exporting a certificate from a wallet with, <i>F-10</i>	password complexity functions
exporting a certificate request from a wallet	aboutr, 3-26
with, <i>F-10</i>	administrative users, for, 3-53
managing certificate revocation lists, <i>F-10</i>	customizing, 3-28
syntax, <i>F-2</i>	enabling, 3-29
viewing a test certificate with, F-3	how database checks password complexity,
viewing a wallet with, F-7	3-26
wallet add command, F-17	ora12c_stig_verify_function, 3-28
wallet convert command, <i>F-17</i>	ora12c_strong_verify_function, 3-27
wallet create command, <i>F-18</i>	ora12c_verify_function, 3-27
wallet display command, <i>F-18</i>	privileges required, 3-26
wallet export command, F-19	verify_function_11G, 3-26
ORAPWD utility	password files
case sensitivity in passwords, 3-36	case sensitivity, effect on
changing SYS password, 2-22	SEC_CASE_SENSITIVE_LOGON
changing SYS password with, 2-21	parameter, 3-31
ORDDATA user account, 2-32	how used to authenticate administrators, 3-59
ORDPLUGINS user account, 2-32	migration of for administrative users, 3-52
ORDSYS user account, 2-32	password limits
OS_AUTHENT_PREFIX parameter, 25-6	administrative logins, 3-59
	password management
OS_ROLES initialization parameter	
operating system role grants, 4-83	inactive user accounts, locking automatically,
operating-system authorization and, 4-39	3-9
REMOTE_OS_ROLES and, 4-83	password versions
using, 4-82	target databases that run earlier releases,
OSS.SOURCE.MY_WALLET parameter, 23-15,	3-42
23-26	using 12C exclusively, 3-40
outer join operations	PASSWORD_LIFE_TIME profile parameter, 3-12
Oracle Virtual Private Database affect on,	PASSWORD_LOCK_TIME profile parameter,
14-45	3-10
OUTLN user account, 2-32	PASSWORD_REUSE_MAX profile parameter,
	3-11
P	PASSWORD_REUSE_TIME profile parameter,
Г	3-11
packages	PASSWORD_ROLLOVER_TIME parameter, 3-19
auditing, 27-14, 27-17	passwords, 3-3
examples, 4-64	10G password version, finding and resetting,
·	3-34
examples of privilege use, 4-64	about managing, 3-5
granting roles to, 4-41	account locking, 3-10
privileges	administrator
divided by construct, 4-63	
executing, 4-62, 4-63	authenticating with, 3-59
parallel execution servers, 13-12	guidelines for securing, A-7

passwords (continued)	passwords (continued)
aging and expiration, 3-12	PASSWORD_LOCK_TIME profile parameter,
ALTER PROFILE statement, 3-5	3-10
altering, 2-20	PASSWORD_REUSE_MAX profile
application design guidelines, 12-4	parameter, 3-11
applications, strategies for protecting	PASSWORD REUSE TIME profile
passwords, 12-3	parameter, 3-11
brute force attacks, 3-3	policies, 3-4
case sensitivity, configuring, 3-30	privileges for changing for roles, 4-37
changing for roles, 4-37	privileges to alter, <i>2-18</i>
changing SYS with ORAPWD utility, 2-22	protections, built-in, 3-3
complexity verification	proxy authentication, 3-86
about, <i>3-26</i>	requirements
complexity, guidelines for enforcing, A-7	additional, A-7
compromised, how to handle, 3-23	minimum, 3-4
connecting without, 3-64	reusing, 3-11, A-7
CREATE PROFILE statement, 3-5	reusing passwords, 3-11
danger in storing as clear text, A-7	role password case sensitivity, 3-31
database user authentication, 3-61	roles authenticated by passwords, 4-34
default profile settings	roles enabled by SET ROLE statement, 4-38
about, 3-6	secure external password store, 3-44
default user account, A-7	security risks, 3-60
default, finding, 3-6	SYS account, <i>2-21</i> SYS and SYSTEM, <i>A-7</i>
delays for incorrect passwords, 3-3	•
duration, A-7	used in roles, 4-25
encrypting, 3-3, A-7	utlpwdmg.sql password script
examples of creating, 3-4	password management, 3-26
expiring	verified using SHA-512 hash function, 3-40
explicitly, 3-13	versions, management of, 3-32
procedure for, 3-12	See also authentication, and access control list (ACL), wallet access
proxy account passwords, 3-82	PDB lockdown profiles
with grace period, 3-13	about, 4-45
failed logins, resetting, 3-10	creating, 4-47
finding users who use old passwords, 3-24	default, 4-47
forcing oracle user to enter when logging in as	disabling, 4-49
SYSDBA, 4-6	dropping, 4-50
grace period, example, 3-13	enabling, 4-49
gradual database rollover, 3-17	inheritance, 4-46
guidelines for security, A-7	PDB DBA role, 4-29
history, 3-11, A-7	PDBs
Java code example to read passwords, 12-8	application common users
length, A-7	about, 2-3
life time set too low, 3-15	auditing
lifetime for, 3-12	types of audit settings allowed, 26-9
lock time, 3-10	unified audit policy syntax, 27-5
management rules, A-7	what can be audited, 26-2
managing, 3-4	
maximum reuse time, 3-11	CDB common users
ORAPWD utility, 3-36	about, 2-3
password complexity verification, 3-26	common roles
how database checks, 3-26	about, 4-19
ora12c_stig_verify_function, 3-28	creating, 4-21
ora12c_verify_function function, 3-27	granting, 4-22
privileges required, 3-26	how they work, 4-19
verify_function_11G function, 3-26	privileges required for management, 4-20
password file risks, 3-60	revoking, 4-22

PDBs (continued)	permissions
common roles (continued)	default, <i>A-10</i>
rules for creating, 4-20	run-time facilities, A-3
common users	PKCS #11 devices, 23-8
accessing data in PDBs, 4-17	PKCS #11 error
creating, 2-15	ORA-40300, 23-67
viewing privilege information, 4-16	ORA-40301, 23-67
Enterprise Manager	ORA-40302, 23-67
about, <i>11-1</i>	PKI
creating common roles, 11-8	See public key infrastructure (PKI)
creating common users, 11-4	PL/SQL
creating local roles, 11-10	roles in procedures, 4-27
creating local users, 11-6	PL/SQL packages
dropping common roles, 11-9	auditing, 27-14, 27-17
dropping common users, 11-5	PL/SQL procedures
dropping local roles, 11-11	setting application context, 13-10
dropping local users, 11-7	PL/SQL stored procedures
editing common roles, 11-8	network access for debugging operations,
editing common users, 11-4	10-21
editing local roles, 11-10	PM user account, 2-36
editing local users, 11-6	PMON background process
logging in, 11-2	application contexts, cleaning up, 13-6
revoking common privilege grants, 11-9	POODLE attacks, preventing with, 23-22
revoking local privilege grants, 11-11	positional parameters
switching to different container, 11-3	security risks, 12-6
fine-grained audit policies, 27-96	predefined schema user accounts, 2-31
local roles	principle of least privilege, A-3
about, <i>4-19</i>	about, A-3
creating, 4-21	granting user privileges, A-3
rules for creating, 4-21	middle-tier privileges, 3-84
local users	privilege analysis
about, <i>2-5</i>	about, 5-2
creating, 2-16	accessing reports in Cloud Control, 5-13
operating system user configuration, 3-69	benefits, 5-2
operating system user for, setting, 3-68	CDBs, 5-4
privilege analysis, 5-4	creating, 5-6
privileges	creating role in Cloud Control, 5-14
common, <i>4-14</i>	data dictionary views, 5-27
granting, 4-16	DBMS_PRIVILEGE_CAPTURE PL/SQL package, 5-5
how affected, 4-3	disabling, 5-10
object, <i>4-15</i>	dropping, 5-13
revoking, 4-16	enabling, 5-9
viewing information about, 4-16	examples of creating and enabling, 5-8
PUBLIC role, 4-20	general steps for managing, 5-5
sqlnet.ora settings, 3-40	generating regrant scripts, 5-17
transparent sensitive data protection, 15-4	generating reports
viewing information about, 4-16	about, 5-10
Virtual Private Database policies, 14-5	in Cloud Control, 5-12
performance	using DBMS_PRIVILEGE_CAPTURE.GENERATE_REF
application contexts, 13-2	5-12
auditing, 26-3	generating revoke scripts, 5-16
Oracle Virtual Private Database policies, 14-3	logon users, 5-3
Oracle Virtual Private Database policy types,	multiple named capture runs, 5-11
14-19	pre-compiled database objects, 5-4
resource limits and, 2-23	privilege uses captured, 5-3
	requirements for using, 5-3

privilege analysis <i>(continued)</i>	privileges (continued)
restrictions, 5-3	revoking privileges (continued)
revoking and re-granting in Cloud Control, 5-15	object privileges, cascading effect, 4-79
revoking and regranting using scripts, 5-15	object privileges, requirements for, 4-76
tutorial, 5-22	schema object, 4-53
tutorial for ANY privileges, 5-17	revoking system privileges, 4-76
use cases, 5-2	roles
finding application pool privileges, 5-2	creating, 4-34
finding overly privileged users, 5-3	dropping, 4-42
privileges, 4-10	restrictions on, 4-28
about, 4-2	roles, why better to grant, 4-3
access control lists, checking for external	schema object, 4-51
network services, 10-19	DML and DDL operations, 4-58
altering	packages, 4-63
passwords, 2-20	procedures, 4-62
users, 2-18	SELECT system privilege, 4-53
altering role authentication method, 4-37	SQL statements permitted, 12-25
applications, managing, 12-18	synonyms and underlying objects, 4-55
auditing use of, 27-9	system
auditing, recommended settings for, A-21	granting and revoking, 4-12
cascading revokes, 4-79	SELECT ANY DICTIONARY, A-10
column, 4-75	SYSTEM and OBJECT, A-3
compiling procedures, 4-63	system privileges
creating or replacing procedures, 4-62	about, <i>4-11</i>
creating users, 2-6	trigger privileges, 9-2
data links	used for Oracle Virtual Private Database
privilege management, 4-57	policy functions, 14-4
dropping profiles, 2-29	view privileges
extended data links	creating a view, 4-60
privilege management, 4-58	using a view, 4-61
granting	views, 4-60
about, 4-12, 4-70	See also access control list (ACL) and system
examples, 4-64	privileges, privilege captures
object privileges, 4-53, 4-72	procedures
system, 4-70	auditing, 27-14, 27-17
system privileges, 4-70	compiling, 4-63
grants, listing, 4-88	definer's rights
grouping with roles, 4-23	about, 9-2
managing, 12-24	roles disabled, 4-27
metadata links, 4-56	examples of, 4-64
middle tier, 3-84	examples of privilege use, 4-64
object, 4-51, 4-53, 12-25	granting roles to, 4-41
granting and revoking, 4-53	invoker's rights
on selected columns, 4-78	about, 9-3
procedures, 4-62	roles used, 4-28
creating and replacing, 4-62	privileges for procedures
executing, 4-62	create or replace, 4-62
in packages, 4-63	executing, 4-62
	executing in packages, 4-63
READ ANY TABLE system privilege	privileges required for, 4-62
about, 4-54	security enhanced by, 9-2
restrictions, 4-54	process monitor process (PMON)
READ object privilege, 4-53	cleans up timed-out sessions, 2-25
reasons to grant, 4-3	
revoking privileges	PRODUCT_USER_PROFILE table
about, <i>4-12</i>	SQL commands, disabling with, 4-43
object, <i>4-76</i>	

profile parameters	Public Key Infrastructure (PKI)
FAILED_LOGIN_ATTEMPTS, 3-6	certificate, 23-6
INACTIVE_ACCOUNT_TIME, 3-6, 3-9	certificate authority, 23-6
PASSWORD_GRACE_TIME, 3-6, 3-13	certificate revocation lists, 23-6
PASSWORD_LIFE_TIME, 3-6, 3-13, 3-15	PKCS #11 hardware devices, 23-8
PASSWORD_LOCK_TIME, 3-6, 3-10	wallets, 23-7
PASSWORD_REUSE_MAX, 3-6, 3-11	PUBLIC role
PASSWORD_REUSE_TIME, 3-6, 3-11	about, <i>4-13</i>
PASSWORD_ROLLOVER_TIME, 3-19	granting and revoking privileges, 4-80
profiles, 2-26	procedures and, 4-80
about, <i>2-26</i>	security domain of users, 4-27
application, 2-28	PUBLIC role, CDBs, 4-20
··	
assigning to user, 2-28	PUBLIC_DEFAULT profile
CDB, 2-28	profiles, dropping, 2-29
common, 2-28	
creating, 2-27	Q
dropping, 2-29	
finding information about, 2-37	quotas
finding settings for default profile, 2-39	tablespace, 2-11
managing, 2-26	temporary segments and, 2-11
ora_stig_profile user profile, 2-27	unlimited, 2-12
privileges for dropping, 2-29	viewing, 2-38
specifying for user, 2-13	
viewing, 2-39	_
program units	R
granting roles to, 4-41	DADUIG 00 4
PROVISIONER role, 4-29	RADIUS, 20-4
proxy authentication	accounting, 24-18
about, 3-79	asynchronous authentication mode, 24-5
advantages, 3-79	authentication modes, 24-3
	authentication parameters, C-8
auditing operations, 3-76	challenge-response
auditing users, 27-23	authentication, 24-5
client-to-middle tier sequence, 3-83	user interface, <i>D-1</i>
creating proxy user accounts, 3-80	configuring, 24-8
middle-tier	database links not supported, 24-1
authorizing but not authenticating users,	initialization parameter file setting, C-13
3-85	location of secret key, 24-13
authorizing to proxy and authenticate	minimum parameters to set, <i>C-13</i>
users, 3-85	smartcards and, 20-4, 24-7, 24-14, D-1
limiting privileges, 3-84	SQLNET.AUTHENTICATION_SERVICES
reauthenticating users, 3-86	parameter, C-9
passwords, expired, 3-82	sqlnet.ora file sample, <i>B-1</i>
privileges required for creating users, 3-80	SQLNET.RADIUS_ALTERNATE parameter, C-9
secure external password store, used with,	
3-82	SQLNET.RADIUS_ALTERNATE_PORT parameter,
security benefits, 3-79	C-9
users, passing real identity of, 3-83	SQLNET.RADIUS_ALTERNATE_RETRIES
proxy user accounts	parameter, C-10
privileges required for creation, 3-80	SQLNET.RADIUS_ALTERNATE_TIMEOUT
	parameter, C-10
PROXY_USERS view, 3-82	SQLNET.RADIUS_AUTHENTICATION parameter,
pseudo columns	C-10
USER, 4-61	SQLNET.RADIUS_AUTHENTICATION_INTERFACE
public and private key pair, defined, 20-5	parameter, <i>C-11</i>
public key infrastructure (PKI), 3-67, 20-5	SQLNET.RADIUS_AUTHENTICATION_PORT
about, 3-67	parameter, <i>C-11</i>

RADIUS (continued)	resource limits (continued)
SQLNET.RADIUS_AUTHENTICATION_RETRIES	sessions
parameter, C-11	concurrent for user, 2-25
SQLNET.RADIUS_CHALLENGE_KEYWORD	elapsed connection time, 2-25
parameter, C-12	idle time, 2-25
SQLNET.RADIUS_CHALLENGE_RESPONSE	SGA space, 2-25
parameter, <i>C-12</i>	types, 2-23
SQLNET.RADIUS_CLASSPATH parameter, C-12	RESOURCE privilege
SQLNET.RADIUS SECRET parameter, C-13	CREATE SCHEMA statement, needed for,
SQLNET.RADIUS_SEND_ACCOUNTING	12-23
parameter, C-13	RESOURCE role, 4-66
synchronous authentication mode, 24-3	about, <i>4-29</i>
system requirements, 20-7	restrictions, 20-7
RADIUS authentication, 3-67	REVOKE CONNECT THROUGH clause
READ ANY TABLE system privilege	revoking proxy authorization, 3-82
about, 4-54	REVOKE statement
restrictions, 4-54	system privileges and roles, 4-76
READ object privilege	when takes effect, 4-84
about, 4-53	revoking privileges and roles
guideline for using, A-3	cascading effects, 4-79
SQL92_SECURITY initialization parameter,	on selected columns, 4-78
4-54	REVOKE statement, 4-76
reads	specifying ALL, 4-53
limits on data blocks, 2-24	when using operating-system roles, 4-83
realm (Kerberos), 22-3	role identification
REDACT_AUDIT transparent sensitive data	operating system accounts, 4-82
protection default policy, 15-17	ROLE_SYS_PRIVS view
redo log files	application privileges, 12-18
auditing committed and rolled back	ROLE_TAB_PRIVS view
transactions, A-20	application privileges, finding, 12-18
REFERENCES privilege	roles, <i>12-20</i>
CASCADE CONSTRAINTS option, 4-78	about, 4-2, 4-24
revoking, 4-78	ACCHK_READ role, 4-29
remote authentication, <i>A-13</i>	ADM_PARALLEL_EXECUTE_TASK role,
remote debugging	4-29
configuring network access, 10-21	ADMIN OPTION and, 4-71
REMOTE_OS_AUTHENT initialization parameter	advantages in application use, 12-18
guideline for securing, A-13	application, 4-26, 4-42, 12-22, 12-24
setting, 3-75	application, 4-20, 4-42, 12-22, 12-24 application privileges, 12-18
REMOTE_OS_ROLES initialization parameter	applications, for user, 12-22
OS role management risk on network, 4-83	AUDIT ADMIN role, 4-29
setting, 4-39	AUDIT VIEWER role, 4-29
	AUTHENTICATEDUSER role, 4-29
REMOTE_SCHEDULER_AGENT user account,	
2-32	authorization, 4-37
resource limits	authorized by enterprise directory service, 4-40
about, 2-23	
call level, limiting, 2-24	CAPTURE_ADMIN role, 4-29
connection time for each session, 2-25	CDB_DBA role, 4-29
CPU time, limiting, 2-24	changing authorization for, 4-37
determining values for, 2-25	changing passwords, 4-37
idle time in each session, 2-25	common, auditing, 27-7
logical reads, limiting, 2-24	common, granting, 4-22
private SGA space for each session, 2-25	CONNECT role
profiles, 2-26	about, 4-29
session level, limiting, 2-24	create your own, A-7
	CSW_USR_ROLE role, 4-29

roles (continued)	roles (continued)
CTXAPP role, 4-29	JAVASYSPRIV role, 4-29
CWM_USER role, 4-29	JAVAUSERPRIV role, 4-29
database role, users, 12-22	JMXSERVER role, 4-29
DATAPUMP_EXP_FULL_DATABASE role,	job responsibility privileges only, A-7
4-29	LBAC_DBA role, 4-29
DATAPUMP_IMP_FULL_DATABASE role,	listing grants, 4-88
4-29	listing privileges and roles in, 4-91
DBA role, 4-29	listing roles, 4-90
DBFS_ROLE role, 4-29	LOGSTDBY_ADMINISTRATOR role, 4-29
DDL statements and, 4-28	management using the operating system,
default, 4-85	4-81
default, setting for user, 2-17	managing roles
definer's rights procedures disable, 4-27	about, 4-23
dependency management in, 4-28	categorizing users, 12-24
disabling, 4-84	managing through operating system, <i>4-29</i>
dropping, 4-42	managing with RADIUS server, 24-20
EJBCLIENT role, 4-29	maximum number a user can enable, 4-85
EM_EXPRESS_ALL role, 4-29	multibyte characters in names, 4-34
EM_EXPRESS_BASIC role, 4-29	multibyte characters in passwords, 4-38
enabled or disabled, 4-24, 4-40	naming, 4-24
enabling, 4-84, 12-22	network authorization, 4-39
enterprise, 3-70, 4-40	
·	network client authorization, 4-39
EXP_FULL_DATABASE role, 4-29 external, 4-36	OEM_ADVISOR role, 4-29 OEM_MONITOR role, 4-29
functionality, 4-3, 4-24	OLAP_USER role, 4-29
functionality of, 4-24	OLAP_VS_ADMIN_role_4.20
GATHER_SYSTEM_STATISTICS role, 4-29 global authorization, 4-40	OLAP_XS_ADMIN role, 4-29 One Big Application User, compromised by,
about, <i>4-40</i>	12-2
global roles	operating system, 4-82
-	
about, 3-70	operating system authorization, 4-39
creating, 4-40	operating system granting of, 4-83
example, 4-36 external sources, and, 4-39	operating system identification of, 4-82
	operating system management and the
GLOBAL_AQ_USER_ROLE role, 4-29	shared server, 4-83
GRANT statement, 4-83 granted to other roles, 4-24	operating system-managed, 4-83 operating-system authorization, 4-39
granting and revoking to program units, 9-16	OPTIMIZER_PROCESSING_RATE role, 4-29
granting roles	password case sensitivity, 3-31
about, 4-70	PDB_DBA role, 4-29
methods for, 4-40	predefined, 4-29
system, 4-70	privilege analysis, 5-3
system privileges, 4-12	privileges for creating, 4-34
granting to program units, 4-41	privileges for dropping, 4-42
guidelines for security, A-7 HS ADMIN EXECUTE ROLE role, 4-29	privileges, changing authorization method for,
	4-37
HS_ADMIN_ROLE role, 4-29	privileges, changing passwords, 4-37
HS_ADMIN_SELECT_ROLE role, 4-29	PROVISIONER role, 4-29
IMP_FULL_DATABASE role, 4-29	RESOURCE role, 4-29
in applications, 4-25 indirectly granted, 4-24	restricting from tool users, 4-42 restrictions on privileges of, 4-28
invoker's rights procedures use, 4-28	REVOKE statement, 4-83
JAVA ADMIN role, 4-29	
-	revoking, 4-40, 4-76
JAVADEBUGPRIV role, 4-29	SCHEDULER_ADMIN role, 4-29
JAVAIDPRIV role, 4-29	schemas do not contain, 4-24

roles (continued)	schema objects (continued)
security domains of, 4-27	privileges on, 4-51
SET ROLE statement	privileges to access, 4-53
about, 4-38	privileges with, 4-53
example, 4-38	revoking privileges, 4-76
OS ROLES parameter, 4-83	schema user accounts, predefined, 2-31
setting in PL/SQL blocks, 4-28	schema-independent users, 12-24
SODA_APP role, 4-29	schema-only accounts, 3-63
unique names for, 4-34	schemas
use of passwords with, 4-25	auditing, recommended settings for, A-21
user, 4-27, 12-24	private, 3-71
users capable of granting, 4-41	shared among enterprise users, 3-71
uses of, 4-24, 4-26	shared, protecting objects in, 12-24
WFS_USR_ROLE role, 4-29	unique, <u>12-23</u>
WITH GRANT OPTION and, 4-73	unique, protecting objects in, 12-23
without authorization, 4-34	SCOTT user
WM_ADMIN_ROLE role, 4-29	about, 2-36
XDB SET INVOKER roles, 4-29	SCOTT user account
XDB_WEBSERVICES role, 4-29	restricting privileges of, A-7
XDB_WEBSERVICES_OVER_HTTP role,	SEC_CASE_SENSITIVE_LOGON initialization
4-29	parameter
XDB_WEBSERVICES_WITH_PUBLIC role,	deprecated, 3-30
4-29	SEC_CASE_SENSITIVE_LOGON parameter
XDBADMIN role, 4-29	conflict with
XS_CACHE_ADMIN role, 4-29	SQLNET.ALLOWED_LOGON_VERSION_SERVER
XS_NSATTR_ADMIN role, 4-29	setting, 3-30
XS RESOURCE role, 4-29	secure role passwords, 3-31
See also secure application roles	SEC_MAX_FAILED_LOGIN_ATTEMPTS
root container	initialization parameter, 12-28
viewing information about, 4-16	SEC_PROTOCOL_ERROR_FURTHER_ACTION
root file paths	initialization parameter, 12-27
for files and packages outside the database,	·
A-3	sec_relevant_cols_opt parameter, 14-15
row-level security	SEC_RETURN_SERVER_RELEASE_BANNER
See fine-grained access control, Oracle Virtual	initialization parameter, 12-28
Private Database (VPD)	SEC_USER_AUDIT_ACTION_BANNER
RSA private key, A-17	initialization parameter, 12-29
run-time facilities, A-3	SEC_USER_UNAUTHORIZED_ACCESS_BANN
restriction permissions, A-3	ER initialization parameter, 12-29
restriction permissions, 71 o	secconf.sql script
•	password settings, 3-8
S	secret key
Carbanas Oylay Ast	location in RADIUS, 24-13
Sarbanes-Oxley Act	secure application roles, 12-20
auditing to meet compliance, 26-2	about, 4-44
SCHEDULER_ADMIN role	creating, 12-19
about, 4-29	creating PL/SQL package, 12-20
schema object privileges, 4-51	finding with DBA_ROLES view, 4-86
schema objects	invoker's rights, 12-20
cascading effects on revoking, 4-79	invoker's rights requirement, 12-20
default tablespace for, 2-9	package for, 12-20
dropped users, owned by, 2-29	user environment information from
granting privileges, 4-72	SYS_CONTEXT SQL function, 12-20
privileges	using to ensure database connection, 4-44
DML and DDL operations, 4-58	secure external password store
granting and revoking, 4-53	about, 3-44
view privileges, 4-60	client configuration, 3-46

secure external password store (continued)	security attacks (continued)
examples, 3-45	shoulder surfing, 12-6
how it works, 3-45	SQL injection attacks, 12-4
proxy authentication, used with, 3-82	unlimited authenticated requests, preventing,
Secure Sockets Layer on Oracle RAC	12-28
remote client, testing configuration, 23-41	user session output, hiding from intruders,
SecurID, <u>24-4</u>	13-16
token cards, 24-4	See also security risks
security, A-3	security domains
application enforcement of, 4-25	enabled roles and, 4-24
default user accounts	security patches
	about, A-2
locked and expired automatically, A-3	downloading, A-2
locking and expiring, A-3	security policies
domains, enabled roles and, 4-40	See Oracle Virtual Private Database, policies
enforcement in application, 12-3	security risks, 3-82, A-3
enforcement in database, 12-3	ad hoc tools, 4-42
multibyte characters in role names, 4-34	
multibyte characters in role passwords, 4-38	application users not being database users,
passwords, 3-61	12-2
policies	applications enforcing rather than database,
applications, 12-1	12-3
SQL*Plus users, restricting, 4-42	bad packets to server, 12-26
tables or views, 14-3	database version displaying, 12-28
procedures enhance, 9-2	encryption keys, users managing, 17-8
products, additional, 1-2	invoker's rights procedures, 9-5
roles, advantages in application use, 12-18	password files, 3-60
See also security risks	passwords exposed in large deployments,
security alerts, A-2	3-44
security attacks, 3-3, 3-82, 17-2, A-14	passwords, exposing in programs or scripts,
access to server after protocol errors,	12-6
preventing, 12-27	positional parameters in SQL scripts, 12-6
application context values, attempts to	privileges carelessly granted, 4-13
change, 13-8	remote user impersonating another user, 4-39
application design to prevent attacks, 12-3	sensitive data in audit trail, A-19
command line recall attacks, 12-4, 12-6	server falsifying identities, <i>A-17</i>
denial of service, <i>A-14</i>	users with multiple roles, 12-22
denial-of-service	See also security attacks
bad packets, addressing, 12-26	security settings scripts
•	password settings
denial-of-service attacks through listener,	secconf.sql, 3-8
A-14	Security Sockets Layer (SSL)
disk flooding, preventing, 12-26	See Transport Layer Security (TLS)
eavesdropping, A-13	Security Technical Implementation Guide (STIG)
encryption, problems not solved by, 17-2	ora stig profile user profile, 2-27
falsified IP addresses, A-13	
falsified or stolen client system identities, A-13	ora12c_stig_verify_function password
hacked operating systems or applications,	complexity function, 3-28
A-13	SELECT ANY DICTIONARY privilege
intruders, 17-2	data dictionary, accessing, A-10
password cracking, 3-3	exclusion from GRANT ALL PRIVILEGES
password protections against, 3-3	privilege, <i>A-10</i>
preventing malicious attacks from clients,	SELECT FOR UPDATE statement in Virtual
12-26	Private Database policies, 14-45
preventing password theft with proxy	SELECT object privilege
authentication and secure external	guideline for using, A-3
password store, 3-82	privileges enabled, 4-53
session ID, need for encryption, 13-42	
,	

SELECT_CATALOG_ROLE role	SQL statements
SYS schema objects, enabling access to,	dynamic, <i>13-12</i>
4-11	object privileges permitting in applications,
sensitive data, auditing of, A-21	12-25
separation of duty concepts, 23	privileges required for, 4-51, 12-25
sequences	resource limits and, 2-24
auditing, 27-14	restricting ad hoc use, 4-42
server.key file	SQL statements, top-level in unified audit policies,
•	· · · · · · · · · · · · · · · · · · ·
pass phrase to read and parse, A-17	27-64
service names	SQL*Net
controlling with	See Oracle Net Services
SSL_ALLOW_WEAK_DN_MATCH	SQL*Plus
and SSL_SERVER_DN_MATCH,	connecting with, 3-64
23-25	restricting ad hoc use, 4-42
session key	statistics monitor, 2-25
defined, 23-4	SQL92_SECURITY initialization parameter
session layer	READ object privilege impact, 4-54
defined, 23-9	SQLNET.ALLOWED_LOGON_VERSION
SESSION_ROLES data dictionary view	See
PUBLIC role, 4-13	SQLNET.ALLOWED_LOGON_VERSION_CLIENT,
SESSION_ROLES view	SQLNET.ALLOWED_LOGON_VERSION_SERVER,
queried from PL/SQL block, 4-27	SQLNET.ALLOWED_LOGON_VERSION_CLIENT
sessions	target databases from earlier releases, 3-42
listing privilege domain of, 4-89	SQLNET.ALLOWED_LOGON_VERSION_SERVER
memory use, viewing, 2-40	target databases from earlier releases, 3-42
time limits on, 2-25	using only 12C password version, 3-40
when auditing options take effect, 28-2	SQLNET.ALLOWED_LOGON_VERSION_SERVER
SET ROLE statement	parameter
application code, including in, 12-23	conflict with SEC_CASE_SENSITIVE_LOGON
	FALSE setting, 3-30
associating privileges with role, 12-22	effect on role passwords, 3-31
disabling roles with, 4-84	SQLNET.AUTHENTICATION_KERBEROS5_SER
enabling roles with, 4-84	VICE parameter, 22-5
when using operating-system roles, 4-83	SQLNET.AUTHENTICATION SERVICES
SGA	parameter, 22-5, 23-21, 23-31, 24-9,
See System Global Area (SGA)	25-2, 25-4, A-17, C-2, C-9
SH user account, 2-36	SQLNET.CRYPTO_CHECKSUM_CLIENT
SHA-512 cryptographic hash function	parameter, 18-13, B-6
enabling exclusive mode, 3-40	SQLNET.CRYPTO_CHECKSUM_SERVER
Shared Global Area (SGA)	parameter, 18-13, B-5
See System Global Area (SGA)	SQLNET.CRYPTO CHECKSUM TYPES CLIEN
shared server	
limiting private SQL areas, 2-25	T parameter, 18-13, B-8
operating system role management	SQLNET.CRYPTO_CHECKSUM_TYPES_SERV
restrictions, 4-83	ER parameter, 18-13, B-7
shoulder surfing, 12-6	SQLNET.ENCRYPTION_CLIENT
SI_INFORMTN_SCHEMA user account, 2-32	with ANO encryption and TLS authentication,
smart cards	18-15
guidelines for security, A-7	SQLNET.ENCRYPTION_CLIENT parameter,
smartcards, 20-4	18-11, 25-2, B-5
and RADIUS, 20-4, 24-7, 24-14, D-1	SQLNET.ENCRYPTION_SERVER
SODA_APP role, 4-29	with ANO encryption and TLS authentication,
SQL Developer	18-15
debugging using Java Debug Wire Protocol,	SQLNET.ENCRYPTION_SERVER parameter,
10-21	18-11, 25-2, B-4
SQL injection attacks, 12-4	SQLNET.ENCRYPTION_TYPES_CLIENT
OQL IIIJOOHOII AHAONO, 12 7	parameter, 18-11, B-7

SQLNET.ENCRYPTION_TYPES_SERVER	sqlnet.ora file (continued)
parameter, <i>18-1</i> 1, <i>B-6</i>	SSL_CLIENT_AUTHETNICATION parameter, 23-26
SQLNET.IGNORE_ANO_ENCRYPTION_FOR_TCPS	SSL_VERSION parameter, 23-19, 23-30
setting, 18-15	Trace File Set Up sample, <i>B-1</i>
with ANO encryption and TLS authentication,	sqlnet.ora parameters
18-15	ADD_SSLV3_TO_DEFAULT, 23-22
SQLNET.KERBEROS5_CC_NAME parameter,	SQLNET.RADIUS_ALTERNATE parameter,
22-8	24-15, C-9
SQLNET.KERBEROS5_CLOCKSKEW	SQLNET.RADIUS_ALTERNATE_PORT
parameter, 22-8	parameter, <i>24-15</i> , <i>C-9</i>
SQLNET.KERBEROS5_CONF parameter, 22-8	SQLNET.RADIUS_ALTERNATE_RETRIES
SQLNET.KERBEROS5_REALMS parameter,	parameter, <i>24-15</i> , <i>C-10</i>
22-8	SQLNET.RADIUS_ALTERNATE_TIMEOUT
sqlnet.ora file	parameter, <i>24-15</i> , <i>C-10</i>
Common sample, <i>B-1</i>	SQLNET.RADIUS_AUTHENTICATION
FIPS 140-2	parameter, <i>C-10</i>
Cipher Suite settings, <i>E-4</i>	SQLNET.RADIUS_AUTHENTICATION_INTERFA
Kerberos sample, <i>B-1</i>	CE parameter, <i>C-11</i>
Oracle Advanced Security checksum sample, B-1	SQLNET.RADIUS_AUTHENTICATION_PORT
Oracle Advanced Security encryption sample, B-1	parameter, <i>C-11</i>
Oracle wallet setting, C-7	SQLNET.RADIUS_AUTHENTICATION_RETRIES
OSS.SOURCE.MY_WALLET parameter, 23-15, 23-26	parameter, <i>C-11</i>
parameters for clients and servers using Kerberos, C-1	SQLNET.RADIUS_AUTHENTICATION_TIMEOUT
parameters for clients and servers using RADIUS, C-8	parameter, <i>C-11</i>
parameters for clients and servers using TLS, C-2	SQLNET.RADIUS_CHALLENGE_KEYWORDpar
PDBs, 3-40	ameter, <i>C-12</i>
RADIUS sample, B-1	SQLNET.RADIUS_CHALLENGE_RESPONSE
sample, B-1	parameter, <i>C-12</i>
SQLNET.AUTHENTICATION_KERBEROS5_SERVICE	SQLNET.RADIUS_CLASSPATH parameter, <i>C-12</i>
parameter, 22-5	SQLNET.RADIUS_SECRET parameter, C-13
SQLNET.AUTHENTICATION_SERVICES parameter,	SQLNET.RADIUS_SEND_ACCOUNTING
22-5, 23-21, 23-31, 25-2, 25-4, A-17	parameter, 24-18, C-13
SQLNET.CRYPTO_CHECKSUM_CLIENT parameter,	SQLNET.SSL_EXTENDED_KEY_USAGE
18-13	parameter, 23-31
SQLNET.CRYPTO CHECKSUM SERVER parameter,	SSL_ALLOW_WEAK_DN_MATCH parameter
18-13	using, 23-25
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	SSL_CIPHER_SUITES parameter, C-3
parameter, 18-13, B-8	SSL_CLIENT_AUTHENTICATION parameter,
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	23-20, 23-26, C-5
parameter, 18-13, B-7	SSL_SERVER_CERT_DN parameter, C-7
SQLNET.ENCRYPTION_CLIEN parameter, 25-2	SSL_SERVER_DN_MATCH parameter, C-6
SQLNET.ENCRYPTION_CLIENT parameter, B-5	affected by
SQLNET.ENCRYPTION_SERVER parameter, 18-11,	SSL_ALLOW_WEAK_DN_MATCH,
25-2, B-4	23-25
SQLNET.ENCRYPTION_TYPES_CLIENT parameter,	SSL_VERSION parameter, 23-19, 23-30, C-4
18-11	standard audit trail
SQLNET.ENCRYPTION_TYPES_SERVER parameter,	records, purging, 28-11
18-11	standard auditing
SQLNET.KERBEROS5_CC_NAME parameter, 22-8	affected by editions, 27-20
SQLNET.KERBEROS5_CLOCKSKEW parameter,	archiving audit trail, 28-12
22-8	privilege auditing
SQLNET.KERBEROS5_CONF parameter, 22-8	about, 27-9
SQLNET.KERBEROS5_REALMS parameter, 22-8	multitier environment, 27-23
SQLNET.KERBEROSS_REALMS parameter, 22-8 SQLNET.SSL_EXTENDED_KEY_USAGE, 23-31	records
SSL sample, <i>B-1</i>	
•	archiving, 28-12
SSL_CLIENT_AUTHENTICATION parameter, 23-20	

standard auditing (continued)	SYS.AUD\$ table
statement auditing	archiving, 28-12
multitier environment, 27-23	SYS.FGA_LOG\$ table
statement_types parameter of	archiving, 28-12
DBMS_RLS.ADD_POLICY procedure,	SYS.LINK\$ system table, 16-1
14-11	SYS.SCHEDULER\$_CREDENTIAL system table,
storage	16-1
quotas and, 2-11	SYS\$UMF user account, 2-32
unlimited quotas, 2-12	SYSASM privilege
stored procedures	password file, 3-59
using privileges granted to PUBLIC role, 4-80	SYSBACKUP privilege
strong authentication	operations supported, 4-6
centrally controlling SYSDBA and SYSOPER	password file, 3-59
access to multiple databases, 3-54	SYSBACKUP user account
disabling, 25-2	about, 2-32
guideline, A-7	SYSDBA administrative privilege
symbolic links	forcing oracle user to enter password, 4-6
restricting, A-10	SYSDBA privilege, 4-5
synchronous authentication mode, RADIUS, 24-3	directory authentication, 3-54
synonyms	Kerberos authentication, 3-55
object privileges, 4-55	password file, 3-59
privileges, guidelines on, A-3	TLS authentication, 3-56
SYS account	SYSDG privilege
auditing, <i>27-80</i>	operations supported, 4-7
changing password, 2-21	password file, 3-59
policy enforcement, 14-47	SYSDG user account
privilege analysis, 5-3	about, 2-32
SYS and SYSTEM	SYSKM privilege
passwords, A-7	operations supported, 4-8
SYS and SYSTEM accounts	password file, 3-59
auditing, 27-80	SYSKM user account
SYS objects	about, 2-32
auditing, <i>27-15</i>	SYSLOG
SYS schema	audit trail records, 28-5
objects, access to, 4-11	capturing audit trail records, 28-6
SYS user	SYSMAN user account, A-7
auditing example, 27-10	SYSOPER privilege, 4-5
SYS user account	directory authentication, 3-54
about, 2-32	password file, 3-59
SYS_CONTEXT function	SYSRAC privilege
about, 13-10	operations supported, 4-9
auditing nondatabase users with, 27-85	System Global Area (SGA), 13-2
Boolean expressions used in privilege	application contexts, storing in, 13-2
analysis, 5-6	global application context information location,
database links, 13-13	13-28
dynamic SQL statements, 13-12	limiting private SQL areas, 2-25
example, 13-14	system privileges, A-3 about, 4-11
parallel query, 13-12	
syntax, 13-11	ADMIN OPTION, 4-12
unified audit policies, 27-26	ANY
used in views, 9-8	guidelines for security, A-10
validating users, 12-20 SVS_DEFAULT Oracle Virtual Private Database	CDBs, 4-14
SYS_DEFAULT Oracle Virtual Private Database	GRANT ANY PRIVILEGE, 4-12
policy group, 14-17	granting, 4-70
SYS_SESSION_ROLES namespace, 13-10	granting and revoking, 4-12
	power of, <i>4-11</i>

system privileges (continued)	transparent sensitive data protection (TSDP) (continued)
restriction needs, 4-11 revoking, cascading effect of, 4-79	altering policies, 15-14
SELECT ANY DICTIONARY, A-10	benefits, <i>15-2</i> bind variables
with common privilege grants, 4-14	about, 15-17
system requirements	expressions of conditions, 15-18
Kerberos, 20-7	creating policies, 15-5
RADIUS, 20-7	disabling policies, 15-14
strong authentication, <i>20-7</i>	disabling REDACT AUDIT policy, 15-20
TLS, 20-7	dropping policies, 15-15
SYSTEM user account	enabling REDACT AUDIT policy, 15-21
about, 2-32	finding information about, 15-34
about, 2-32	fine-grained auditing
_	general steps, 15-31
T	general steps, 15-2
table energytion	PDBs, 15-4
table encryption	privileges required, 15-4
transparent sensitive data protection policy	REDACT_AUDIT policy, 15-17
settings, 15-34	sensitive columns in INSERT or UPDATE
tables	operations, 15-20
auditing, 27-14	sensitive columns in same SELECT query,
privileges on, 4-58	15-19
tablespaces	sensitive columns in views, 15-20
assigning defaults for users, 2-9	TDE column encryption
default quota, 2-11	general steps, 15-33
quotas for users, 2-11	settings used, 15-34
quotas, viewing, 2-38	unified auditing:settings used, 15-30
temporary	use cases, 15-3
assigning to users, 2-12	Virtual Private Database
unlimited quotas, <i>2-12</i> TCPS protocol	DBMS_RLS.ADD_POLICY parameters,
tnsnames.ora file, used in, <i>A-17</i>	15-23
Transport Layer Security, used with, A-14	general steps, 15-22
TELNET service, A-14	tutorial, 15-24
TFTP service, A-14	transparent sensitive data protection (TSDP);
thin JDBC support, 19-1	fine-grained auditing
TLS	settings used, 15-31
See Transport Layer Security (TLS)	transparent tablespace encryption
token cards, 20-4, A-7	about, <i>17-8</i>
trace file	transport layer
set up sample for sqlnet.ora file, B-1	defined, 23-9
trace files	Transport Layer Security
access to, importance of restricting, A-10	compared with native network encryption,
bad packets, 12-26	18-3
location of, finding, 13-55	Transport Layer Security (SSL)
Oracle DBaaS-to-IAM client side tracing, 7-39	sqlnet.ora file sample, <i>B-1</i>
Transparent Data Encryption	Transport Layer Security (TLS), 3-66, 20-5, 23-2
about, 17-8	about, 3-66
enabling for FIPS 140-2, <i>E-2</i>	ANO encryption and, 18-15
SYSKM administrative privilege, 4-8	application containers, 23-2
Transparent Data Encryption (TDE), 16-1	architecture, 23-9
TSDP with TDE column encryption, 15-33	AUTHENTICATION parameter, C-2
transparent sensitive data protection (TSDP	authentication parameters, C-2
unified auditing	authentication process in an Oracle
general steps, 15-29	environment, 23-4
transparent sensitive data protection (TSDP)	certificate key algorithm, A-17
about 15-2	cipher suites. A-17. C-4

Transport Layer Security (TLS) (continued) client and server parameters, C-2	Transport Layer Security on Oracle RAC <i>(continued)</i> restarting instances, 23-40
client authentication parameter, C-5	restarting listeners, 23-40
client configuration, 23-22	sqlnet.ora, 23-40
combining with other authentication methods,	TCPS protocol endpoints, 23-34
23-8	wallet and certificate creation, 23-37
compared to SSL, 23-2	wallet creation in nodes, 23-39
·	
configuration files, securing, <i>A-17</i>	Transport Layer Security, MCS
configuration troubleshooeting, 23-48	about, 23-43
configuring, 23-13	configuing database parameters, 23-47
configuring ANO encryption with, 18-15	configuring client sqlnet.ora file, 23-47
connection without client wallet, about, 23-11	configuring server listener.ora file, 23-45
connection without client wallet, configuring,	configuring server sqlnet.ora file, 23-46
23-12	creating and configuring client wallet, 23-44
enabling, 23-13	creating and configuring server wallet, 23-43
filtering certificates, 23-31	creating user account for, 23-45
FIPS library location setting (SSLFIPS_LIB),	importing client wallet into Microsoft
E-3	Certificate Store, 23-46
FIPS mode setting (SSLFIPS_140), <i>E-3</i>	testing configuration, 23-47
global users with private schemas, 3-71	Transport Layer Security(TLS)
guidelines for security, A-17	configuring for SYSDBA or SYSOPER
handshake, 23-4	access, 3-56
industry standard protocol, 23-2	triggers
listener, administering, A-14	auditing, 27-14, 27-17
MD5 certification, <i>F-3</i>	CREATE TRIGGER ON, 12-25
mode, <i>A-17</i>	logon
multiple certificates, filtering, 23-31	examples, <i>13-15</i>
parameters, ways of configuring, C-2	externally initialized application contexts,
pass phrase, A-17	13-15
requiring client authentication, 23-20	privileges for executing, 9-2
RSA private key, A-17	roles, 4-27
securing TLS connection, A-17	WHEN OTHERS exception, 13-16
server configuration, 23-14	troubleshooting, 22-22, 22-24
server.key file, A-17	finding errors by checking trace files, 13-55
SHA–1 certification, <i>F-3</i>	Kerberos common configuration problems,
SQLNET.AUTHENTICATION_SERVICES	22-23
parameter, C-2	ORA-01017 connection errors in CMU
SSL_CIPHER_SUITES parameter, C-3	configuration, 6-36, 6-40
SSL_CLIENT_AUTHENTICATION parameter,	ORA-01017 errors in Kerberos configuration,
23-20, 23-26, C-5	22-24
SSL_SERVER_CERT_DN, C-7	ORA-12631 errors in Kerberos configuration,
SSL SERVER DN MATCH, C-6	22-23
SSL_VERSION parameter, C-4	ORA-28030 connection errors in CMU
system requirements, 20-7	configuration, 6-37, 6-41
TCPS, <i>A-17</i>	ORA-28274 connection errors in CMU
Transport Layer Security (TLS), 3-66, 20-5,	configuration, 6-36, 6-40
23-2	ORA-28276 connection errors in CMU
SSL_CLIENT_AUTHENTICATION, C-5	configuration, 6-35, 6-39
version parameter, <i>C-4</i>	trace files for in CMU connection errors, 6-37,
wallet location, parameter, <i>C-7</i>	6-41
wallet search order, 23-33	trusted procedure
ways to configure parameters for, <i>C-2</i>	database session-based application contexts,
Transport Layer Security on Oracle RAC	13-2
cluster node, testing configuration, 23-41	tsnames.ora configuration file, A-17
listener.ora, 23-40	
local_listener startup parameter, 23-36	

tutorials, 13-17, 14-26	unified audit policies (continued)
application context, database session-based,	users, success or failure, 27-80
13-17	unified audit policies, administrative users
auditing	configuring, 27-12
creating policy to audit nondatabase	example, 27-12
users, 27-84	users that can be audited, 27-12
creating policy using email alert, 27-105	unified audit policies, altering
definer's rights, database links, 9-26	about, 27-77
external network services, using email alert,	configuring, 27-77
27-105	examples, 27-78
global application context with client session	unified audit policies, application containers
ID, 13-44	example, 27-75
invoker's rights procedure using CBAC, 9-17	unified audit policies, CDBs
nondatabase users	about, 27-71
creating Oracle Virtual Private Database	appearance in audit trail, 27-76
policy group, 14-38	configuring, 27-73
global application context, 13-44	examples, 27-74, 27-75
Oracle Virtual Private Database	unified audit policies, conditions
policy groups, 14-38	about, 27-26
policy implementing, 14-31	configuring, 27-26
. , ,	
simple example, 14-27 privilege analysis, 5-22	examples, 27-28
	unified audit policies, disabling
privilege analysis for ANY privileges, 5-17	about, 27-80, 27-82
TSDP with VPD, 15-24	configuring, 27-82
See also examples	unified audit policies, enabling
types	about, 27-80
creating, 4-67	configuring, 27-81
privileges on, 4-65	for groups of users through roles, 27-80
user defined	unified audit policies, object actions
creation requirements, 4-66	about, <i>27-13</i>
	actions that can be audited, 27-14
U	appearance in audit trail, 27-17
	configuring, 27-14
UDP and TCP ports	dictionary tables
close for ALL disabled services, A-14	auditing, 27-15
UGA	examples, 27-15
See User Global Area (UGA)	GRANT operations, 27-15
unified audit policies, 26-2, 27-1	SYS objects, 27-15
about, 27-4	unified audit policies, objects actions
best practices for creating, 27-5	REVOKE operations, 27-15
dropping	unified audit policies, Oracle Data Miner
about, 27-83	about, 27-57
procedure, 27-84	unified audit policies, Oracle Data Mining
location of, 27-5	configuring, 27-57
predefined	how events appear in audit trail, 27-58
ORA_ACCOUNT_MGMT, 27-90	unified audit policies, Oracle Data Pump
ORA_CIS_RECOMMENDATIONS, 27-90	about, 27-59
ORA_DATABASE_PARAMETER, 27-89	appearance in audit trail, 27-61, 27-63
ORA DV AUDPOL, <u>27-92</u>	configuring, 27-60
ORA DV AUDPOL2, 27-92	examples, 27-60
ORA LOGON FAILURES, 27-88	how events appear in audit trail, 27-61
ORA_SECURECONFIG, 27-88	unified audit policies, Oracle Database Rea
syntax for creating, 27-5	Application Security
top-level statements, 27-64	about, 27-33
users, applying to, 27-80	configuring, 27-38
users, excluding, 27-80	events to audit, 27-34

unified audit policies, Oracle Database Real Application	n Secu rity if(eo iratuidireta)ail (continued)
examples, 27-39	writing audit trail records to AUDSYS
how events appear in audit trail, 27-39	about, 28-4
predefined	immediate-write mode, 28-4
about, <i>27-91</i>	minimum flush threshold for queues, 28-2
ORA_RAS_POLICY_MGMT, 27-91	queued-write mode, 28-4
ORA_RAS_SESSION_MGMT, 27-91	unified audit trail, object actions
unified audit policies, Oracle Database Vault	READ object actions, 27-20
about, 27-42	SELECT object actions, 27-21
appearance in audit trail, 27-50	unified audit trail, Oracle Data Mining
attributes to audit, 27-43	examples, 27-58
configuring, 27-49	unified audit trail, top-level statements, 27-64
data dictionary views, 27-43	unified audit trial
example of auditing factors, 27-50	Oracle Data Mining audit events, 27-57
example of auditing realm, 27-49	Oracle Data Pump audit events, 27-60
example of auditing rule set, 27-50	Oracle Database Real Application Security
example of auditing two events, 27-50	ALL audit events, 27-38
how events appear in audit trail, 27-50	Oracle Database Real Application Security
unified audit policies, Oracle Label Security	security class and ACL audit events,
about, 27-51	27-36
appearance in audit trail, 27-56	Oracle Database Real Application Security
configuring, 27-54	session audit events, 27-37
examples, 27-55	Oracle Database Real Application Security
how events appear in audit trail, 27-56	user, privilege, and role audit events,
LBACSYS.ORA_GET_AUDITED_LABEL	27-34
function, 27-56	Oracle Database Vault command rule events,
unified audit policies, Oracle Recovery Manager	27-45
about, 27-40	Oracle Database Vault Data Pump events,
how events appear in audit trail, 27-41	27-48
unified audit policies, Oracle SQL*Loader	Oracle Database Vault enable and disable
about, 27-61	events, 27-48
configuring, 27-62	Oracle Database Vault factor events, 27-46
example, 27-63	Oracle Database Vault OLS events, 27-47
how events appear in audit trail, 27-63	Oracle Database Vault realm events, 27-44
unified audit policies, privileges	Oracle Database Vault rule set and rule
about, 27-9	events, 27-44
appearance in audit trail, 27-11	Oracle Database Vault secure application role
configuring, 27-10	events, 27-47
examples, <i>27-10</i>	Oracle Label Security audit events, 27-52
privileges that can be audited, 27-9	Oracle Label Security user session label
privileges that cannot be audited, 27-10	events, 27-54
unified audit policies, roles	Oracle Recovery Manager audit events, 27-40
about, 27-7	Oracle SQL*Loader Direct Load Path audit
configuring, 27-8	events, 27-62
examples, 27-8	unified auditing
unified audit policies, top-level statements, 27-64	benefits, 26-5
appearance in audit trail, 27-70	compared with mixed mode auditing, 26-6
how events appear in audit trail, 27-70	database creation, 26-8
unified audit session ID, finding, 27-29	disabling, 28-10
unified audit trail	enablement of, 26-7
about, 26-4	finding if migrated to, 26-5
archiving, 28-12	mixed mode auditing
loading audit records to, 28-8	about, 26-6
Oracle Data Pump, 28-9	capabilities, 26-8
when records are created, 28-2	purging records
	example, 28-24

unified auditing (continued)	user accounts, predefined (continued)
purging records (continued)	LBACSYS, 2-32
general steps for manual purges, 28-14	MDDATA, 2-35
general steps for scheduledl purges,	MDSYS, 2-32
28-13	OE, 2-36
transparent sensitive data protection policy	OJVMSYS, 2-32
settings, 15-30	OLAPSYS, 2-32
tutorial, <u>27-84</u>	ORACLE_OCM, 2-35
unified audting	ORDDATA, 2-32
TSDP policies and, 15-29	ORDPLUGINS, 2-32
UNIFIED_AUDIT_COMMON_SYSTEMLOG	ORDSYS, 2-32
initialization parameter	OUTLN, 2-32
using, 28-6	PM, 2-36
UNIFIED_AUDIT_SYSTEMLOG initialization	REMOTE_SCHEDULER_AGENT, 2-32
parameter	SCOTT, 2-36
about, 28-5	SH, 2-36
using, 28-6	SI_INFORMTN_SCHEMA, 2-32
UNIFIED_AUDIT_TRAIL data dictionary view	SYS, 2-32
best practices for using, A-22	SYS\$UMF, 2-32
UNLIMITED TABLESPACE privilege, 2-12	SYSBACKUP, 2-32
UPDATE privilege	SYSDG, 2-32
revoking, 4-78	SYSKM, 2-32
user accounts	SYSTEM, 2-32
administrative user passwords, A-7	WMSYS, 2-32
application common user	XDB, 2-32
about, 2-3	XS\$NULL, 2-35
CDB common user	User Global Area (UGA), 13-2
about, 2-3	application contexts, storing in, 13-2
common	user names
creating, 2-15	schemas, <i>12-23</i>
default user account, A-7	user privileges
local	CDBs, 4-3
creating, 2-16	USER pseudo column, 4-61
local user	user sessions, multiple within single database
about, 2-5	connection, 3-83
password guidelines, A-7	USERENV function
passwords, encrypted, A-7	used in views, 9-8
predefined	USERENV namespace, 3-89
administrative, 2-32	about, <i>13-11</i>
non-administrative, 2-35	See also CLIENT_IDENTIFIER USERENV
sample schema, 2-36	attribute
predefined schema, 2-31	users
privileges required to create, 2-6	administrative option (ADMIN OPTION), 4-71
proxy users, 3-80	altering, 2-18
user accounts, predefined	altering common users, 2-18
ANONYMOUS, 2-32	altering local users, 2-18
APPQOSSYS, 2-32	application users not known to database, 3-88
ASMSNMP, 2-32	assigning unlimited quotas for, 2-12
AUDSYS, 2-32	auditing, <i>27-80</i>
CTXSYS, 2-32	database role, current, 12-22
DBSFWUSER, 2-32	default roles, changing, 2-17
DBSNMP, 2-32	default tablespaces, 2-9
DIP, 2-35	dropping, 2-29, 2-30
GSMROOTUSER, 2-32	dropping profiles and, 2-29
HR, 2-36	dropping roles and, 4-42
IX, 2-36	enabling roles for, 12-22

users (continued)	users (continued)
enterprise, 3-70, 4-40	user name, specifying with CREATE USER
enterprise, shared schema protection, 12-24	statement, 2-8
external authentication	views for finding information about, 2-36
about, 3-73	users supported, 6-4
advantages, 3-73	utlpwdmg.sql
assigning profiles, 2-28	about, 3-26
operating system, 3-75	
user creation, 3-74	\/
finding information about, 2-37	V
finding information about authentication, 3-92	valid node checking, A-14
global, 3-70	views, 4-86
assigning profiles, 2-28	about, 4-60
hosts, connecting to multiple	access control list data
See external network services, fine-	external network services, 10-22
grained access to, 10-2	wallet access, 10-22
information about, viewing, 2-38	application contexts, 13-55
listing roles granted to, 4-88	audit management settings, 28-25
memory use, viewing, 2-40	audit trail usage, <i>27-111</i>
names	audited activities, 27-111
case sensitivity, 2-8	auditing, 27-14
how stored in database, 2-8	authentication, 3-92
network authentication, external, 3-75	bind variables in TSDP sensitive columns,
nondatabase, 13-29, 13-37	15-20
objects after dropping, 2-29	
operating system external authentication,	DBA_COL_PRIVS, 4-89
3-75	DBA_HOST_ACES, 10-22
password encryption, 3-3	DBA_HOST_ACLS, 10-22
privileges	DBA_ROLE_PRIVS, 4-88
	DBA_ROLES, 4-90
for changing passwords, 2-18	DBA_SYS_PRIVS, 4-88
for creating, 2-6	DBA_TAB_PRIVS, 4-89
granted to, listing, 4-88	DBA_USERS_WITH_DEFPWD, 3-6
of current database role, 12-22	DBA_WALLET_ACES, 10-22
profiles	DBA_WALLET_ACLS, 10-22
assigning, 2-28	definer's rights, 9-8
creating, 2-27	encrypted data, 17-20
specifying, 2-13	invoker's rights, 9-8
profiles, CDB or application, 2-28	Oracle Virtual Private Database policies,
proxy authentication, 3-79	14-50
proxy users, connecting as, 3-79	privileges, 4-60, 4-86
PUBLIC role, 4-27, 4-80	privileges to query views in other schemas,
quota limits for tablespace, 2-12	4-60
restricting application roles, 4-42	profiles, 2-37
restrictions on user names, 2-8	ROLE_SYS_PRIVS, 4-91
roles and, 4-25	ROLE_TAB_PRIVS, 4-91
for types of users, 4-27	security applications of, 4-61
schema-independent, 12-24	SESSION_PRIVS, 4-89
schemas, private, 3-71	SESSION_ROLES, 4-89
security domains of, 4-27	transparent sensitive data protection, 15-34
security, about, 2-1	USER_HOST_ACES, 10-22
tablespace quotas, 2-11	USER_WALLET_ACES, 10-22
tablespace quotas, viewing, 2-38	users, 2-37
user accounts, creating, 2-6	Virtual Private Database
user models and Oracle Virtual Private	See Oracle Virtual Private Database
Database, 14-49	VPD
	See Oracle Virtual Private Database

vulnerable run-time call, <i>A-3</i> made more secure, <i>A-3</i> W Wallet Manager See Oracle Wallet Manager wallets, <i>10-2</i> , <i>23-7</i> adding certificate to, <i>6-16</i> authentication method, <i>3-67</i> certificates adding to wallet, <i>6-16</i> See also access control lists (ACL), wallet access Web applications user connections, <i>13-29</i> , <i>13-37</i> Web-based applications Oracle Virtual Private Database, how it works with, <i>14-49</i> WFS_USR_ROLE role, <i>4-29</i> WHEN OTHERS exceptions logon triggers, used in, <i>13-16</i> Windows Event Viewer capturing audit trail records, <i>28-6</i>	Windows Event Viewer (continued) Windows native authentication, 3-58 WITH GRANT OPTION clause about, 4-73 user and role grants, 4-52 WM_ADMIN_ROLE role, 4-29 WMSYS user account, 2-32

