Oracle® Database

2 Day + Security Guide

11*g* Release 2 (11.2)

E10575-06

October 2010

ORACLE®

Oracle Database 2 Day + Security Guide, 11*g* Release 2 (11.2)

E10575-06

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

Primary Author: Patricia Huey

Contributors: Naveen Gopal, Rahil Mir, Gopal Mulagund, Nina Lewis, Paul Needham, Deborah Owens, Sachin Sonawane, Ashwini Surpur, Kamal Tbeileh, Mark Townsend, Peter Wahl, Xiaofang Wang, Peter M. 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 software or related documentation is delivered to the U.S. Government or anyone licensing it on behalf of the U.S. Government, the following notice is applicable:

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

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

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

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

Contents

Preface ix

Audience ix
Documentation Accessibility ix
Related Documents x
Conventions xi

1 Introduction to Oracle Database Security

About This Guide 1-1
Before Using This Guide 1-1
What This Guide Is and Is Not 1-1
Common Database Security Tasks 1-2
Tools for Securing Your Database 1-2
Securing Your Database: A Roadmap 1-3

2 Securing the Database Installation and Configuration

About Securing the Database Installation and Configuration 2-1
Using the Default Security Settings 2-1
Securing the Oracle Data Dictionary 2-2
About the Oracle Data Dictionary 2-2
Enabling Data Dictionary Protection 2-3
Guidelines for Securing Operating System Access to Oracle Database 2-4
Guideline for Granting Permissions to Run-Time Facilities 2-5
Initialization Parameters Used for Installation and Configuration Security 2-5
Modifying the Value of an Initialization Parameter 2-6

3 Securing Oracle Database User Accounts

About Securing Oracle Database User Accounts 3-1
Predefined User Accounts Provided by Oracle Database 3-2
Predefined Administrative Accounts 3-2
Predefined Non-Administrative User Accounts 3-5
Predefined Sample Schema User Accounts 3-6
Expiring and Locking Database Accounts 3-7
Requirements for Creating Passwords 3-8
Finding and Changing Default Passwords 3-9
Guideline for Handling the Default Administrative User Passwords 3-10

Guideline for Enforcing Password Management 3-11
Parameters Used to Secure User Accounts 3-12

4 Managing User Privileges

About Privilege Management 4-1
Guideline for Granting Privileges 4-1
Guideline for Granting Roles to Users 4-2
Guideline for Handling Privileges for the PUBLIC User Group 4-2
Controlling Access to Applications with Secure Application Roles 4-2
About Secure Application Roles 4-3
Tutorial: Creating a Secure Application Role 4-3
Step 1: Create a Security Administrator Account 4-4
Step 2: Create User Accounts for This Tutorial 4-5
Step 3: Create the Secure Application Role 4-6
Step 4: Create a Lookup View 4-7
Step 5: Create the PL/SQL Procedure to Set the Secure Application Role 4-8
Step 6: Grant the EXECUTE Privilege for the Procedure to Matthew and Winston 4-10
Step 7: Test the EMPLOYEE_ROLE Secure Application Role 4-10
Step 8: Optionally, Remove the Components for This Tutorial 4-11
Initialization Parameters Used for Privilege Security 4-12

5 Securing the Network

About Securing the Network 5-1
Securing the Client Connection on the Network 5-1
Guidelines for Securing Client Connections 5-1
Guidelines for Securing the Network Connection 5-2
Protecting Data on the Network by Using Network Encryption 5-5
About Network Encryption 5-5
Configuring Network Encryption 5-5
Initialization Parameters Used for Network Security 5-8

6 Securing Data

About Securing Data 6-1
Encrypting Data Transparently with Transparent Data Encryption 6-2
About Encrypting Sensitive Data 6-2
When Should You Encrypt Data? 6-2
How Transparent Data Encryption Works 6-3
Configuring Data to Use Transparent Data Encryption 6-4
Step 1: Configure the Wallet Location 6-4
Step 2: Create the Wallet 6-5
Step 3: Open (or Close) the Wallet 6-6
Step 4: Encrypt (or Decrypt) Data 6-6
Checking Existing Encrypted Data 6-9
Checking Whether a Wallet Is Open or Closed 6-10
Checking Encrypted Columns of an Individual Table 6-10
Checking All Encrypted Table Columns in the Current Database Instance 6-10

Checking Encrypted Tablespaces in the Current Database Instance 6-11
Choosing Between Oracle Virtual Private Database and Oracle Label Security 6-11
Controlling Data Access with Oracle Virtual Private Database 6-12
About Oracle Virtual Private Database 6-13
Tutorial: Creating an Oracle Virtual Private Database Policy 6-14
Step 1: If Necessary, Create the Security Administrator Account 6-15
Step 2: Update the Security Administrator Account 6-15
Step 3: Create User Accounts for This Tutorial 6-16
Step 4: Create the F_POLICY_ORDERS Policy Function 6-17
Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy 6-19
Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy 6-20
Step 7: Optionally, Remove the Components for This Tutorial 6-21
Enforcing Row-Level Security with Oracle Label Security 6-21
About Oracle Label Security 6-22
Guidelines for Planning an Oracle Label Security Policy 6-23
Tutorial: Applying Security Labels to the HR.LOCATIONS Table 6-24
Step 1: Register Oracle Label Security and Enable the LBACSYS Account 6-25
Step 2: Create a Role and Three Users for the Oracle Label Security Tutorial 6-28
Step 3: Create the ACCESS_LOCATIONS Oracle Label Security Policy 6-30
Step 4: Define the ACCESS_LOCATIONS Policy-Level Components 6-31
Step 5: Create the ACCESS_LOCATIONS Policy Data Labels 6-32
Step 6: Create the ACCESS_LOCATIONS Policy User Authorizations 6-33
Step 7: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table 6-35
Step 8: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data 6-35
Step 9: Test the ACCESS_LOCATIONS Policy 6-38
Step 10: Optionally, Remove the Components for This Tutorial 6-40
Controlling Administrator Access with Oracle Database Vault 6-41
About Oracle Database Vault 6-41
Tutorial: Controlling Administrator Access to the OE Schema 6-42
Step 1: Enable Oracle Database Vault 6-42
Step 2: Grant the SELECT Privilege on the OE.CUSTOMERS Table to User SCOTT 6-45
Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT 6-46
Step 4: Create a Realm to Protect the OE.CUSTOMERS Table 6-47
Step 5: Test the OE Protections Realm 6-49
Step 6: Optionally, Remove the Components for This Tutorial 6-49

7 Auditing Database Activity

About Auditing 7-1
Why Is Auditing Used? 7-2
Where Are Standard Audit Activities Recorded? 7-2
Auditing General Activities Using Standard Auditing 7-3
About Standard Auditing 7-3
Enabling or Disabling the Standard Audit Trail 7-3
Using Default Auditing for Security-Relevant SQL Statements and Privileges 7-5
Individually Auditing SQL Statements 7-6
Individually Auditing Privileges 7-6
Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment 7-7

Individually Auditing Schema Objects 7-7
Auditing Network Activity 7-7
Tutorial: Creating a Standard Audit Trail 7-8
Step 1: Log In and Enable Standard Auditing 7-8
Step 2: Enable Auditing for SELECT Statements on the OE.CUSTOMERS Table 7-9
Step 3: Test the Audit Settings 7-10
Step 4: Optionally, Remove the Components for This Tutorial 7-11
Step 5: Remove the SEC_ADMIN Security Administrator Account 7-11
Guidelines for Auditing 7-12
Guideline for Using Default Auditing of SQL Statements and Privileges 7-12
Guidelines for Managing Audited Information 7-12
Guidelines for Auditing Typical Database Activity 7-13
Guidelines for Auditing Suspicious Database Activity 7-13
Initialization Parameters Used for Auditing 7-14

Index

List of Tables

2–1 Default Security Settings for Initialization and Profile Parameters 2-2
2–2 Initialization Parameters Used for Installation and Configuration Security 2-5
3–1 Predefined Oracle Database Administrative User Accounts 3-2
3–2 Predefined Oracle Database Non-Administrative User Accounts 3-5
3–3 Default Sample Schema User Accounts 3-7
3–4 Initialization and Profile Parameters Used for User Account Security 3-12
4–1 Initialization Parameters Used for Privilege Security 4-12
5–1 Initialization Parameters Used for Network Security 5-8
6–1 Data Dictionary Views for Encrypted Tablespaces 6-11
6–2 Comparing Oracle Virtual Private Database with Oracle Label Security 6-12
7–1 Initialization Parameters Used for Auditing 7-14

Preface

Welcome to *Oracle Database 2 Day + Security Guide*. This guide is for anyone who wants to perform common day-to-day security tasks with Oracle Database.

This preface contains:

- Audience
- Documentation Accessibility
- Related Documents
- Conventions

Audience

Oracle Database 2 Day + Security Guide expands on the security knowledge that you learned in *Oracle Database 2 Day DBA* to manage security in Oracle Database. The information in this guide applies to all platforms. For platform-specific information, see the installation guide, configuration guide, and platform guide for your platform.

This guide is intended for the following users:

- Oracle database administrators who want to acquire database security administrative skills
- Database administrators who have some security administrative knowledge but are new to Oracle Database

This guide is not an exhaustive discussion about security. For detailed information about security, see the Oracle Database Security documentation set. This guide does not provide information about security for Oracle E-Business Suite applications. For information about security in the Oracle E-Business Suite applications, see the documentation for those products.

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

Access to Oracle Support

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

Related Documents

For more information, use the following resources:

Oracle Database Documentation

For more security-related information, see the following documents in the Oracle Database documentation set:

- *Oracle Database 2 Day DBA*
- *Oracle Database Administrator's Guide*
- *Oracle Database Security Guide*
- *Oracle Database Concepts*
- *Oracle Database Reference*
- *Oracle Database Vault Administrator's Guide*

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

Oracle Technology Network (OTN)

You can download free release notes, installation documentation, updated versions of this guide, white papers, or other collateral from the Oracle Technology Network (OTN). Visit:

`http://www.oracle.com/technology/index.html`

If you are not a member, you can register for free at:

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

For security-specific information about OTN, visit:

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

For the latest version of the Oracle documentation, including this guide, visit:

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

Oracle Documentation Search Engine

To access the database documentation search engine directly, visit:

http://tahiti.oracle.com/

My Oracle Support (formerly Oracle*MetaLink*)

You can find information about security patches, certifications, and the support knowledge base by visiting My Oracle Support at:

https://support.oracle.com

Conventions

The following text conventions are used in this document:

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

1

Introduction to Oracle Database Security

This chapter contains:

- About This Guide
- Common Database Security Tasks
- Tools for Securing Your Database
- Securing Your Database: A Roadmap

About This Guide

Oracle Database 2 Day + Security Guide teaches you how to perform day-to-day database security tasks. Its goal is to help you understand the concepts behind Oracle Database security. You will learn how to perform common security tasks needed to secure your database. The knowledge you gain from completing the tasks in *Oracle Database 2 Day + Security Guide* helps you to better secure your data and to meet common regulatory compliance requirements, such as the Sarbanes-Oxley Act.

The primary administrative interface used in this guide is Oracle Enterprise Manager in Database Console mode, featuring all the self-management capabilities introduced in Oracle Database.

This section contains the following topics:

- Before Using This Guide
- What This Guide Is and Is Not

Before Using This Guide

Before using this guide:

- Complete *Oracle Database 2 Day DBA*
- Obtain the necessary products and tools described in "Tools for Securing Your Database" on page 1-2

What This Guide Is and Is Not

Oracle Database 2 Day + Security Guide is task oriented. The objective of this guide is to describe why and when you must perform security tasks.

Where appropriate, this guide describes the concepts and steps necessary to understand and complete a task. This guide is not an exhaustive discussion of all Oracle Database concepts. For this type of information, see *Oracle Database Concepts*.

Where appropriate, this guide describes the necessary Oracle Database administrative steps to complete security tasks. This guide does not describe basic Oracle Database administrative tasks. For this type of information, see *Oracle Database 2 Day DBA*. Additionally, for a complete discussion of administrative tasks, see *Oracle Database Administrator's Guide*.

In addition, this guide is not an exhaustive discussion of all Oracle Database security features and does not describe available APIs that provide equivalent command line functionality to the tools used in this guide. For this type of information, see *Oracle Database Security Guide*.

Common Database Security Tasks

As a database administrator for Oracle Database, you should be involved in the following security-related tasks:

- Ensuring that the database installation and configuration is secure
- Managing the security aspects of user accounts: developing secure password policies, creating and assigning roles, restricting data access to only the appropriate users, and so on
- Ensuring that network connections are secure
- Encrypting sensitive data
- Ensuring the database has no security vulnerabilities and is protected against intruders
- Deciding what database components to audit and how granular you want this auditing to be
- Downloading and installing security patches

In a small to midsize database environment, you might perform these tasks as well and all database administrator-related tasks, such as installing Oracle software, creating databases, monitoring performance, and so on. In large, enterprise environments, the job is often divided among several database administrators—each with their own specialty—such as database security or database tuning.

Tools for Securing Your Database

To achieve the goals of securing your database, you need the following products, tools, and utilities:

- **Oracle Database 11*g* Release 2 (11.2) Enterprise Edition**

  Oracle Database 11*g* Release 2 (11.2) Enterprise Edition provides enterprise-class performance, scalability, and reliability on clustered and single-server configurations. It includes many security features that are used in this guide.

- **Oracle Enterprise Manager Database Control**

  Oracle Enterprise Manager is a Web application that you can use to perform database administrative tasks for a single database instance or a clustered database.

- **SQL*Plus**

  SQL*Plus is a development environment that you can use to create and run SQL and PL/SQL code. It is part of the Oracle Database 11*g* Release 2 (11.2) installation.

- **Database Configuration Assistant (DBCA)**

  Database Configuration Assistant enables you to perform general database tasks, such as creating, configuring, or deleting databases. In this guide, you use DBCA to enable default auditing.

- **Oracle Net Manager**

  Oracle Net Manager enables you to perform network-related tasks for Oracle Database. In this guide, you use Oracle Net Manager to configure network encryption.

Securing Your Database: A Roadmap

To learn the fundamentals of securing an Oracle database, follow these steps:

1. **Secure your Oracle Database installation and configuration.**

   Complete the tasks in Chapter 2, "Securing the Database Installation and Configuration" to secure access to an Oracle Database installation.

2. **Secure user accounts for your site.**

   Complete the tasks in Chapter 3, "Securing Oracle Database User Accounts", which builds on *Oracle Database 2 Day DBA*, where you learned how to create user accounts. You learn the following:

   - How to expire, lock, and unlock user accounts
   - Guidelines to choose secure passwords
   - How to change a password
   - How to enforce password management

3. **Understand how privileges work.**

   Complete the tasks in Chapter 4, "Managing User Privileges". You learn about the following:

   - How privileges work
   - Why you must be careful about granting privileges
   - How database roles work
   - How to create secure application roles

4. **Secure data as it travels across the network.**

   Complete the tasks in Chapter 5, "Securing the Network" to learn how to secure client connections and to configure network encryption.

5. **Control access to data.**

   Complete the tasks in Chapter 6, "Securing Data", in which you learn about the following:

   - How to use transparent data encryption to automatically encrypt database table columns and tablespaces
   - How to control data access with Oracle Virtual Private Database
   - How to enforce row-level security with Oracle Label Security
   - How to control system administrative access to sensitive data with Oracle Database Vault.

6. **Configure auditing so that you can monitor the database activities.**

   Complete the tasks in Chapter 7, "Auditing Database Activity" to learn about standard auditing.

2

Securing the Database Installation and Configuration

This chapter contains:

- About Securing the Database Installation and Configuration
- Using the Default Security Settings
- Securing the Oracle Data Dictionary
- Guidelines for Securing Operating System Access to Oracle Database
- Guideline for Granting Permissions to Run-Time Facilities
- Initialization Parameters Used for Installation and Configuration Security

About Securing the Database Installation and Configuration

After you install Oracle Database, you should secure the database installation and configuration. The methods in this chapter describe commonly used ways to do this, all of which involve restricting permissions to specific areas of the database files.

Oracle Database is available on several operating systems. Consult the following guides for detailed platform-specific information about Oracle Database:

- *Oracle Database Platform Guide for Microsoft Windows*
- *Oracle Database Administrator's Reference for Linux and UNIX*
- *Oracle Database Installation Guide* for your platform

Using the Default Security Settings

When you create a new database, Oracle Database provides the following default security settings:

- **Enables default auditing settings.** See "Using Default Auditing for Security-Relevant SQL Statements and Privileges" on page 7-5 for detailed information.
- **Creates stronger enforcements for new or changed passwords.** "Requirements for Creating Passwords" on page 3-8 describes the new password requirements.
- **Removes the CREATE EXTERNAL JOB privilege from PUBLIC.** For greater security, grant the `CREATE EXTERNAL JOB` privilege only to `SYS`, database administrators, and those trusted users who need it.

- **Sets security-related initialization and profile parameter settings.** Table 2–1 lists the modified parameter settings.

Table 2–1 Default Security Settings for Initialization and Profile Parameters

Setting	10*g* Default	11*g* Default
AUDIT_TRAIL	NONE	DB
O7_DICTIONARY_ACCESSIBILITY	FALSE	FALSE
PASSWORD_GRACE_TIME	UNLIMITED	7
PASSWORD_LOCK_TIME	UNLIMITED	1
FAILED_LOGIN_ATTEMPTS	10	10
PASSWORD_LIFE_TIME	UNLIMITED	180
PASSWORD_REUSE_MAX	UNLIMITED	UNLIMITED
PASSWORD_REUSE_TIME	UNLIMITED	UNLIMITED
REMOTE_OS_ROLES	FALSE	FALSE

Note: If your applications use the default password security settings from Oracle Database 10*g* Release 2 (10.2), then you can revert to these settings until you modify them to use the Release 11*g* password security settings. To do so, run the `undopwd.sql` script.

After you have modified your applications to conform to the Release 11*g* password security settings, you can manually update your database to use the password security configuration that suits your business needs, or you can run the `secconf.sql` script to apply the Release 11g default password settings.

The `undopwd.sql` and `secconf.sql` scripts are in the `$ORACLE_HOME/rdbms/admin` directory. The `undopwd.sql` script affects password settings only, and the `secconf.sql` script affects both password and audit settings. They have no effect on other security settings.

Securing the Oracle Data Dictionary

This section describes how you can secure the data dictionary. The data dictionary is a set of database tables that provide information about the database, such as schema definitions or default values.

This section contains:

- About the Oracle Data Dictionary
- Enabling Data Dictionary Protection

About the Oracle Data Dictionary

The Oracle data dictionary is a set of database tables that provides information about the database. A data dictionary has the following contents:

- The names of Oracle Database users
- Privileges and roles granted to each user

- The definitions of all schema objects in the database (tables, views, indexes, clusters, synonyms, sequences, procedures, functions, packages, triggers, and so on)
- The amount of space allocated for, and is currently used by, the schema objects
- Default values for columns
- Integrity constraint information
- Auditing information, such as who has accessed or updated various schema objects
- Other general database information

The data dictionary tables and views for a given database are stored in the `SYSTEM` tablespace for that database. All the data dictionary tables and views for a given database are owned by the user `SYS`. Connecting to the database with the `SYSDBA` privilege gives full access to the data dictionary. Oracle strongly recommends limiting access to the `SYSDBA` privilege to only those operations necessary such as patching and other administrative operations. The data dictionary is central to every Oracle database.

You can view the contents of the data dictionary by querying data dictionary views, which are described in *Oracle Database Reference*. Be aware that not all objects in the data dictionary are exposed to users. A subset of data dictionary objects, such as those beginning with `USER_%` are exposed as read only to all database users.

Example 2–1 shows how you can find a list of database views specific to the data dictionary by querying the `DICTIONARY` view.

Example 2–1 Finding Views That Pertain to the Data Dictionary

```
sqlplus system
Enter password: password

SQL> SELECT TABLE_NAME FROM DICTIONARY;
```

Enabling Data Dictionary Protection

You can protect the data dictionary by setting the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter to `FALSE`. This parameter prevents users who have the `ANY` system privilege from using those privileges on the data dictionary, that is, on objects in the `SYS` schema.

Oracle Database provides highly granular privileges. One such privilege, commonly referred to as the `ANY` privilege, is typically granted to only application owners and individual database administrators. For example, you could grant the `DROP ANY TABLE` privilege to an application owner. You can protect the Oracle data dictionary from accidental or malicious use of the `ANY` privilege by turning on or off the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter.

To enable data dictionary protection:

1. Start Oracle Enterprise Manager Database Control (Database Control).

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Log in as `SYS` and connect with the `SYSDBA` privilege.

   - **User Name**: Enter the name of a user who has administrative privileges. In this case, you enter `SYS`.

- **Password**: Enter the `SYS` user's password.
- **Connect As**: From the list, select **SYSDBA**.

The Oracle Enterprise Manager Database Home page (Database Home page) appears.

3. Click **Server** to display the Server subpage.
4. In the Database Configuration section, click **Initialization Parameters**.

   The Initialization Parameters page appears.
5. In the list, search for `O7_DICTIONARY_ACCESSIBILITY`.

   In the **Name** field, enter `O7_` (the letter `O`), and then click **Go**. You can enter the first few characters of a parameter name. In this case, `O7_` displays the `O7_DICTIONARY_ACCESSIBILTY` parameter.

   Depending on the parameter, you may have to modify the value from the SPFile subpage. Click the **SPFile** tab to display the SPFile subpage.
6. Set the value for `O7_DICTIONARY_ACCESSIBILTY` to `FALSE`.
7. Click **Apply**.
8. Restart the Oracle Database instance.
   a. Click the **Database Instance** link.
   b. Click **Home** to display the Database Control home page.
   c. Under General, click **Shutdown**.
   d. In the Startup/Shutdown Credentials page, enter your credentials.

      See *Oracle Database 2 Day DBA* for more information.
   e. After the shutdown completes, click **Startup**.

Note:

- In a default installation, the `O7_DICTIONARY_ACCESSIBILITY` parameter is set to `FALSE`.
- The `SELECT ANY DICTIONARY` privilege is not included in the `GRANT ALL PRIVILEGES` statement, but you can grant it through a role. Roles are described in "Guideline for Granting Roles to Users" on page 4-2 and *Oracle Database 2 Day DBA*.

Guidelines for Securing Operating System Access to Oracle Database

You can secure access to Oracle Database on the operating system level by following these guidelines:

- Limit the number of operating system users.
- Limit the privileges of the operating system accounts (administrative, root-privileged, or DBA) on the Oracle Database host (physical computer). Only grant the user the least number of privileges needed to perform his or her 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 some trusted account, such as *root*.

  This recommendation applies to all types of files: data files, log files, trace files, external tables, BFILEs, and so on.

Guideline for Granting Permissions to 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 root file paths for facilities that might run files and packages outside the database.

Here is an example of a vulnerable run-time call, in which an individual file (in ***bold*** typeface) is specified:

```
call dbms_java.grant_permission('wsmith',
 'SYS:java.io.FilePermission','filename','read');
```

The following example is a better (more secure) run-time call, because by specifying a directory path (in ***bold*** typeface), it protects *all* files within the directory.

```
call dbms_java.grant_permission('wsmith',
 'SYS:java.io.FilePermission','directory_path','read');
```

Initialization Parameters Used for Installation and Configuration Security

Table 2–2 lists initialization parameters that you can set to better secure your Oracle Database installation and configuration.

Table 2–2 Initialization Parameters Used for Installation and Configuration Security

Initialization Parameter	Default Setting	Description
SEC_RETURN_SERVER_RELEASE_BANNER	FALSE	Controls the display of the product version information, such as the release number, in a client connection. An intruder could use the database release number to find information about security vulnerabilities that may be present in the database software. You can enable or disable the detailed product version display by setting this parameter. See *Oracle Database Security Guide* for more information about this and similar parameters. *Oracle Database Reference* describes this parameter in detail.
O7_DICTIONARY_ACCESSIBILITY	FALSE	Controls restrictions on SYSTEM privileges. See "Enabling Data Dictionary Protection" on page 2-3 for more information about this parameter. *Oracle Database Reference* describes this parameter in detail.

See Also: *Oracle Database Reference* for more information about initialization parameters

Modifying the Value of an Initialization Parameter

This section explains how to use Database Control to modify the value of an initialization parameter. To find detailed information about the initialization parameters available, see *Oracle Database Reference*.

To modify the value of an initialization parameter:

1. Start Database Control.
2. Log in as user `SYS` with the `SYSDBA` privilege.
   - **User Name**: `SYS`
   - **Password**: Enter your password.
   - **Connect As**: `SYSDBA`
3. Click **Server** to display the Server subpage.
4. In the Database Configuration section, click **Initialization Parameters**.

   The Initialization Parameters page appears.
5. In the **Name** field, enter the name of the parameter to change, and then click **Go**.

   You can enter the first few letters of the parameter, for example, `SEC_RETURN` if you are searching for the `SEC_RETURN_SERVER_RELEASE_NUMBER` parameter. Alternatively, you can scroll down the list of parameters to find the parameter you want to change.

   Depending on the parameter, you might have to modify the value from the SPFile subpage. Click the **SPFile** tab to display the SPFile subpage.
6. In the **Value** field, either enter the new value or if a list is presented, select from the list.
7. Click **Apply**.
8. If the parameter is static, restart the Oracle Database instance.

   To find out if an initialization parameter is static, check its description in *Oracle Database Reference*. If the Modifiable setting in its summary table shows No, then you must restart the database instance.

   a. Click the **Database Instance** link.

   b. Click **Home** to display the Database Control home page.

   c. Under General, click **Shutdown**.

   d. In the Startup/Shutdown Credentials page, enter your credentials.

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

   e. After the shutdown completes, click **Startup**.

3

Securing Oracle Database User Accounts

This chapter contains:

- About Securing Oracle Database User Accounts
- Predefined User Accounts Provided by Oracle Database
- Expiring and Locking Database Accounts
- Requirements for Creating Passwords
- Finding and Changing Default Passwords
- Guideline for Handling the Default Administrative User Passwords
- Guideline for Enforcing Password Management
- Parameters Used to Secure User Accounts

> **See Also:** *Oracle Database Security Guide* for detailed information about securing user accounts

About Securing Oracle Database User Accounts

You can use many methods to secure database user accounts. For example, Oracle Database has a set of built-in protections for passwords. This chapter explains how you can safeguard default database accounts and passwords, and describes ways to manage database accounts.

Oracle Database 2 Day DBA describes the fundamentals of creating and administering user accounts, including how to manage user roles, what the administrative accounts are, and how to use profiles to establish a password policy.

After you create user accounts, you can use the procedures in this section to further secure these accounts by following these methods:

- **Safeguarding predefined database accounts.** When you install Oracle Database, it creates a set of predefined accounts. You should secure these accounts as soon as possible by changing their passwords. You can use the same method to change all passwords, whether they are with regular user accounts, administrative accounts, or predefined accounts. This guide also provides guidelines on how to create the most secure passwords.
- **Managing database accounts.** You can expire and lock database accounts.
- **Managing passwords.** You can manage and protect passwords by setting initialization parameters. *Oracle Database Reference* describes the initialization parameters in detail.

See Also:

- *Oracle Database Security Guide* for detailed information about managing user accounts and authentication
- "Predefined User Accounts Provided by Oracle Database" on page 3-2 for a description of the predefined user accounts that are created when you install Oracle Database

Predefined User Accounts Provided by Oracle Database

When you install Oracle Database, the installation process creates a set of predefined accounts. These accounts are in the following categories:

- Predefined Administrative Accounts
- Predefined Non-Administrative User Accounts
- Predefined Sample Schema User Accounts

Predefined Administrative Accounts

A default Oracle Database installation provides a set of predefined administrative accounts. These are accounts that have special privileges required to administer areas of the database, such as the `CREATE ANY TABLE` or `ALTER SESSION` privilege, or `EXECUTE` privileges on packages owned by the `SYS` schema. The default tablespace for administrative accounts is either `SYSTEM` or `SYSAUX`.

To protect these accounts from unauthorized access, the installation process expires and locks most of these accounts, except where noted in Table 3–1. As the database administrator, you are responsible for unlocking and resetting these accounts, as described in "Expiring and Locking Database Accounts" on page 3-7.

Table 3–1 lists the administrative user accounts provided by Oracle Database.

Table 3–1 Predefined Oracle Database Administrative User Accounts

User Account	Description	Status After Installation
`ANONYMOUS`	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.	Expired and locked
`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*.	Expired and locked
`DBSNMP`	The account used by the Management Agent component of Oracle Enterprise Manager to monitor and manage the database. See *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*.	Open Password is created at installation or database creation time.

Table 3–1 (Cont.) Predefined Oracle Database Administrative User Accounts

User Account	Description	Status After Installation
EXFSYS	The account used internally to access the EXFSYS schema, which is associated with the Rules Manager and Expression Filter feature. This feature enables you to build complex PL/SQL rules and expressions. The EXFSYS schema contains the Rules Manager and Expression Filter DDL, DML, and associated metadata. See *Oracle Database Rules Manager and Expression Filter Developer's Guide*.	Expired and locked
LBACSYS	The account used to administer Oracle Label Security (OLS). It is created only when you install the Label Security custom option. See "Enforcing Row-Level Security with Oracle Label Security" on page 6-21 and *Oracle Label Security Administrator's Guide*.	Expired and locked
MDSYS	The Oracle Spatial and Oracle Multimedia Locator administrator account. See *Oracle Spatial Developer's Guide*.	Expired and locked
MGMT_VIEW	An account used by Oracle Enterprise Manager Database Control.	Open Password is randomly generated at installation or database creation time. Users do not need to know this password.
OLAPSYS	The account that owns the OLAP Catalog (CWMLite). This account has been deprecated, but is retained for backward compatibility.	Expired and locked
ORDDATA	This account contains the Oracle Multimedia DICOM data model. See *Oracle Multimedia DICOM Developer's Guide* for more information.	Expired and locked
OWBSYS	The account for administrating the Oracle Warehouse Builder repository. Access this account during the installation process to define the base language of the repository and to define Warehouse Builder workspaces and users. A data warehouse is a relational or multidimensional database that is designed for query and analysis. See *Oracle Warehouse Builder Installation and Administration Guide*.	Expired and locked
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* and *Oracle Multimedia Reference*.	Expired and locked
ORDSYS	The Oracle Multimedia administrator account. See *Oracle Multimedia User's Guide*, *Oracle Multimedia Reference*, and *Oracle Multimedia DICOM Developer's Guide*.	Expired and locked

Table 3–1 (Cont.) Predefined Oracle Database Administrative User Accounts

User Account	Description	Status After Installation
OUTLN	The account that supports plan stability. Plan stability prevents certain database environment changes from affecting the performance characteristics of applications by preserving execution plans in stored outlines. OUTLN acts as a role to centrally manage metadata associated with stored outlines. See *Oracle Database Performance Tuning Guide*.	Expired and locked
SI_INFORMTN_SCHEMA	The account that stores the information views for the SQL/MM Still Image Standard. See *Oracle Multimedia User's Guide* and *Oracle Multimedia Reference*.	Expired and locked
SYS	An account used to perform database administration tasks. See *Oracle Database 2 Day DBA*.	Open Password is created at installation or database creation time.
SYSMAN	The account used to perform Oracle Enterprise Manager database administration tasks. The SYS and SYSTEM accounts can also perform these tasks. See *Oracle Enterprise Manager Grid Control Installation and Basic Configuration*.	Open Password is created at installation or database creation time.
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*.	Open Password is created at installation or database creation time.
WK_TEST	The instance administrator for the default instance, WK_INST. After you unlock this account and assign this user a password, then you must also update the cached schema password using the administration tool Edit Instance Page. Ultra Search provides uniform search-and-location capabilities over multiple repositories, such as Oracle databases, other ODBC compliant databases, IMAP mail servers, HTML documents managed by a Web server, files on disk, and more. See *Oracle Ultra Search Administrator's Guide*.	Expired and locked
WKSYS	An Ultra Search database super-user. WKSYS can grant super-user privileges to other users, such as WK_TEST. All Oracle Ultra Search database objects are installed in the WKSYS schema. See *Oracle Ultra Search Administrator's Guide*.	Expired and locked

Table 3–1 (Cont.) Predefined Oracle Database Administrative User Accounts

User Account	Description	Status After Installation
WKPROXY	An administrative account of Oracle9*i* Application Server Ultra Search. See *Oracle Ultra Search Administrator's Guide*.	Expired and locked
WMSYS	The account used to store the metadata information for Oracle Workspace Manager. See *Oracle Database Workspace Manager Developer's Guide*.	Expired and locked
XDB	The account used for storing Oracle XML DB data and metadata. Oracle XML DB provides high-performance XML storage and retrieval for Oracle Database data. See *Oracle XML DB Developer's Guide*.	Expired and locked

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 privilege. For more information, see *Oracle Database Storage Administrator's Guide*.

Predefined Non-Administrative User Accounts

Table 3–2 lists default non-administrative user accounts that are created when you install Oracle Database. Non-administrative user accounts only have the minimum privileges needed to perform their jobs. Their default tablespace is USERS.

To protect these accounts from unauthorized access, the installation process locks and expires these accounts immediately after installation, except where noted in Table 3–2. As the database administrator, you are responsible for unlocking and resetting these accounts, as described in "Expiring and Locking Database Accounts" on page 3-7.

Table 3–2 Predefined Oracle Database Non-Administrative User Accounts

User Account	Description	Status After Installation
APEX_PUBLIC_USER	The Oracle Database Application Express account. Use this account to specify the Oracle schema used to connect to the database through the database access descriptor (DAD). Oracle Application Express is a rapid, Web application development tool for Oracle Database. See *Oracle Database Application Express User's Guide*.	Expired and locked
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*.	Expired and locked

Table 3–2 (Cont.) Predefined Oracle Database Non-Administrative User Accounts

User Account	Description	Status After Installation
FLOWS_30000	The account that owns most of the database objects created during the installation of Oracle Database Application Express. These objects include tables, views, triggers, indexes, packages, and so on. See *Oracle Database Application Express User's Guide*.	Expired and locked
FLOWS_FILES	The account that owns the database objects created during the installation of Oracle Database Application Express related to modplsql document conveyance, for example, file uploads and downloads. These objects include tables, views, triggers, indexes, packages, and so on. See *Oracle Database Application Express User's Guide*.	Expired and locked
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 Developer's Guide*.	Expired and locked
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.	Expired and locked
SPATIAL_CSW_ADMIN_USR	The Catalog Services for the Web (CSW) account. It is used by Oracle Spatial CSW Cache Manager to load all record-type metadata and record instances from the database into the main memory for the record types that are cached. See *Oracle Spatial Developer's Guide*.	Expired and locked
SPATIAL_WFS_ADMIN_USR	The Web Feature Service (WFS) account. It is used by Oracle Spatial WFS Cache Manager to load all feature type metadata and feature instances from the database into main memory for the feature types that are cached. See *Oracle Spatial Developer's Guide*.	Expired and locked
XS$NULL	An internal account that represents the absence of a user in a session. Because XS$NULL is not a user, this account can only be accessed by the Oracle Database instance. XS$NULL has no privileges and no one can authenticate as XS$NULL, nor can authentication credentials ever be assigned to XS$NULL.	Expired and locked

Predefined Sample Schema User Accounts

If you install the sample schemas, which you must do to complete the examples in this guide, Oracle Database creates a set of sample user accounts. 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, as described in

"Expiring and Locking Database Accounts" on page 3-7. For more information about the sample schema accounts, see *Oracle Database Sample Schemas*.

Table 3–3 lists the sample schema user accounts, which represent different divisions of a fictional company that manufactures various products.

Table 3–3 Default Sample Schema User Accounts

User Account	Description	Status After Installation
BI	The account that owns the BI (Business Intelligence) schema included in the Oracle Sample Schemas. See also *Oracle Warehouse Builder User's Guide*.	Expired and locked
HR	The account used to manage the HR (Human Resources) schema. This schema stores information about the employees and the facilities of the company.	Expired and locked
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.	Expired and locked
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.	Expired and locked
IX	The account used to manage the IX (Information Exchange) schema. This schema manages shipping through business-to-business (B2B) applications.	Expired and locked
SH	The account used to manage the SH (Sales) schema. This schema stores business statistics to facilitate business decisions.	Expired and locked

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.

Expiring and Locking Database Accounts

When you expire the password of a user, that password no longer exists. If you want to *unexpire* the password, you change the password of that account. Locking an account preserves the user password and other account information, but makes the account unavailable to anyone who tries to log in to the database using that account. Unlocking it makes the account available again.

Oracle Database 2 Day DBA explains how you can use Database Control to unlock database accounts. You also can use Database Control to expire or lock database accounts.

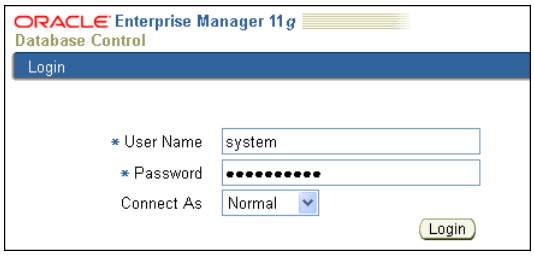
To expire and lock a database account:

1. Start Database Control.

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Log in with administrative privileges.

   For example:



The Database Home page appears.

3. Click **Server** to display the Server subpage.
4. In the Security section, click **Users**.

   The Users page lists the user accounts created for the current database instance. The Account Status column indicates whether an account is expired, locked, or open.
5. In the Select column, select the account you want to expire, and then click **Edit**.

   The Edit User page appears.
6. Do one of the following:
   - To expire a password, click **Expire Password now**.

     To unexpire the password, enter a new password in the **Enter Password** and **Confirm Password** fields. See "Requirements for Creating Passwords" on page 3-8 for password requirements.
   - To lock the account, select **Locked**.
7. Click **Apply**.

Requirements for Creating Passwords

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.

For greater security, follow these guidelines when you create passwords:

- Make the password between 10 and 30 characters and numbers.
- Use mixed case letters and special characters in the password. (See *Oracle Database Security Guide* for more information.)
- Use the database character set for the password's characters, which can include the underscore (_), dollar ($), and number sign (#) characters.
- Do not use an actual word for the entire password.

Oracle Database Security Guide describes more ways that you can further secure passwords.

See Also:

- "Finding and Changing Default Passwords" on page 3-9 for information about changing user passwords
- "Expiring and Locking Database Accounts" on page 3-7 for information about locking accounts and expiring passwords
- "Predefined User Accounts Provided by Oracle Database" on page 3-2 a description of the predefined user accounts that are created when you install Oracle Database
- *Oracle Database 2 Day DBA* for an introduction to password policies
- *Oracle Database Security Guide* for detailed information about managing passwords

Finding and Changing Default Passwords

When you install Oracle Database, the default database user accounts, including administrative accounts, are created without default passwords. Except for the administrative accounts whose passwords you create during installation (such as user SYS), the default user accounts arrive locked with their passwords expired. If you have upgraded from a previous release of Oracle Database, you may have database accounts that still have default passwords. These are default accounts that are created when you create a database, such as the HR, OE, and SCOTT accounts.

Security is most easily compromised when a default database user account still has a default password *after installation*. This is particularly true for the user account SCOTT, which is a well known account that may be vulnerable to intruders. Find accounts that use default passwords and then change their passwords.

To find and change default passwords:

1. Log into SQL*Plus with administrative privileges.

   ```
   sqlplus system
   Enter password: password
   ```

2. Select from the DBA_USERS_WITH_DEFPWD data dictionary view.

   ```
   SELECT * FROM DBA_USERS_WITH_DEFPWD;
   ```

   The DBA_USERS_WITH_DEFPWD lists the accounts that still have user default passwords. For example:

   ```
   USERNAME
   ------------
   SCOTT
   ```

3. Change the password for the accounts the DBA_USERS_WITH_DEFPWD data dictionary view lists.

   For example, to change the password for user SCOTT, enter the following:

   ```
   PASSWORD SCOTT
   Changing password for SCOTT
   New password: password
   Retype new password: password
   Password changed
   ```

Replace *password* with a password that is secure, according to the guidelines listed in "Requirements for Creating Passwords" on page 3-8. For greater security, do not reuse the same password that was used in previous releases of Oracle Database.

Alternatively, you can use the `ALTER USER` SQL statement to change the password:

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

You can use Database Control to change a user account passwords (not just the default user account passwords) if you have administrative privileges. Individual users can also use Database Control to change their own passwords.

To use Database Control to change the password of a database account:

1. Start Database Control.

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Enter an administrator user name and password (for example, `SYSTEM`), and then click **Login**.
3. Click **Server** to display the Server subpage.
4. In the Security section, click **Users**.

   The Users page lists the user accounts created for the current database instance. The Account Status column indicates whether an account is expired, locked, or open.

5. In the Select column, select the account you want to change, and then click **Edit**.

   The Edit User page appears.

6. Enter a new password in the **Enter Password** and **Confirm Password** fields.
7. Click **Apply**.

See Also:

- *Oracle Database Security Guide* for additional methods of configuring password protection
- "Predefined User Accounts Provided by Oracle Database" on page 3-2

Guideline for Handling the Default Administrative User Passwords

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 Database 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 create passwords for the `SYS` and `SYSTEM` accounts.

Do not use default passwords for any administrative accounts, including `SYSMAN` and `DBSNMP`. Oracle Database 11*g* Release 2 (11.2) and later does not install these accounts with default passwords, but if you have upgraded from an earlier release of Oracle Database, you may still have accounts that use default passwords. You should find and change these accounts by using the procedures in "Finding and Changing Default Passwords" on page 3-9.

At the end of database creation, Database Configuration Assistant displays a page that requires you to enter and confirm new passwords for the SYS and SYSTEM user accounts.

After installation, you can use Database Control to change the administrative user passwords. See "Finding and Changing Default Passwords" on page 3-9 for more information on changing a password.

Guideline for Enforcing 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. "Requirements for Creating Passwords" on page 3-8 provides guidelines for creating password policies. Table 3–4 on page 3-12 lists initialization parameters that you can set to enforce password management.

You can find information about user accounts by querying the DBA_USERS view. The DBA_USERS view provides useful information such as the user account status, whether the account is locked, and password versions. You can query DBA_USERS as follows:

```
sqlplus system
Enter password: password

SQL> SELECT * FROM DBA_USERS;
```

Oracle also recommends, if possible, using Oracle Advanced Security (an option to Oracle Database Enterprise Edition) with network authentication services (such as Kerberos), token cards, smart cards, or X.509 certificates. These services provide strong authentication of users, and provide better protection against unauthorized access to Oracle Database.

See Also:

- *Oracle Database Security Guide* for more information about password management
- *Oracle Database Security Guide* for additional views you can query to find information about users and profiles
- *Oracle Database Advanced Security Administrator's Guide* for more information about Oracle Database Advanced Security

Parameters Used to Secure User Accounts

Table 3–4 lists initialization and profile parameters that you can set to better secure user accounts.

Table 3–4 Initialization and Profile Parameters Used for User Account Security

Parameter	Default Setting	Description
SEC_CASE_SENSITIVE_LOGON	TRUE	Controls case sensitivity in passwords. TRUE enables case sensitivity; FALSE disables it.
SEC_MAX_FAILED_LOGIN_ATTEMPTS	No default setting	Sets the maximum number of times a user is allowed to fail when connecting to an Oracle Call Interface (OCI) application.
FAILED_LOGIN_ATTEMPTS	10	Sets the maximum times a user login is allowed to fail before locking the account. **Note:** You also can set limits on the number of times an unauthorized user (possibly an intruder) attempts to log in to Oracle Call Interface 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 his or her password before it expires.
PASSWORD_LIFE_TIME	180	Sets the number of days the user can use his or her 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.
PASSWORD_REUSE_MAX	UNLIMITED	Specifies the number of password changes required before the current password can be reused.
PASSWORD_REUSE_TIME	UNLIMITED	Specifies the number of days before which a password cannot be reused.

Note: You can use most of these parameters to create a user profile. See *Oracle Database Security Guide* for more information about user profile settings.

To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see *Oracle Database Reference* and*Oracle Database Administrator's Guide*.

4

Managing User Privileges

This chapter contains:

- About Privilege Management
- Guideline for Granting Privileges
- Guideline for Granting Roles to Users
- Guideline for Handling Privileges for the PUBLIC User Group
- Controlling Access to Applications with Secure Application Roles
- Initialization Parameters Used for Privilege Security

See Also:

- *Oracle Database Security Guide*
- *Oracle Label Security Administrator's Guide*

About Privilege Management

You can control user privileges in the following ways:

- **Granting and revoking individual privileges.** You can grant individual privileges, for example, the privilege to perform the `UPDATE` SQL statement, to individual users or to groups of users.
- **Creating a role and assigning privileges to it.** A role is a named group of related privileges that you grant, as a group, to users or other roles.
- **Creating a secure application role.** A secure application role enables you to define conditions that control when a database role can be enabled. For example, a secure application role can check the IP address associated with a user session before allowing the session to enable a database role.

Guideline for Granting Privileges

Because privileges are the rights to perform a specific action, such as updating or deleting a table, do not provide database users more privileges than are necessary. For an introduction to managing privileges, see "About User Privileges and Roles" in *Oracle Database 2 Day DBA*. *Oracle Database 2 Day DBA* also provides an example of how to grant a privilege.

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

For example, generally the CREATE ANY TABLE privilege is not granted to a user who does not have database administrator privileges.

Guideline for Granting Roles to Users

A role is a named group of related privileges that you grant, as a group, to users or other roles. To learn the fundamentals of managing roles, see "Administering Roles" in *Oracle Database 2 Day DBA*. In addition, see "Example: Creating a Role" in *Oracle Database 2 Day DBA*.

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, 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 required for the responsibility of a particular job. 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 restrictive role.

Do not grant powerful privileges, such as the CREATE DATABASE LINK privilege, to regular users such as user SCOTT. (Particularly do not grant *any* powerful privileges to SCOTT, because this is a well known default user account that may be vulnerable to intruders.) Instead, grant the privilege to a database role, and then grant this role to the users who must use the privilege. And remember to only grant the minimum privileges the user needs.

Guideline for Handling Privileges for the PUBLIC User Group

You should revoke unnecessary privileges and roles from the database server user group PUBLIC. PUBLIC acts as a default role granted to every user in an Oracle database. Any database user can exercise privileges that are granted to PUBLIC. These privileges include EXECUTE on various PL/SQL packages, potentially enabling someone with minimal privileges to access and execute functions that this user would not otherwise be permitted to access directly.

Controlling Access to Applications with Secure Application Roles

A secure application role is a role that can be enabled only by an authorized PL/SQL package. The PL/SQL package itself reflects the security policies necessary to control access to the application.

This section contains:

- About Secure Application Roles
- Tutorial: Creating a Secure Application Role

About Secure Application Roles

A secure application role is a role that can be enabled only by an authorized PL/SQL package. This package defines one or more security policies that control access to the application. Both the role and the package are typically created in the schema of the person who creates them, which is typically a security administrator. A security administrator is a database administrator who is responsible for maintaining the security of the database.

The advantage of using a secure application role is you can create additional layers of security for application access, in addition to the privileges that were granted to the role itself. Secure application roles strengthen security because passwords are not embedded in application source code or stored in a table. This way, the decisions the database makes are based on the implementation of your security policies. Because these definitions are stored in one place, the database, rather than in your applications, you modify this policy once instead of modifying the policy in each application. No matter how many users connect to the database, the result is always the same, because the policy is bound to the role.

A secure application role has the following components:

- **The secure application role itself.** You create the role using the `CREATE ROLE` statement with the `IDENTIFIED USING` clause to associate it with the PL/SQL package. Then, you grant the role the privileges you typically grant a role.
- **A PL/SQL package, procedure, or function associated with the secure application role.** The PL/SQL package sets a condition that either grants the role or denies the role to the person trying to log in to the database. You must create the PL/SQL package, procedure, or function using invoker's rights, not definer's rights. An invoker's right procedure executes with the privileges of the current user, that is, the user who invokes the procedure. This user must be granted the `EXECUTE` privilege for the underlying objects that the PL/SQL package accesses. The invoker's right procedures are not bound to a particular schema. They can be run by a variety of users and enable multiple users to manage their own data by using centralized application logic. To create the invoker's rights package, use the `AUTHID CURRENT_USER` clause in the declaration section of the procedure code.

  The PL/SQL package also must contain a `SET ROLE` statement or `DBMS_SESSION.SET_ROLE` call to enable (or disable) the role for the user.

  After you create the PL/SQL package, you must grant the appropriate users the `EXECUTE` privilege on the package.
- **A way to execute the PL/SQL package when the user logs on.** To execute the PL/SQL package, you must call it directly from the application before the user tries to use the privileges the role grants. You cannot use a logon trigger to execute the PL/SQL package automatically when the user logs on.

When a user logs in to the application, the policies in the package perform the checks as needed. If the user passes the checks, then the role is granted, which enables access to the application. If the user fails the checks, then the user is prevented from accessing the application.

Tutorial: Creating a Secure Application Role

This tutorial shows how two employees, Matthew Weiss and Winston Taylor, try to gain information from the `OE.ORDERS` table. Access rights to this table are defined in the `EMPLOYEE_ROLE` secure application role. Matthew is Winston's manager, so Matthew, as opposed to Winston, will be able to access the information in `OE.ORDERS`.

In this tutorial:

- Step 1: Create a Security Administrator Account
- Step 2: Create User Accounts for This Tutorial
- Step 3: Create the Secure Application Role
- Step 4: Create a Lookup View
- Step 5: Create the PL/SQL Procedure to Set the Secure Application Role
- Step 6: Grant the EXECUTE Privilege for the Procedure to Matthew and Winston
- Step 7: Test the EMPLOYEE_ROLE Secure Application Role
- Step 8: Optionally, Remove the Components for This Tutorial

Step 1: Create a Security Administrator Account

For greater security, you should apply separation of duty concepts when you assign responsibilities to the system administrators on your staff. For the tutorials used in this guide, you will create and use a security administrator account called `sec_admin`.

To create the sec_admin security administrator account:

1. Start Database Control.

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Enter an administrator user name (for example, `SYSTEM`) and password, and then click **Login**.

   For the `SYSTEM` user, connect as `Normal`.

   The Database Home page appears.

3. Click **Server** to display the Server subpage.

4. Under Security, select **Users**.

   The Users page appears.

5. Click **Create**.

   The Create User page appears.

6. Enter the following information:

   - **Name**: `sec_admin`
   - **Profile**: `Default`
   - **Authentication**: `Password`
   - **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in "Requirements for Creating Passwords" on page 3-8.
   - **Default Tablespace**: `SYSTEM`
   - **Temporary Tablespace**: `TEMP`
   - **Status**: `UNLOCKED`

7. Click **Roles** to display the Edit User: SEC_ADMIN page.

8. Click **Edit List**.

   The Modify Roles page appears.

9. In the Available Roles list, select the **SELECT_CATALOG_ROLE** role and then then click **Move** to move it to the Selected Roles list. Then click **OK** to return to the Edit User page.
10. Click **System Privileges** to display the System Privileges subpage.
11. Click **Edit List**.

    The Modify System Privileges page appears.
12. In the Available System Privileges list, select the following privileges and then click **Move** to move each one to the Selected System Privileges list. (Hold down the Control key to select multiple privileges.)
    - `CREATE PROCEDURE`
    - `CREATE ROLE`
    - `CREATE SESSION`
    - `SELECT ANY DICTIONARY`
13. Click **OK**.

    The Create User page appears.
14. Under Admin Option, do not select the boxes.
15. Click **OK**.

    The Users page appears. User `sec_admin` is listed in the UserName list.

Step 2: Create User Accounts for This Tutorial

Matthew and Winston both are sample employees in the `HR.EMPLOYEES` schema, which provides columns for the manager ID and e-mail address of the employees, among other information. You must create user accounts for these two employees so that they can later test the secure application role.

To create the user accounts:

1. In the Users page, click **Create**.

   The Create User page appears.
2. Enter the following information:
   - **Name**: `mweiss` (to create the user account for Matthew Weiss)
   - **Profile**: `DEFAULT`
   - **Authentication**: `Password`
   - **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in "Requirements for Creating Passwords" on page 3-8.
   - **Default Tablespace**: `USERS`
   - **Temporary Tablespace**: `TEMP`
   - **Status**: `Unlocked`
3. Click **System Privileges** to display the System Privileges subpage.
4. Click **Edit List**.

   The Modify System Privileges page appears.
5. In the Available System Privileges lists, select the `CREATE SESSION` privilege, and then click **Move** to move it to the Selected System Privileges list.

6. Click **OK**.

   The Create User page appears, with `CREATE SESSION` listed as the system privilege for user `mweiss`.

7. Ensure that the Admin Option for `CREATE SESSION` is not selected, and then click **OK**.

   The Users page appears.

8. Select the option for user **MWEISS** from the list of users, and then from the **Actions** list, select **Create Like**. Then, click **Go**.

9. In the Create User page, enter the following information to create the user account for Winston, which will be almost identical to the user account for Matthew:

   - **Name**: `wtaylor`
   - **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in "Requirements for Creating Passwords" on page 3-8.

   You do not need to specify the default and temporary tablespaces, or the `CREATE SESSION` system privilege, for user `wtaylor` because they are already specified.

10. Click **OK**.

    You do not need to grant `wtaylor` the `CREATE SESSION` privilege, because the **Create Like** action has done of this for you.

11. Log out of Database Control.

Now both Matthew Weiss and Winston Taylor have user accounts that have identical privileges.

Step 3: Create the Secure Application Role

Now, you are ready to create the `employee_role` secure application role. To do so, you must log on as the security administrator `sec_admin`. "Step 1: Create a Security Administrator Account" on page 4-4 explains how to create the `sec_admin` account.

To create the secure application role:

1. Start SQL*Plus and log on as the security administrator `sec_admin`.

   ```
   SQLPLUS sec_admin
   Enter password: password
   ```

   SQL*Plus starts, connects to the default database, and then displays a prompt.

   ```
   SQL>
   ```

   For detailed information about starting SQL*Plus, see *Oracle Database 2 Day DBA*.

2. Create the following secure application role:

   ```
   CREATE ROLE employee_role IDENTIFIED USING sec_roles;
   ```

   The `IDENTIFIED USING` clause sets the role to be enabled (or disabled) only within the associated PL/SQL package, in this case, `sec_roles`. At this stage, the `sec_roles` PL/SQL package does not need to exist.

3. Connect as user `OE`.

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

If you receive an error message saying that OE is locked, then you can unlock the OE account and reset its password by entering the following statements. For greater security, do not reuse the same password that was used in previous releases of Oracle Database. Enter any password that is secure, according to the password guidelines described in "Requirements for Creating Passwords" on page 3-8.

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

PASSWORD OE  -- First, change the OE account password.
Changing password for OE
New password: password
Retype new password: password
Password changed.

ALTER USER OE ACCOUNT UNLOCK; -- Next, unlock the OE account.
```

Another way to unlock a user account and create a new password is to use the following syntax:

```
ALTER USER account_name ACCOUNT UNLOCK IDENTIFIED BY new_password:
```

Now you can connect as user OE.

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

4. Enter the following statement to grant the EMPLOYEE_ROLE role SELECT privileges on the OE.ORDERS table.

```
GRANT SELECT ON OE.ORDERS TO employee_role;
```

Do not grant the role directly to the user. The PL/SQL package will do that for you, assuming the user passes its security policies.

Step 4: Create a Lookup View

You are almost ready to create the procedure that determines who is granted the employee_role role. The procedure will grant the employee_role only to managers who report to Steven King, whose employee ID is 100. This information is located in the HR.EMPLOYEES table. However, you should not use that table in this procedure, because it contains sensitive data such as salary information, and for it to be used, everyone will need access to it. In most real world cases, you create a lookup view that contains only the information that you need. (You could create a lookup table, but a view will reflect the most recent data.) For this tutorial, you create your own lookup view that only contains the employee names, employee IDs, and their manager IDs.

To create the HR.HR_VERIFY lookup view:

1. In SQL*Plus, connect as user HR.

```
CONNECT HR
Enter password: password
```

If you receive an error message saying that HR is locked, then you can unlock the account and reset its password by entering the following statements. For greater security, do not reuse the same password that was used in previous releases of Oracle Database. Enter any password that is secure, according to the password guidelines described in "Requirements for Creating Passwords" on page 3-8.

```
CONNECT SYSTEM
Enter password: password

PASSWORD HR
Changing password for HR
New password: password
Retype new password: password
Password changed.

ALTER USER HR ACCOUNT UNLOCK;

CONNECT HR
Enter password: password
```

2. Enter the following `CREATE VIEW` SQL statement to create the lookup view:

```
CREATE VIEW hr_verify AS
SELECT EMPLOYEE_ID, FIRST_NAME, LAST_NAME, EMAIL, MANAGER_ID
FROM EMPLOYEES;
```

3. Grant the `EXECUTE` privilege for this view to `mweiss`, `wtaylor`, and `sec_admin` by entering the following SQL statements:

```
GRANT SELECT ON hr.hr_verify TO mweiss;
GRANT SELECT ON hr.hr_verify TO wtaylor;
GRANT SELECT ON hr.hr_verify TO sec_admin;
```

Step 5: Create the PL/SQL Procedure to Set the Secure Application Role

Now, you are ready to create the secure application role procedure. In most cases, you create a package to hold the procedure, but because this is a simple tutorial that requires only one secure application role test (as defined in the procedure), you will create a procedure by itself. If you want to have a series of procedures to test for the role, create them in a package.

A PL/SQL package defines a simple, clear interface to a set of related procedures and types that can be accessed by SQL statements. Packages also make code more reusable and easier to maintain. The advantage here for secure application roles is that you can create a group of security policies that used together present a solid security strategy designed 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.

To create the secure application role procedure:

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

```
CONNECT sec_admin
Enter password: password
```

2. Enter the following `CREATE PROCEDURE` statement to create the secure application role procedure. (You can copy and paste this text by positioning the cursor at the start of `CREATE OR REPLACE` in the first line.)

```
1   CREATE OR REPLACE PROCEDURE sec_roles AUTHID CURRENT_USER
2    AS
3   v_user varchar2(50);
4   v_manager_id number :=1;
5    BEGIN
6     v_user := lower((sys_context ('userenv','session_user')));
7     SELECT manager_id
8        INTO v_manager_id FROM hr.hr_verify WHERE lower(email)=v_user;
9      IF v_manager_id = 100
10      THEN
11       EXECUTE IMMEDIATE 'SET ROLE employee_role';
12      ELSE NULL;
13     END IF;
14    EXCEPTION
15     WHEN NO_DATA_FOUND THEN v_manager_id:=0;
16     DBMS_OUTPUT.PUT_LINE(v_manager_id);
17  END;
18  /
```

In this example:

- **Line 1**: Appends the `AUTHID CURRENT_USER` clause to the `CREATE PROCEDURE` statement, which creates the procedure using invoker's rights. The `AUTHID CURRENT_USER` clause creates the package using invoker's rights, using the privileges of the current user.

  You *must* create the package using invoker's rights for the package to work. Invoker's rights allow the user to have `EXECUTE` privileges on all objects that the package accesses.

  Roles that are enabled inside an invoker's right procedure remain in effect even after the procedure exits, but after the user exits the session, he or she no longer has the privileges associated with the secure application role. In this case, you can have a dedicated procedure that enables the role for the rest of the session.

  Because users cannot change the security domain inside definer's rights procedures, secure application roles can only be enabled inside invoker's rights procedures.

  See "About Secure Application Roles" on page 4-3 for information about the importance of creating the procedure using invoker's rights.

- **Line3**: Declares the `v_user` variable, which will store the user session information.

- **Line 4**: Declares the `v_manager_id` variable, which will store the manager's ID of the `v_user` user.

- **Line 6**: Retrieves the user session information for the user logging on, in this case, Matthew or Winston. To retrieve user session information, use the `SYS_CONTEXT` SQL function with the `USERENV` namespace attributes ('`userenv`', *`session_attribute`*), and writes this information to the `v_user` variable.

  The information returned by this function indicates the way in which the user was authenticated, the IP address of the client, and whether the user connected through a proxy. See *Oracle Database SQL Language Reference* for more information about `SYS_CONTEXT`.

- **Lines 7–8**: Get the manager's ID of the current user. The `SELECT` statement copies the manager ID into the `v_manager_id` variable, and then checking the `HR.HR_VERIFY` view for the manager ID of the current user. This example

uses the employees' e-mail addresses because they are the same as their user names.

- **Lines 9–13**: Use an `IF` condition to test whether the user should be granted the `sec_roles` role. In this case, the test condition is whether the user reports to Matthew's manager, Steven King, whose employee number is 100. If the user reports to King, as Matthew does, then the secure application role is granted to the user. Otherwise, the role is not granted.

  The result is that the secure application role will grant Matthew Weiss the role because he is a direct report of Steven King, but will deny the role to Winston, because he is not a direct report of Steven King.

- **Lines 10–12**: Within the `IF` condition, the `THEN` condition grants the role by executing immediately the `SET ROLE` statement. Otherwise, the `ELSE` condition denies the grant.

- **Lines 14–15**: Use an `EXCEPTION` statement to set `v_manager_id` to `0` if no data is found.

- **Line 16**: Copies the manager's ID, which is now `0`, into a buffer so that it is readily available.

  Tip: If you have problems creating or running PL/SQL code, check the Oracle Database trace files. The `USER_DUMP_DEST` initialization parameter specifies the current location of the trace files. You can find the value of this parameter by issuing `SHOW PARAMETER USER_DUMP_DEST` in SQL*Plus. See *Oracle Database Performance Tuning Guide* for more information about trace files.

Step 6: Grant the EXECUTE Privilege for the Procedure to Matthew and Winston

At this stage, Matthew and Winston can try to access the `OE.ORDERS` table, but they are denied access. The next step is to grant them the `EXECUTE` privilege on the `sec_roles` procedure, so that the `sec_roles` procedure can execute, and then grant or deny access, when they try to select from the `OE.ORDERS` table.

To grant EXECUTE privileges for the sec_roles procedure:

- In SQL*Plus, as user `sec_admin`, enter the following `GRANT` SQL statements:

```
GRANT EXECUTE ON sec_admin.sec_roles TO mweiss;
GRANT EXECUTE ON sec_admin.sec_roles TO wtaylor;
```

Step 7: Test the EMPLOYEE_ROLE Secure Application Role

You are ready to test the `employee_role` secure application role by logging on as Matthew and Winston and trying to access the `OE.ORDERS` table. When Matthew and Winston log on, and before they issue a `SELECT` statement on the `OE.ORDERS` table, the `sec_roles` procedure must be executed for the role verification to take place.

To test the employee_role secure application role, as user MWEISS:

1. Connect as user `mweiss`.

```
CONNECT mweiss
Enter password: password
```

2. Enter the following SQL statement to run the `sec_roles` procedure:

```
EXEC sec_admin.sec_roles;
```

This statement executes the `sec_roles` procedure for the current session. (In a real world scenario, this statement would be automatically run when the user logs in to the application.)

3. Perform the following `SELECT` statement on the `OE.ORDERS` table:

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

Matthew has access to the `OE.ORDERS` table:

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

Now, Winston will try to access the secure application.

To test the employee_role secure application role as user WTAYLOR:

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

```
CONNECT wtaylor
Enter password: password
```

2. Enter the following SQL statement to run the `sec_roles` procedure:

```
EXEC sec_admin.sec_roles;
```

This statement executes the `sec_roles` procedure for the current session.

3. Perform the following `SELECT` statement on the `OE.ORDERS` table:

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

Because Winston does not report directly to Steven King, he does not have access to the `OE.ORDERS` table. He will never learn the true number of orders in the `ORDERS` table, at least not by performing a `SELECT` statement on it.

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

Step 8: Optionally, Remove the Components for This Tutorial

Remove the components that you created for this tutorial.

To remove the components:

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

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

2. Enter the following `DROP` statements:

```
DROP USER mweiss;
DROP USER wtaylor;
```

Do not drop user `sec_admin`. You will need this user for other tutorials in this guide.

3. In SQL*Plus, connect as user `sec_admin`.

```
CONNECT sec_admin
Enter password: password
```

4. Enter the following DROP SQL statements:

```
DROP ROLE employee_role;
DROP PROCEDURE sec_roles;
```

5. Connect as user HR, and then drop the HR_VERIFY view.

```
CONNECT HR
Enter password: password
DROP VIEW hr_verify;
```

6. Exit SQL*Plus.

```
EXIT
```

Initialization Parameters Used for Privilege Security

Table 4–1 lists initialization parameters that you can use to secure user privileges.

Table 4–1 Initialization Parameters Used for Privilege Security

Initialization Parameter	Default Setting	Description
O7_DICTIONARY_ACCESSIBILITY	FALSE	Controls restrictions on SYSTEM privileges. See "Enabling Data Dictionary Protection" on page 2-3 for more information about this parameter.
OS_ROLES	FALSE	Determines whether the operating system identifies and manages the roles of each user.
MAX_ENABLED_ROLES	30	Specifies the maximum number of database roles that users can enable, including roles contained within other roles.
REMOTE_OS_ROLES	FALSE	Specifies whether operating system roles are allowed for remote clients. The default value, FALSE, causes Oracle to identify and manage roles for remote clients.
SQL92_SECURITY	FALSE	Specifies whether users must be granted the SELECT object privilege to execute UPDATE or DELETE statements.

To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see *Oracle Database Reference* and *Oracle Database Administrator's Guide*.

5

Securing the Network

This chapter contains:

- About Securing the Network
- Securing the Client Connection on the Network
- Protecting Data on the Network by Using Network Encryption
- Initialization Parameters Used for Network Security

About Securing the Network

You can configure the client connection to your Oracle Database installation by following the procedures in "Configuring the Network Environment" in *Oracle Database 2 Day DBA* and the Oracle Database Installation Guide for your platform. This chapter explains how you can encrypt data as it travels through the network, and also provides guidelines that you can follow to secure the network connections for Oracle Database.

Securing the Client Connection on the Network

This section describes how you can tightens security for the client connection to ensure thorough protection. Encrypting network traffic is essential for securing communications with the database.

These guidelines are as follows:

- Guidelines for Securing Client Connections
- Guidelines for Securing the Network Connection

Guidelines for Securing 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, compromised operating systems or applications, and falsified or stolen client system identities. Nevertheless, the following guidelines improve the security of client connections:

1. **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 remote user from impersonating another operating system 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 a nonsecure connection and use it for account access. (To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6.) 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 users connecting to an Oracle database.

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 apply its standard authentication processes.

2. **Configure the connection to use encryption.**

   Oracle network encryption makes eavesdropping difficult. To learn how to configure encryption, see *Oracle Database Advanced Security Administrator's Guide*.

3. **Set up strong authentication.**

   See *Oracle Database Advanced Security Administrator's Guide* for more information about using Kerberos and public key infrastructure (PKI).

Guidelines for Securing the Network Connection

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 an introduction to using Oracle Net Manager, see *Oracle Database 2 Day DBA*. See also *Oracle Database Net Services Administrator's Guide*.

The following practices improve network security:

1. **Monitor listener activity.**

   You can monitor listener activity by using Oracle Enterprise Manager Database Control. In the Database Control home page, under General, click the link for your listener. The Listener page appears. This page provides detailed information, such as the category of alert generated, alert messages, when the alert was triggered, and so on. This page provides other information, such as performance statistics for the listener.

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 modify this line in the `listener.ora` file:

      ```
      ADMIN_RESTRICTIONS_LISTENER=ON
      ```

   b. Use `RELOAD` to reload the configuration.

   c. Use SSL 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 = shobeen.us.example.com)
  (PORT = 8281)))
```

To administer the listener remotely, 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 = shobeen.us.example.com)
      (PORT = 8281))
    )
  )
```

For more information about the parameters in `listener.ora`, see *Oracle Database Net Services Reference*.

3. **Do not set the listener password.**

   Ensure that the password has not been set in the `listener.ora` 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.

   Remember that the listener password has been deprecated in this release, and will not be supported in the next release of Oracle Database.

4. **When a host has multiple IP addresses associated with multiple NIC cards, configure the listener to the specific IP address.**

   This enables the listener to monitor all the IP addresses. You can restrict the listener to monitor a specific IP address. Oracle recommends that you specify the specific IP addresses on these types of computers, rather than enabling the listener to monitor all IP addresses. Restricting the listener to specific IP addresses helps to prevent an intruder from stealing a TCP end point from 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*.

6. **Use encryption to secure the data in flight.**

   See "Protecting Data on the Network by Using Network Encryption" on page 5-5 to learn about how to protect Oracle data over the network. *Oracle Database Advanced Security Administrator's Guide* describes network encryption in detail.

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 (formerly known as Net8 and 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 Oracle Connection Manager to multiplex multiple-client, network sessions through a single network connection to the database. It can filter using the 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 using the 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.

   You can use Oracle Net Manager to configure these parameters. See *Oracle Database Net Services Administrator's Guide* for more information.

10. **Encrypt network traffic.**

    If possible, use Oracle Advanced Security to encrypt network traffic among clients, databases, and application servers. For an introduction to Oracle network encryption, see "Protecting Data on the Network by Using Network Encryption" on page 5-5. For detailed information about network encryption, see *Oracle Database Advanced Security Administrator's Guide*.

11. **Secure the host operating system (the system on which Oracle Database resides).**

    Secure the host operating system by disabling all unnecessary operating system services. Both UNIX and Windows platforms 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.

Protecting Data on the Network by Using Network Encryption

In addition to protecting information by encrypting it at the database level, you must protect it as it travels across the network.

This section contains:

- About Network Encryption
- Configuring Network Encryption

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* for detailed information about network encryption

About Network Encryption

Network encryption refers to encrypting data as it travels across the network between the client and server. The reason you should encrypt data at the network level, and not just the database level, is because data can be exposed on the network level. For example, an intruder can use a network packet sniffer to capture information as it travels on the network, and then spool it to a file for malicious use. Encrypting data on the network prevents this sort of activity.

To encrypt data on the network, you need the following components:

- **An encryption seed.** The encryption seed is a random string of up to 256 characters. It generates the cryptographic keys that encrypts data as it travels across the network.
- **An encryption algorithm.** You can specify any of the supported algorithm types: AES, RC4, DES, or 3DES.
- **Whether the settings apply to a client or server.** You must configure the server and each client to which it connects.
- **How the client or server should processes the encrypted data.** The settings you select (you have four options) must complement both server and client.
- **A mechanism for configuring the encryption.** You can use Oracle Net Manager to configure the encryption. Alternatively, you can edit the `sqlnet.ora` configuration file. Both Oracle Net Manager and the `sqlnet.ora` file are available in a default Oracle Database installation.

Configuring Network Encryption

You can configure network encryption by using either Oracle Net Manager or by editing the `sqlnet.ora` file. This guide explains how to use Oracle Net Manager to configure network encryption.

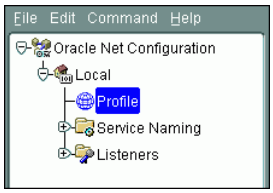
To configure network encryption:

1. On the server computer, start Oracle Net Manager.
   - **UNIX:** From `$ORACLE_HOME/bin`, enter the following at the command line:

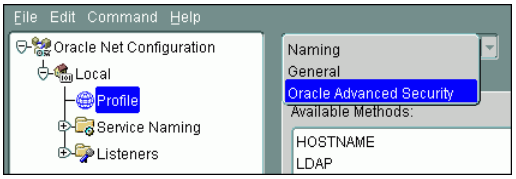
     ```
     netmgr
     ```

   - **Windows:** From the **Start** menu, click **All Programs**. Then, click **Oracle - *HOME_NAME***, **Configuration and Migration Tools**, and then **Net Manager**

2. From the Oracle Net Configuration navigation tree, expand **Local**, and then select **Profile**.

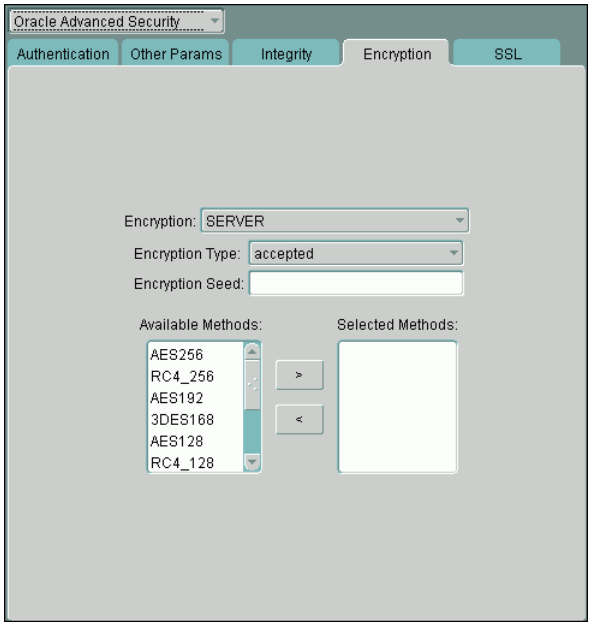


3. From the list, select **Oracle Advanced Security**.



4. Under Oracle Advanced Security, select the **Encryption** tab.

   The Encryption settings pane appears.



5. Enter the following settings:

   - **Encryption**: From the list, select **SERVER** to configure the network encryption for the server. (For the client computer, you select **CLIENT**.)
   - **Encryption Type**: Select from the following values to specify the actions of the server (or client) when negotiating encryption and integrity:
     - **accepted**: Service will be active if the other side of the connection specifies either required or requested, and there is a compatible algorithm available on the receiving database; it will otherwise be inactive.
     - **rejected**: Service must not be active, and the connection will fail if the other side requires any of the methods in this list.

  - **requested**: Service will be active if the other side of the connection specifies either accepted, required, or requested, and there is a compatible algorithm available on the other side. Otherwise, the service is inactive.
  - **required**: Service must be active, and the connection will fail if the other side specifies rejected, or if there is no compatible algorithm on the other side.

- **Encryption Seed**: Enter a random string of up to 256 characters. Oracle Database uses the encryption seed to generate cryptographic keys. This is required when either encryption or integrity is enabled.

  If you choose to use special characters such as a comma [,] or a right parenthesis [)] as a part of the **Encryption Seed** parameter, enclose the value within single quotation marks.

- **Available Methods**: Select one or more of the following algorithms, and use the move button (>) to move them to the Selected Methods list. The order in which they appear in the Selected Methods list determines the preferred order for negotiation. That is, the first algorithm listed is selected first, and so on.
  - **AES256**: Advanced Encryption Standard (AES). AES was approved by the National Institute of Standards and Technology (NIST) to replace Data Encryption Standard (DES). AES256 enables you to encrypt a block size of 256 bits.
  - **RC4_256**: Rivest Cipher 4 (RC4), which is the most commonly used stream cipher that protects protocols such as Secure Sockets Layer (SSL). RC4_256 enables you to encrypt up to 256 bits of data.
  - **AES192**: Enables you to use AES to encrypt a block size of 192 bits.
  - **3DES168**: Triple Data Encryption Standard (TDES) with a three-key option. 3DES168 enables you to encrypt up to 168 bits of data.
  - **AES128**: Enables you to use AES to encrypt a block size of 128 bits.
  - **RC4_128**: Enables you to use RC4 to encrypt up to 128 bits of data.
  - **3DES112**: Enables you to use Triple DES with a two-key (112 bit) option.
  - **DES**: Data Encryption Standard (DES) 56-bit key. Note that National Institute of Standards and Technology (NIST) no longer recommends DES.
  - **RC4_40**: Enables you to use RC4 to encrypt up to 40 bits of data. (Not recommended.)
  - **DES40**: Enables you to use DES to encrypt up to 40 bits of data. (Not recommended.)

6. From the **File** menu, select **Save Network Configuration**, and then select **Exit** to exit Oracle Net Manager.
7. Repeat these steps for each client computer that connects to the server.

See Also:

- *Oracle Database Net Services Reference* for information about editing the `sqlnet.ora` file parameters to configure network encryption
- *Oracle Database Advanced Security Administrator's Guide* for more information about network data encryption

Initialization Parameters Used for Network Security

Table 5–1 lists initialization parameters that you can set to better secure user accounts.

Table 5–1 Initialization Parameters Used for Network Security

Initialization Parameter	Default Setting	Description
`OS_AUTHENT_PREFIX`	`OPS$`	Specifies a prefix that Oracle Database uses to identify users attempting to connect to the database. Oracle Database concatenates the value of this parameter to the beginning of the user operating system account name and password. When a user attempts a connection request, Oracle Database compares the prefixed username with user names in the database.
`REMOTE_LISTENER`	No default setting	Specifies a network name that resolves to an address or address list of Oracle Net remote listeners (that is, listeners that are not running on the same computer as this instance). The address or address list is specified in the `tnsnames.ora` file or other address repository as configured for your system.
`REMOTE_OS_AUTHENT`	`FALSE`	Specifies whether remote clients will be authenticated with the value of the `OS_AUTHENT_PREFIX` parameter.
`REMOTE_OS_ROLES`	`FALSE`	Specifies whether operating system roles are allowed for remote clients. The default value, `FALSE`, causes Oracle Database to identify and manage roles for remote clients.

To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see *Oracle Database Reference* and*Oracle Database Administrator's Guide*.

6

Securing Data

This chapter contains:

- About Securing Data
- Encrypting Data Transparently with Transparent Data Encryption
- Choosing Between Oracle Virtual Private Database and Oracle Label Security
- Controlling Data Access with Oracle Virtual Private Database
- Enforcing Row-Level Security with Oracle Label Security
- Controlling Administrator Access with Oracle Database Vault

About Securing Data

Oracle Database provides many ways to secure data. This chapter describes the following methods that you can use to secure data on your site:

- **Transparent data encryption.** Transparent data encryption encrypts data in one or more database table columns, or it can encrypt an entire tablespace. Transparent data encryption is the quickest and easiest way to encrypt data. Transparent data encryption supports the Advanced Encryption Standard (AES) and Triple Data Encryption Standard (3DES) algorithms.

  You can also encrypt data on the network. "Protecting Data on the Network by Using Network Encryption" on page 5-5 explains how.

- **Oracle Virtual Private Database (VPD).** This feature restricts row and column level data access by creating a policy that enforces a `WHERE` clause for all SQL statements that query the database. You create and manage the VPD policy at the database table or view level, which means that you do not modify the applications that access the database.

- **Oracle Label Security (OLS).** This feature secures your database tables at the row level, and assigns these rows different levels of security based on security labels. You then create a security authorization for users based on the OLS labels.

- **Oracle Database Vault.** This feature enables you to restrict administrator access to your databases, enforce separation of duty, and control who, when, where and how applications, databases, and data are accessed.

Encrypting Data Transparently with Transparent Data Encryption

Transparent data encryption enables you to quickly encrypt one or more table columns or a tablespace. It is easy to implement and has many advantages over other types of database encryption.

This section contains:

- About Encrypting Sensitive Data
- When Should You Encrypt Data?
- How Transparent Data Encryption Works
- Configuring Data to Use Transparent Data Encryption
- Checking Existing Encrypted Data

About Encrypting Sensitive Data

Encrypted data can only be read by its recipient. You use encryption to protect data in a potentially unprotected environment, such as data you have placed on backup media that is sent to an offsite storage location.

The encryption data includes the following components:

- **An algorithm to encrypt the data.** The encryption algorithm is used by Oracle databases to encrypt data. Oracle Database supports several industry-standard encryption and hashing algorithms, including the Advanced Encryption Standard (AES) encryption algorithm, which has been approved by the National Institute of Standards and Technology (NIST).
- **A key to encrypt and decrypt data.** When you encrypt data, Oracle Database uses the key and clear text data as input into the encryption algorithm. Conversely, when you decrypt data, the key is used as input into the algorithm to reverse the process and retrieve the clear text data. Oracle Database uses a symmetric encryption key to perform this task, in which the same key is used to both encrypt and decrypt the data. The encryption key is stored in the data dictionary, but encrypted with another master key.

As mentioned earlier, you can encrypt individual table columns or an entire tablespace. Be careful that you do not mix the two. For example, suppose you encrypt a table column and then encrypt its surrounding tablespace. This double encryption can cause performance problems. In addition, column encryption has limitations in data type support, and only supports B-tree indexes for equality searches. To check the current encrypted settings, you can query the `V$ENCRYPTED_TABLESPACES` data dictionary view for tablespaces, and the `DBA_ENCRYPTED_COLUMNS` view for encrypted columns.

When Should You Encrypt Data?

In most cases, you encrypt sensitive data on your site to meet a regulatory compliance. For example, sensitive data such as credit card numbers, Social Security numbers, or patient health information must be encrypted.

Historically, users have wanted to encrypt data because they want to restrict data access from their database administrators. However, this problem is more of an access control problem, not an encryption problem. You can address this problem by using Oracle Database Vault to control the access to your application data from database administrators.

In most cases, you encrypt sensitive data such as credit cards, and Social Security numbers to prevent access when backup tapes or disk drives are lost or stolen. In recent years industry regulations such as the Payment Card Industry (PCI) Data Security Standard and the Healthcare Insurance Portability and Accountability Act (HIPAA) have become a driving factor behind increased usage of encryption for protecting credit card and health care information, respectively.

> **See Also:** *Oracle Database Security Guide* for common misconceptions about encrypting stored data

How Transparent Data Encryption Works

Transparent data encryption enables you to encrypt individual table columns or an entire tablespace. When a user inserts data into an encrypted column, transparent data encryption automatically encrypts the data. When users select the column, the data is automatically decrypted.

To encrypt data by using transparent data encryption, you create the following components:

- **A wallet to store the master encryption key.** The wallet is an operating system file located outside the database. The database uses the wallet to store the master encryption key. To create the wallet, you can use Enterprise Manager or the `ALTER SYSTEM` command. The wallet is encrypted using a password as the encryption key. You create the password when you create the wallet. Access to the contents (or master key) of the wallet is thus restricted to only those who know the password. After the wallet is created, you must open the wallet using the password so that the database can access the master encryption key.
- **A location for the wallet.** You can specify the wallet location in the `sqlnet.ora` file.

Afterward, when a user enters data, Oracle Database performs the following steps:

1. Retrieves the master key from the wallet.
2. Decrypts the encryption key using the master key.
3. Uses the encryption key to encrypt the data the user entered.
4. Stores the data in encrypted format in the database.

If the user is selecting data, the process is similar: Oracle Database decrypts the data and then displays it in clear text format.

Transparent data encryption has the following advantages:

- As a security administrator, you can be sure that sensitive data is safe if the storage media or data file is stolen or lost.
- Implementing transparent data encryption helps you address security-related regulatory compliance issues.
- Data from tables is transparently decrypted for the database user. You do not need to create triggers or views to decrypt data.
- Database users need not be aware of the fact that the data they are accessing is stored in encrypted form. Data is transparently decrypted for the database users and does not require any action on their part.
- Applications need not be modified to handle encrypted data. Data encryption and decryption is managed by the database.

Transparent data encryption has a minimal impact on performance. Transparent data encryption column encryption affects performance only when data is retrieved from or inserted into an encrypted column. There is no impact on performance for operations involving unencrypted columns, even if these columns are in a table containing encrypted columns. However, be aware that encrypted data needs more storage space than clear text data. On average, encrypting a single column requires between 32 and 48 bytes of additional storage for each row. Transparent tablespace encryption provides even better performance because Oracle Database performs the encryption and decryption at the I/O block layer. Once blocks are decrypted, they are cached in Oracle Database memory for optimal performance.

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* for detailed information about using transparent data encryption

Configuring Data to Use Transparent Data Encryption

To start using transparent data encryption, you must create a wallet and set a master key. The wallet can be the default database wallet shared with other Oracle Database components, or a separate wallet specifically used by transparent data encryption. Oracle recommends that you use a separate wallet to store the master encryption key. This wallet will be used for all data that is being encrypted through transparent data encryption.

You follow these steps to configure table columns to use transparent data encryption:

- Step 1: Configure the Wallet Location
- Step 2: Create the Wallet
- Step 3: Open (or Close) the Wallet
- Step 4: Encrypt (or Decrypt) Data

> **See Also:** *Oracle Database Advanced Security Administrator's Guide* for detailed information about using tablespace encryption

Step 1: Configure the Wallet Location

You designate the directory location for the wallet in the `sqlnet.ora` file. You perform this step once.

To configure the wallet location:

1. Create a directory in the `$ORACLE_HOME` directory in which to store the wallet.

   For example, create a directory called `ORA_WALLETS` in the `C:\oracle\product\11.2.0\db_1` directory.

2. Create a backup copy of the `sqlnet.ora` file, which by default is located in the `$ORACLE_HOME/network/admin` directory.

3. At the end of the `sqlnet.ora` file, add code similar to the following, where `ORA_WALLETS` is the name of the directory where you plan to store the wallet:

```
ENCRYPTION_WALLET_LOCATION=
 (SOURCE=
  (METHOD=file)
   (METHOD_DATA=
    (DIRECTORY=C:\oracle\product\11.2.0\db_1\ORA_WALLETS)))
```

4. Save and close the `sqlnet.ora` file.

5. If the compatibility of the database is set to a release earlier than Oracle Database Release 10.2, then restart the database.

   a. Log in to SQL*Plus and then check the database compatibility.

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

   SQL*Plus starts, connects to the default database, and then displays a `SQL>` prompt.

   For detailed information about starting SQL*Plus, see *Oracle Database 2 Day DBA*.

   b. Check the value of the `COMPATIBLE` parameter.

   ```
   SHOW PARAMETER COMPATIBLE

   NAME                         TYPE          VALUE
   ------------------------- -------------- --------------------
   compatible                   string        11.2.0
   ```

   c. If the value is greater than 10.2, then you can go to Step 2: Create the Wallet. If the value is less than 10.2, then restart the database as follows.

   ```
   SHUTDOWN IMMEDIATE
   STARTUP
   ```

Step 2: Create the Wallet

To create the wallet, use the `ALTER SYSTEM` SQL statement. By default, the Oracle wallet stores a history of retired master keys, which enables you to change them and still be able to decrypt data that was encrypted under an old master key. A case-sensitive wallet password unknown to the database administrator provides separation of duty: The database administrator might be able to restart the database, but the wallet is closed and must be manually opened by a security administrator before the database can encrypt or decrypt the data.

To create the wallet:

1. In SQL*Plus, connect as a user with administrative privileges, such as `SYS`, or as a security administrator.

   For example:

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

2. Enter the following `ALTER SYSTEM` statement, where password is the *`password`* you want to use to protect the Oracle wallet:

   ```
   ALTER SYSTEM SET ENCRYPTION KEY IDENTIFIED BY "password";
   ```

   Enclose the password in double quotation marks. As with other passwords that you create in Oracle Database, the password does not appear in clear text or in any dynamic views or logs.

   This statement generates the wallet with a new encryption key and sets it as the current transparent data encryption master key. If you plan to use public key infrastructure (PKI) to configure the master encryption key, then specify a certificate ID, which is an optional string that contains the unique identifier of a certificate stored in the Oracle wallet. Use the following syntax:

   ```
   ALTER SYSTEM SET ENCRYPTION KEY certificate_ID IDENTIFIED BY "password";
   ```

Step 3: Open (or Close) the Wallet

Immediately after you create the wallet key, the wallet is open, and you are ready to start encrypting data. However, if you have restarted the database after you created the wallet, you must manually open the wallet before you can use transparent data encryption.

To open the wallet:

- In SQL*Plus, enter the following `ALTER SYSTEM` statement, where password is the *`password`* you use to protect the wallet:

```
ALTER SYSTEM SET ENCRYPTION WALLET OPEN IDENTIFIED BY "password";
```

You must inclose the password in quotation marks.

In most cases, leave the wallet open unless you have a reason for closing it. You can close the wallet to disable access to the master key and prevent access to the encrypted columns. The wallet must be open for transparent data encryption to work. To reopen the wallet, use the `ALTER SYSTEM SET WALLET OPEN IDENTIFIED BY` *`password`* statement.

To close the wallet:

- In SQL*Plus, enter the following statement, and ensure that you enclose the password in quotation marks:

```
ALTER SYSTEM SET ENCRYPTION WALLET CLOSE IDENTIFIED BY "password";
```

Step 4: Encrypt (or Decrypt) Data

After you have created a directory location for the wallet in the `sqlnet.ora` file and created the wallet itself, you are ready to encrypt either individual table columns or an entire tablespace.

This section contains the following topics:

- Encrypting Individual Table Columns
- Encrypting a Tablespace

Encrypting Individual Table Columns The decisions that you make when you identify columns to be encrypted are determined by governmental security regulations, such as California Senate Bill 1386, or by industry standards such as the Payment Card Industry (PCI) Data Security Standard. Credit card numbers, Social Security numbers, and other personally identifiable information (PII) fall under this category. Another need for encryption is defined by your own internal security policies — trade secrets, research results, or employee salaries and bonuses. See "When Should You Encrypt Data?" on page 6-2 for guidelines about when and when not to encrypt data.

Follow these guidelines when you select columns to encrypt:

- **Check the data types of the columns you plan to encrypt.** Transparent data encryption supports the following data types:

`BINARY_FLOAT`	`NUMBER`
`BINARY_DOUBLE`	`NVARCHAR2`
`CHAR`	`RAW`
`DATE`	`TIMESTAMP`
`NCHAR`	`VARCHAR2`

- **Ensure that the columns you select are not part of a foreign key.** With transparent data encryption, each table has its own encryption key, which is stored in the database data dictionary and encrypted with the external master key. Encrypted columns cannot be used as foreign keys.

To encrypt a column in a table:

1. Ensure that you have created and opened a wallet key.

   "Step 2: Create the Wallet" on page 6-5 explains how to create a wallet key. To open an existing wallet key, see "Step 3: Open (or Close) the Wallet" on page 6-6.

2. Start Database Control.

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

3. Enter an administrator user name (for example, `SYSTEM`, or the name of a security administrator) and password, and then click **Login**.

   The Database Home page appears.

4. Click **Schema** to display the Schema subpage.

5. Under Database Objects, select **Tables**.

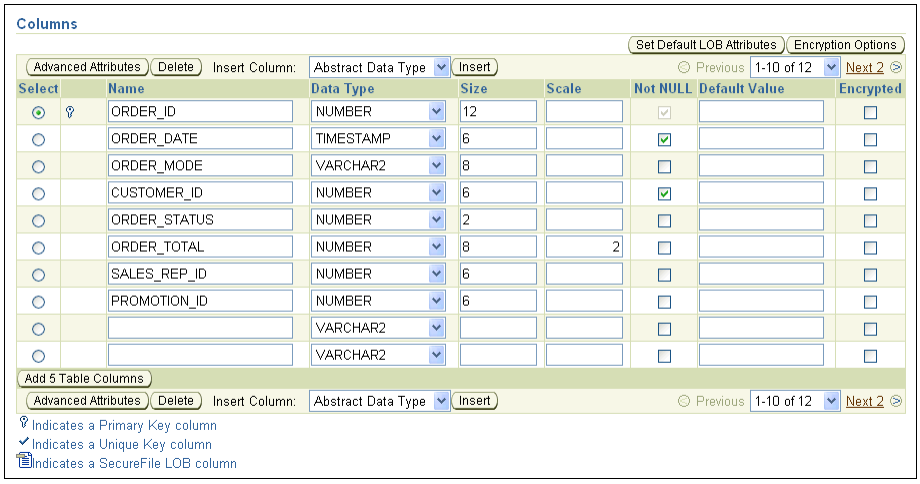
   The Tables page appears.

6. Do one of the following:

   - To create a new table, click **Create**, and then answer the questions in the subsequent page to start creating the table.
   - To modify an existing table, search for the table name by entering its schema name into the **Schema** field and the table name in the **Object Name** field. (You can use the percent sign (%) wildcard character to search for a group of tables, for example `O%` to find all tables beginning with the letter O.) When the table is listed in the Tables page, select the table, and then click **Edit**.

   In the Create Table or Edit Table page, you can set its encryption options.

   For example, to encrypt columns in the `OE.ORDERS` table, the Edit Table page appears as follows:



7. In the Create Table (or Edit Table) page, do the following:

   a. Select the column that you want to encrypt.

Do not select columns that are part of a foreign key constraint (primary or unique key columns). You cannot encrypt these columns. These columns are indicated with a key or check mark icon to the left of their names.

b. Click **Encryption Options** to display the Encryption Options for the Table page.

c. From the Encryption Algorithm list, select from the following options:

   – **AES192**: Sets the key length to 192 bits. AES is the abbreviation for Advanced Encryption Standard.

   – **3DES168**: Sets the key length to 168 bits. 3DES is the abbreviation for Triple Data Encryption Standard.

   – **AES128**: Sets the key length to 128 bits. This option is the default.

   – **AES256**: Sets the key length to 256 bits.

d. Under Key Generation, select either **Generate Key Randomly** or **Specify Key**. If you select **Specify Key**, enter characters for the seed values in the **Enter Key** and **Confirm Key** fields.

   The **Generate Key Randomly** setting enables salt. **Salt** is a way to strengthen the security of encrypted data. It is a random string added to the data before it is encrypted, causing the same text to appear different when encrypted. Salt removes one method attackers use to steal data, namely, matching patterns of encrypted text.

e. Click **Continue** to return to the Create Table (or Edit Table) page.

f. Enable encryption for the column by selecting its box under **Encrypted**.

8. Click **Continue**.

   The Create Table (or Edit Table) page appears.

While a table is being updated, read access is still possible. Afterward, existing and future data in the column is encrypted when it is written to the database file, and it is decrypted when an authorized user selects it. If data manipulation language (DML) statements are needed, you can use online redefinition statements.

Encrypting a Tablespace You can encrypt a new tablespace while you are creating it, but you cannot encrypt an existing tablespace. As a workaround, you can use the `CREATE TABLE AS SELECT`, `ALTER TABLE MOVE`, or use Oracle Data Pump import to get data from an existing tablespace into an encrypted tablespace. For details about creating a tablespace, see *Oracle Database 2 Day DBA*.

To encrypt a tablespace:

1. Ensure that you have created and opened a wallet key.

   "Step 2: Create the Wallet" on page 6-5 explains how to create a wallet key. To open an existing wallet key, see "Step 3: Open (or Close) the Wallet" on page 6-6.

2. Start Database Control.

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

3. Enter an administrator user name (for example, `SYSTEM`, or the name of a security administrator) and password, and then click **Login**.

   The Database Home page appears.

4. Click **Server** to display the Server subpage.

5. Under Storage, click **Tablespaces**.

   The Tablespaces page appears.

6. Click **Create**, and then answer the questions in the subsequent page to start creating the tablespace and its required data file.

7. In the Create Tablespace page, do the following:

   a. Under Type, select the **Encryption** box, under Permanent.

   b. Select **Encryption** options to display the Encryption Options page.

   c. From the Encryption Algorithm list, select from the following options:

      – **AES192**: Sets the key length to 192 bits. AES is the abbreviation for Advanced Encryption Standard.

      – **3DES168**: Sets the key length to 168 bits. 3DES is the abbreviation for Triple Data Encryption Standard.

      – **AES128**: Sets the key length to 128 bits. This option is the default.

      – **AES256**: Sets the key length to 256 bits.

      See "Available Methods" under Step 5 in "Configuring Network Encryption" on page 5-5 for more information about these encryption algorithms.

   d. Click **Continue**.

      The Create Tablespace page appears.

8. Click **OK**.

   The new tablespace appears in the list of existing tablespaces. Remember that you cannot encrypt an existing tablespace.

   See Also:

   - "Checking Encrypted Tablespaces in the Current Database Instance" on page 6-11 to query the database for existing encrypted tablespaces
   - *Oracle Database Advanced Security Administrator's Guide* for detailed information about tablespace encryption
   - *Oracle Database SQL Language Reference* for more information about the `CREATE TABLESPACE` statement

Checking Existing Encrypted Data

You can query the database for the data that you have encrypted. You can check for individually encrypted columns, all tables in the current database instance that have encrypted columns, or all tablespaces that are encrypted.

This section contains:

- Checking Whether a Wallet Is Open or Closed
- Checking Encrypted Columns of an Individual Table
- Checking All Encrypted Table Columns in the Current Database Instance
- Checking Encrypted Tablespaces in the Current Database Instance

Checking Whether a Wallet Is Open or Closed

You can find out if a wallet is open or closed by running the `V$ENCRYPTION_WALLET` view.

To check whether a wallet is open or closed:

- In SQL*Plus, query the `V$ENCRYPTION_VIEW` view as follows:

  ```
  SELECT * FROM V$ENCRYPTION_WALLET;
  ```

  The wallet status appears, similar to the following:

  ```
  WRL_TYPE  WRL_PARAMETER                            STATUS
  --------  ---------------------------------------  -------
  file      C:\oracle\product\11.2.0\db_1\wallets    OPEN
  ```

Checking Encrypted Columns of an Individual Table

You use the `DESC` (for `DESCRIBE`) statement in SQL*Plus to check the encrypted columns in a database table.

To check the encrypted columns of an individual table:

- In SQL*Plus, run the `DESC` statement using the following syntax.

  ```
  DESC tablename;
  ```

  For example:

  ```
  DESC OE.ORDER_ITEMS;
  ```

  A description of the table schema appears. For example:

  ```
  Name                                      Null?     Type
  ----------------------------------------  --------  --------------------------
  ORDER_ID                                  NOT NULL  NUMBER(12)
  LINE_ITEM_ID                              NOT NULL  NUMBER(3)
  PRODUCT_ID                                NOT NULL  NUMBER(6)
  UNIT_PRICE                                          NUMBER(8,2)
  QUANTITY                                            NUMBER(8) ENCRYPT
  ```

Checking All Encrypted Table Columns in the Current Database Instance

To check all encrypted table columns, you use the `DBA_ENCRYPTED_COLUMNS` view.

To check all encrypted table columns in the current database instance:

- In SQL*Plus, select from the `DBA_ENCRYPTED_COLUMNS` view:

  For example:

  ```
  SELECT * FROM DBA_ENCRYPTED_COLUMNS;
  ```

  This `SELECT` statement lists all tables and column in the database that contain columns encrypted using Oracle Transparent Data Encryption. For example:

  ```
  OWNER        TABLE_NAME    COLUMN_NAME    ENCRYPTION_ALG     SALT
  -----------  ----------    -----------    ----------------   ----
  OE           CUSTOMERS     INCOME_LEVEL   AES 128 bits key   YES
  OE           UNIT_PRICE    ORADER_ITEMS   AES 128 bits key   YES
  HR           EMPLOYEES     SALARY         AES 192 bits key   YES
  ```

  See Also: *Oracle Database Reference* for more information about the `DBA_ENCRYPTED_COLUMNS` view

Checking Encrypted Tablespaces in the Current Database Instance

Table 6–1 lists data dictionary views that you can use to check encrypted tablespaces.

Table 6–1 Data Dictionary Views for Encrypted Tablespaces

Data Dictionary View	Description

DBA_TABLESPACES	Describes all tablespaces in the database. For example, find out if the tablespace has been encrypted, enter the following:
	`SELECT TABLESPACE_NAME, ENCRYPTED FROM DBA_TABLESPACES`
	
	`TABLESPACE_NAME ENC`
	`---------- ----`
	`SYSTEM NO`
	`SYSAUX NO`
	`UNCOTBS1 NO`
	`TEMP NO`
	`USERS NO`
	`EXAMPLE NO`
	`SECURESPACE YES`

USER_TABLESPACES	Describes the tablespaces accessible to the current user. It has the same columns as DBA_TABLESPACES, except for the PLUGGED_IN column.

V$ENCRYPTED_TABLESPACES	Displays information about the tablespaces that are encrypted. For example:
	`SELECT * FROM V$ENCRYPTED_TABLESPACES;`
	
	`TS# ENCRYPTIONALG ENCRYPTEDTS`
	`--- --- ---`
	`6 AES128 YES`
	
	The list includes the tablespace number, its encryption algorithm, and whether its encryption is enabled or disabled.
	If you want to find the name of the tablespace, use the following join operation:
	`SELECT NAME, ENCRYPTIONALG ENCRYPTEDTS`
	`FROM V$ENCRYPTED_TABLESPACES, V$TABLESPACE`
	`WHERE V$ENCRYPTED_TABLESPACES.TS# = V$TABLESPACE.TS#;`

See Also: *Oracle Database Reference* for more information about data dictionary views

Choosing Between Oracle Virtual Private Database and Oracle Label Security

Both Oracle Virtual Private Database (VPD) and Oracle Label Security (OLS) enable you to restrict the data that different users can see in database tables. But when should you use Virtual Private Database and when should you use Oracle Label Security? Virtual Private Database is effective when there is existing data you can use to determine the access requirements. For example, you can configure a sales representative to see only the rows and columns in a customer order entry table for orders he or she handles. Oracle Label Security is useful if you have no natural data (such as user accounts or employee IDs) that can be used to indicate a table's access

requirements. To determine this type of user access, you assign different levels of sensitivity to the table rows.

In some cases, Oracle Virtual Private Database and Oracle Label Security can complement each other. The following Oracle Technology Network hands-on tutorial demonstrates how a Virtual Private Database policy can compare an Oracle Label Security user clearance with a minimum clearance. When the user clearance dominates the threshold, the `Salary` column is not hidden.

http://www.oracle.com/technology/deploy/security/database-security/label-security/ols_cs1.html

Table 6–2 compares the features of Oracle Virtual Private Database with Oracle Label Security.

Table 6–2 Comparing Oracle Virtual Private Database with Oracle Label Security

Feature	VPD	OLS
Provides row-level security	Yes	Yes
Provides column-level security (column masking)	Yes	No
Binds a user-defined PL/SQL package to a table, view, or synonym	Yes	No[1]
Modifies SQL by dynamically adding a `WHERE` clause returned from the PL/SQL procedures	Yes	No
Restricts database operations by privileged users[2]	No	No
Controls access to a set of rows based on the sensitivity label of the row and the security level of the user	No	Yes
Adds a column (optionally hidden) designed to store sensitivity labels for rows in the protected table[3]	No	Yes
Provides a user account to manage its administration	No[4]	Yes[5]
Provides pre-defined PL/SQL packages for row-level security	No	Yes
Is provided in the default installation of Oracle Database	Yes	No
Is provided as an additional option to Oracle Database and must be licensed	No	Yes

[1] Oracle Label Security uses predefined PL/SQL packages, not user-created packages, to attach security policies to tables.

[2] If you must restrict privileged user access, consider using Oracle Database Vault.

[3] Usually, this column is hidden to achieve transparency and not break applications that are not designed to show an additional column.

[4] Oracle Virtual Private Database does not provide a user account, but you can create a user account that is solely responsible for managing Virtual Private Database policies.

[5] The `LBACSYS` account manages Oracle Label Security policies. This provides an additional layer of security in that one specific user account is responsible for these policies, which reduces the risk of another user tampering with the policies.

Controlling Data Access with Oracle Virtual Private Database

Oracle Virtual Private Database (VPD) enables you to dynamically add a `WHERE` clause in any SQL statement that a user executes. The `WHERE` clause filters the data the user is allowed to access, based on the identity of a user.

This section contains:

- About Oracle Virtual Private Database
- Tutorial: Creating an Oracle Virtual Private Database Policy

> **See Also:** *Oracle Database Security Guide* for detailed information about how Oracle Virtual Private Database works

About Oracle Virtual Private Database

Oracle Virtual Private Database (VPD) provides row-level security at the database table or view level. You can extend it to provide column-level security as well. Essentially, Virtual Private Database inserts an additional `WHERE` clause to any SQL statement that is used on any table or view to which a Virtual Private Database security policy has been applied. (A security policy is a function that allows or prevents access to data.) The `WHERE` clause allows only users whose identity passes the security policy, and hence, have access to the data that you want to protect.

An Oracle Virtual Private Database policy has the following components, which are typically created in the schema of the security administrator:

- **A PL/SQL function to append the dynamic WHERE clause to SQL statements that affect the Virtual Private Database tables.** For example, a PL/SQL function translates the following `SELECT` statement:

  ```
  SELECT * FROM ORDERS;
  ```

  to the following:

  ```
  SELECT * FROM ORDERS
    WHERE SALES_REP_ID = 159;
  ```

  In this example, the user can only view orders by Sales Representative 159. The PL/SQL function used to generate this `WHERE` clause is as follows:

  ```
  1    CREATE OR REPLACE FUNCTION auth_orders(
  2     schema_var IN VARCHAR2,
  3     table_var  IN VARCHAR2
  4    )
  5    RETURN VARCHAR2
  6    IS
  7     return_val VARCHAR2 (400);
  8    BEGIN
  9     return_val := 'SALES_REP_ID = 159';
  10    RETURN return_val;
  11   END auth_orders;
  12   /
  ```

  In this example:

  - **Lines 2–3:** Create parameters to store the schema name, `OE`, and table name, `ORDERS`. (The second parameter, `table_var`, for the table, can also be used for views and synonyms.) Always create these two parameters in this order: create the parameter for the schema first, followed by the parameter for the table, view, or synonym object. Note that the function itself does not specify the `OE` schema or its `ORDERS` table. The Virtual Private Database policy you create uses these parameters to specify the `OE.ORDERS` table.
  - **Line 5:** Returns the string that will be used for the `WHERE` predicate clause.
  - **Lines 6–10:** Encompass the creation of the `WHERE SALES_REP_ID = 159` predicate.

  You can design the `WHERE` clause to filter the user information based on the session information of that user, such as the user ID. To do so, you create an application context. Application contexts can be used to authenticate both

database and nondatabase users. An application context is a name-value pair. For example:

```
SELECT * FROM oe.orders
 WHERE sales_rep_id = SYS_CONTEXT('userenv','session_user');
```

In this example, the `WHERE` clause uses the `SYS_CONTEXT` PL/SQL function to retrieve the user session ID (`session_user`) designated by the `userenv` context. See *Oracle Database Security Guide* for detailed information about application contexts.

- **A way to attach the policy the package.** Use the `DBMS_RLS.ADD_POLICY` function to attach the policy to the package. Before you can use the `DBMS_RLS` PL/SQL package, you must be granted `EXECUTE` privileges on it. User `SYS` owns the `DBMS_RLS` package.

The advantages of enforcing row-level security at the database level rather than at the application program level are enormous. Because the security policy is implemented in the database itself, where the data to be protected is, this data is less likely to be vulnerable to attacks by different data access methods. This layer of security is present and enforced no matter how users (or intruders) try to access the data it protects. The maintenance overhead is low because you maintain the policy in one place, the database, rather than having to maintain it in the applications that connect to this database. The policies that you create provide a great deal of flexibility because you can write them for specific DML operations.

Tutorial: Creating an Oracle Virtual Private Database Policy

The `ORDERS` table in the Order Entry database, `OE`, contains the following information:

```
Name                                    Null?    Type
----------------------------------------- -------- ----------------------------------
ORDER_ID                                NOTNULL  NUMBER(12)
ORDER_DATE                              NOTNULL  TIMESTAMP(6) WITH LOCAL TIME ZONE
ORDER_MODE                                       VARCHAR2(8)
CUSTOMER_ID                             NOTNULL  NUMBER(6)
ORDER_STATUS                                     NUMBER(2)
ORDER_TOTAL                                      NUMBER(8,2)
SALES_REP_ID                                     NUMBER(6)
PROMOTION_ID                                     NUMBER(6)
```

Suppose you want to limit access to this table based on the person who is querying the table. For example, a sales representative should only see the orders that he or she have created, but other employees should not. In this tutorial, you create a sales representative user account and an account for a finance manager. Then, you create an Oracle Virtual Private Database policy that will limit the data access to these users based on their roles.

The Virtual Private Database policy that you will create is associated with a PL/SQL function. Because VPD policies are controlled by PL/SQL functions or procedures, you can design the policy to restrict access in many different ways. For this tutorial, the function you create will restrict access by the employees based on to whom they report. The function will restrict the customer access based on the customer's ID.

You may want to store VPD policies in a database account separate from the database administrator and from application accounts. In this tutorial, you will use the `sec_admin` account, which was created in "Tutorial: Creating a Secure Application Role" on page 4-3, to create the VPD policy. This provides better security by separating the VPD policy from the applications tables.

To restrict access based on the sensitivity of row data, you can use Oracle Label Security (OLS). OLS lets you categorize data into different levels of security, with each level determining who can access the data in that row. This way, the data access restriction is focused on the data itself, rather than on user privileges. See "Enforcing Row-Level Security with Oracle Label Security" on page 6-21 for more information.

In this tutorial:

- Step 1: If Necessary, Create the Security Administrator Account
- Step 2: Update the Security Administrator Account
- Step 3: Create User Accounts for This Tutorial
- Step 4: Create the F_POLICY_ORDERS Policy Function
- Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy
- Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy
- Step 7: Optionally, Remove the Components for This Tutorial

Step 1: If Necessary, Create the Security Administrator Account

In "Tutorial: Creating a Secure Application Role" on page 4-3, you created a security administrator account called `sec_admin` for that tutorial. You can use that account for this tutorial. If you have not yet created this account, follow the steps in "Step 1: Create a Security Administrator Account" on page 4-4 to create `sec_admin`.

Step 2: Update the Security Administrator Account

The `sec_admin` account user must have privileges to use the `DBMS_RLS` packages. User `SYS` owns this package, so you must log on as `SYS` to grant these package privileges to `sec_admin`. The user `sec_admin` also must have `SELECT` privileges on the `CUSTOMERS` table in the `OE` schema and the `EMPLOYEES` table in the `HR` schema.

To grant sec_admin privileges to use the DBMS_RLS package:

1. Start Database Control.

   See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.

2. Log in as user `SYS` and connect with the `SYSDBA` privilege:

   - **User Name**: `SYS`
   - **Password**: Enter the password for `SYS`.
   - **Connect As**: `SYSDBA`

3. Click **Server** to display the Server subpage.
4. Under Security, select **Users**.

   The Users Page appears.

5. Select the **SEC_ADMIN** user, and in the View User page, click **Edit**.

   The Edit User page appears.

6. Click **Object Privileges** to display the Object Privileges page.
7. From the **Select Object Type** list, select **Package**, and then click **Add**.

   The Add Package Object Privileges page appears.

8. Under Select Package Objects, enter `SYS.DBMS_RLS` so that `sec_admin` will have access to the `DBMS_RLS` package.
9. Under Available Privileges, select **EXECUTE**, and then click **Move** to move it to the Selected Privileges list.
10. Click **OK**.

    The Edit User page appears.
11. From the **Select Object Type** list, select **Table**, and then click **Add**.

    The Add Table Object Privileges page appears.
12. In the **Select Table Objects** field, enter `HR.EMPLOYEES` so that `sec_admin` will have access to the `HR.EMPLOYEES` table.
13. Under Available Privileges, select **SELECT**, and then click **Move** to move it to the Selected Privileges list.
14. Click **OK**.

    The Edit User page appears. It shows that user sec_admin has object privileges for the `EMPLOYEES` table and `DBMS_RLS` PL/SQL package. Ensure that you do not select the grant option for either of these objects.
15. Click **Apply**.

    All the changes you have made, in this case, the addition of the two object privileges, are applied to the `sec_admin` user account.

Step 3: Create User Accounts for This Tutorial

You are ready to create accounts for the employees who must access the `OE.ORDERS` table.

To create the employee user accounts:

1. In Database Control, click **Users** in the **Database Instance** link to return to the Users page.

   The Users page appears.
2. Click **Create**.

   The Create User page appears.
3. Enter the following information:
   - **Name**: `LDORAN` (to create the user account Louise Doran)
   - **Profile**: `DEFAULT`
   - **Authentication**: `Password`
   - **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in "Requirements for Creating Passwords" on page 3-8.
   - **Default Tablespace**: `USERS`
   - **Temporary Tablespace**: `TEMP`
   - **Status**: `Unlocked`
4. Select the **Object Privileges** tab.
5. From the **Select Object Type** list, select **Table**, and then click **Add**.

   The Add Table Object Privileges page appears.

6. In the **Select Table Objects** field, enter the following text:

   ```
   OE.ORDERS
   ```

   Do not include spaces in this text.

7. In the Available Privileges list, select **SELECT**, and then click **Move** to move it to the Selected Privileges list. Click **OK**.

   The Create User page appears, with `SELECT` privileges for `OE.ORDERS` listed.

8. Click **OK**.

   The Users page appears, with user `ldoran` is listed in the UserName column.

9. Select the selection button for user **LDORAN**, and from the **Actions** list, select **Create Like**. Then, click **Go**.

   The Create User page appears.

10. Enter the following information:

    - **Name**: `LPOPP` (to create the user account for Finance Manager Luis Popp.)
    - **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in "Requirements for Creating Passwords" on page 3-8.

11. Click **OK**.

Both employee accounts have been created, and they have identical privileges. If you check the privileges for user `LPOPP`, you will see that they are identical to those of user `LDORAN`'s. At this stage, if either of these users performs a `SELECT` statement on the `OE.ORDERS` table, he or she will be able to see all of its data.

Step 4: Create the F_POLICY_ORDERS Policy Function

The `f_policy_orders` policy is a PL/SQL function that defines the policy used to filter users who query the `ORDERS` table. To filter the users, the policy function uses the `SYS_CONTEXT` PL/SQL function to retrieve session information about users who are logging in to the database.

To create the application context and its package:

1. In Database Control, click **Logout** and then **Login**.
2. Log in as user `sec_admin`.
3. Click **Schema** to display the Schema subpage.
4. Under Programs, select **Functions**.

   The Functions page appears.

5. Ensure that the **Object Type** menu is set to **Function**, and then click **Create**.

   The Create Function page appears.

6. Enter the following information:

   - **Name**: `F_POLICY_ORDERS`
   - **Schema**: `SEC_ADMIN`
   - **Source**: Delete the empty function code that has been provided, and then enter the following code (but not the line numbers on the left side of the code) to create a function that checks whether the user who has logged on is a sales representative. You can copy and paste this text by positioning the cursor at the start of `(schema in varchar2)` in the first line.

The `f_policy_orders` function accomplishes this by using the `SYS_CONTEXT` PL/SQL function to get the session information of the user, and then it compares this information with the job ID of that user in the `HR.EMPLOYEES` table, for which `sec_admin` has `SELECT` privileges.

```
1     (schema in varchar2,
2     tab in varchar2)
3     return varchar2
4     as
5      v_job_id   varchar2(20);
6      v_user     varchar2(100);
7      predicate  varchar2(400);
8
9     begin
10     v_job_id  := null;
11     v_user    := null;
12     predicate := '1=2';
13
14    v_user := lower(sys_context('userenv','session_user'));
15
16     select lower(job_id) into v_job_id from hr.employees
17       where lower(email) = v_user;
18
19     if  v_job_id='sa_rep' then
20        predicate := '1=1';
21     else
22        null;
23     end if;
24
25     return predicate;
26
27     exception
28      when no_data_found then
29       null;
30    end;
```

In this example:

- **Lines 1–2:** Define parameters for the schema (`schema`) and table (`tab`) that must be protected. Notice that the function does not mention the `OE.ORDERS` table. The `ACCESSCONTROL_ORDERS` policy that you create in Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy uses these parameters to specify the `OE` schema and `ORDERS` table. Ensure that you create the `schema` parameter first, followed by the `tab` parameter.
- **Line 3:** Returns the string that will be used for the `WHERE` predicate clause. Always use `VARCHAR2` as the data type for this return value.
- **Lines 4–7:** Define variables to store the job ID, user name of the user who has logged on, and predicate values.
- **Lines 9–25:** Encompass the creation of the `WHERE` predicate, starting the with the `BEGIN` clause at **Line 9**.
- **Lines 10–12:** Sets the `v_job_id` and `v_user` variables to null, and the `predicate` variable to 1=2, that is, to a false value. At this stage, no `WHERE` predicate can be generated until these variables pass the tests starting with **Line 16**.

- **Line 14:** Uses the `SYS_CONTEXT` function to retrieve the session information of the user and write it to the `v_user` variable.
- **Lines 16–23:** Checks if the user is a sales representative by comparing the job ID with the user who has logged on. If the job ID of the user who has logged on is `sa_rep` (sales representative), then the `predicate` variable is set to `1=1`. In other words, the user, by being a sales representative, has passed the test.
- **Line 25:** Returns the `WHERE` predicate, which translates to `WHERE` *`role_of_user_logging_on`* `IS "sa_rep"`. Oracle Database appends this `WHERE` predicate onto any `SELECT` statement that users `LDORAN` and `LPOPP` issue on the `OE.ORDERS` table.
- **Lines 27–29:** Provide an `EXCEPTION` clause for cases where a user without the correct privileges has logged on.

7. Click **OK**.

Step 5: Create the ACCESSCONTROL_ORDERS Virtual Private Database Policy

Now that you have created the Virtual Private Database policy function, you can create the Virtual Private Database policy, `accesscontrol_orders`, and then attach it to the `ORDERS` table. To increase performance, add the `CONTEXT_SENSITIVE` parameter to the policy, so that Oracle Database only executes the `f_policy_orders` function when the content of the application context changes, in this case, when a new user logs on. Oracle Database only activates the policy when a user performs a SQL `SELECT` statement on the `ORDERS` table. Hence, the user cannot run the `INSERT`, `UPDATE`, and `DELETE` statements, because the policy does not allow him or her to do so.

To create the ACCESSCONTROL_ORDERS Virtual Private Database policy:

1. In Database Control, click the **Database Instance** link to display the Database Home page.
2. Click **Server** to display the Server subpage.
3. In the Security section, click **Virtual Private Database**.

   The Virtual Private Database Policies page appears.
4. Click **Create**.

   The Create Policy page appears, with the Policy subpage displaying.
5. Under General, enter the following:
   - **Policy Name**: `ACCESSCONTROL_ORDERS`
   - **Object Name**: `OE.ORDERS`
   - **Policy Type**: Select `CONTEXT_SENSITIVE`.

     This type reevaluates the policy function at statement run-time if it detects context changes since the last use of the cursor. For session pooling, where multiple clients share a database session, the middle tier must reset the context during client switches. Note that Oracle Database does not cache the value that the function returns for this policy type; it always runs the policy function during statement parsing. The `CONTEXT_SENSITIVE` policy type applies to only one object.

     To enable the Policy Type, select the **Enabled** box.
6. Under Policy Function, enter the following:

- **Policy Function**: Enter the name of the function that generates a predicate for the policy, in this case, `SEC_ADMIN.F_POLICY_ORDERS`.
- **Long Predicate**: Do not select this box.

  Typically, you select this box to return a predicate with a length of up to 32K bytes. By not selecting this box, Oracle Database limits the predicate to 4000 bytes.

7. Under Enforcement, select the **SELECT** option but deselect the remaining options that already may be selected.
8. Do not select any options under Security Relevant Columns.
9. Click **OK**.

   The Virtual Private Database Policies page appears, with the `ACCESSCONTROL_ORDERS` policy listed in the list of policies.

Step 6: Test the ACCESSCONTROL_ORDERS Virtual Private Database Policy

At this stage, you are ready to test the `accesscontrol_orders` policy by logging on as each user and attempting to select data from the `ORDERS` table.

To test the ACCESSCONTROL_ORDERS policy:

1. Start SQL*Plus.

   From a command prompt, enter the following command to start SQL*Plus, and log in as Sales Representative Louise Doran, whose user name is `ldoran`:

   ```
   sqlplus ldoran
   Enter password: password
   ```

   SQL*Plus starts, connects to the default database, and then displays a prompt.

   For detailed information about starting SQL*Plus, see *Oracle Database 2 Day DBA*.

2. Enter the following `SELECT` statement:

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

   The following results should appear for Louise. As you can see, Louise is able to access all the orders in the `OE.ORDERS` table.

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

3. Connect as Finance Manager Luis Popp.

   ```
   CONNECT lpopp
   Enter password: password
   ```

4. Enter the following `SELECT` statement:

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

   The following results should appear, because Mr. Popp, who is not a sales representative, does not have access to the data in the `OE.ORDERS` table. Because Mr. Popp does not have access, Oracle Database only allows him access to 0 rows.

   ```
   COUNT(*)
   --------
          0
   ```

5. Exit SQL*Plus:

```
EXIT
```

Step 7: Optionally, Remove the Components for This Tutorial

After completing this tutorial, you can remove the data structures that you used if you no longer need them.

To remove the data structures created by sec_admin:

1. In Database Control, log in as user `sec_admin`.
2. Click **Server** to display the Server subpage.
3. Under Security, select **Virtual Private Database**.

   The Virtual Private Database Policies page appears.
4. Under Search, enter the following information, and then click **Go**:
   - **Schema Name:** `OE`
   - **Object Name:** `ORDERS`
   - **Policy Name:** %

   The policy you created, `ACCESSCONTROL_ORDERS`, is listed.
5. Select **ACCESSCONTROL_ORDERS**, and then click **Delete**.
6. In the Confirmation page, click **Yes**.

To remove the user accounts and roles:

1. In Database Control, click **Logout**, and then **Login**.
2. Log in as the administrative user (for example, `SYSTEM`) who created the user accounts and roles used in this tutorial.
3. Click **Server** to display the Server subpage.
4. Under Security, select **Users**.

   The Users page appears.
5. Select each of the following users, and then click **Delete** to remove them:
   - `LDORAN`
   - `LPOPP`

   Do not remove `sec_admin` because you will need this account for later tutorials in this guide.
6. Exit Database Control.

Enforcing Row-Level Security with Oracle Label Security

Oracle Label Security (OLS) provides row-level security for your database tables. You can accomplish this by assigning one or more security labels that define the level of security you want for the data rows of the table.

This section contains:

- About Oracle Label Security
- Guidelines for Planning an Oracle Label Security Policy

- Tutorial: Applying Security Labels to the HR.LOCATIONS Table

About Oracle Label Security

You use Oracle Label Security to secure your database tables at the row level, and assign these rows different levels of security based on the needs of your site. For example, rows that contain highly sensitive data can be assigned a label entitled `HIGHLY SENSITIVE`; rows that are less sensitive can be labeled as `SENSITIVE`, and so on. Rows that all users can have access to can be labeled `PUBLIC`. You can create as many labels as you need, to fit your site's security requirements.

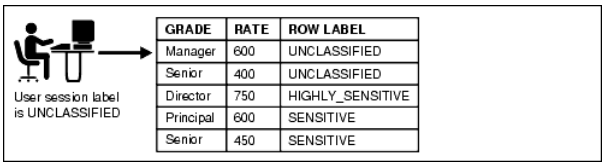
After you create and assign the labels, you can use Oracle Label Security to assign specific users authorization for specific rows, based on these labels. Afterward, Oracle Label Security automatically compares the label of the data row with the security clearance of the user to determine whether the user is allowed access to the data in the row.

An Oracle Label Security policy has the following components:

- **Labels.** Labels for data and users, along with authorizations for users and program units, govern access to specified protected objects. Labels are composed of the following:
  - **Levels.** Levels indicate the type of sensitivity that you want to assign to the row, for example, `SENSITIVE` or `HIGHLY SENSITIVE`.
  - **Compartments.** (Optional) Data can have the same level (Public, Confidential and Secret), but can belong to different projects inside a company, for example ACME Merger and IT Security. Compartments represent the projects in this example, that help define more precise access controls. They are most often used in government environments.
  - **Groups.** (Optional) Groups identify organizations owning or accessing the data, for example, UK, US, Asia, Europe. Groups are used both in commercial and government environments, and frequently used in place of compartments due to their flexibility.
- **Policy.** A policy is a name associated with these labels, rules, and authorizations.

You can create Oracle Label Security labels and policies in Database Control, or you can create them using the `SA_SYSDBA`, `SA_COMPONENTS`, and `SA_LABEL_ADMIN` PL/SQL packages. For information about using the PL/SQL packages, see *Oracle Label Security Administrator's Guide*. This guide explains how to create Oracle Label Security labels and policies by using Database Control.

For example, assume that a user has the `SELECT` privilege on an application table. As illustrated in the following figure, when the user runs a `SELECT` statement, Oracle Label Security evaluates each row selected to determine whether the user can access it. The decision is based on the privileges and access labels assigned to the user by the security administrator. You can also configure Oracle Label Security to perform security checks on `UPDATE`, `DELETE`, and `INSERT` statements.



Guidelines for Planning an Oracle Label Security Policy

Before you create an Oracle Label Security policy, you must determine where and how to apply the labels to the application schema.

To determine where and how to apply Oracle Label Security policies for application data, follow these guidelines:

1. **Analyze the application schema.** Identify the tables that require an Oracle Label Security policy. In most cases, only a small number of the application tables will require an Oracle Label Security policy. For example, tables that store lookup values or constants usually do not need to be protected with a security policy. However, tables that contain sensitive data, such as patient medical histories or employee salaries, do.
2. **Analyze the use of data levels**. After you identify the candidate tables, evaluate the data in the tables to determine the level of security for the table. Someone who has broad familiarity with business operations can provide valuable assistance with this stage of the analysis.

   Data levels refer to the sensitivity of the data. `PUBLIC`, `SENSITIVE`, and `HIGHLY SENSITIVE` are examples of data levels. You should also consider future sensitivities. Doing so creates a robust set of label definitions.

   Remember that if a data record is assigned a sensitivity label whose level component is lower than the clearance of the user, then a user attempting to read the record is granted access to that row.
3. **Analyze the use of data compartments.** Data compartments are used primarily in government environments. If your application is a commercial application, in most cases, you will not create data compartments.
4. **Analyze the data groups.** Data groups and data compartments are typically used to control access to data by organization, region, or data ownership. For example, if the application is a sales application, access to the sales data can be controlled by country or region.

   When a data record is assigned a sensitivity label with compartments and groups, a user attempting to read the record must have a user clearance that contains a level that is equal to or greater than the level of the data label, all of its compartments, and at least one of the groups in the sensitivity label. Because groups are hierarchical, a user could have the parent of one of the groups in the sensitivity label assigned to the data label and still be able to access that record.
5. **Analyze the user population.** Separate the users into one or more designated user types. For example, a user might be designated as a typical user, privileged user, or administrative user. After you create these categories of users, compare the categories with the data levels you created in Step 2. They must correspond correctly for each table identified during the schema analysis you performed in Step 1. Then, compare the organizational structure of the user population with the data groups that you identified in Step 4.
6. **Examine the highly privileged and administrative users to determine which Oracle Label Security authorizations should be assigned to the user.** Oracle Label Security has several special authorizations that can be assigned to users. In general, typical users do not require any special authorizations. See *Oracle Label Security Administrator's Guide* for a complete list of these authorizations.
7. **Review and document the data you gathered.** This step is crucial for continuity across the enterprise, and the resulting document should become part of the

enterprise security policy. For example, this document should contain a list of protected application tables and corresponding justifications.

Tutorial: Applying Security Labels to the HR.LOCATIONS Table

This tutorial demonstrates the general concepts of using Oracle Label Security. In it, you will apply security labels to the `HR.LOCATIONS` table. Three users, `sking`, `kpartner`, and `ldoran` will have access to specific rows within this table, based on the cities listed in the `LOCATIONS` table.

With Oracle Label Security, you restrict user access to data by focusing on row data, and designing different levels of access based on the sensitivity of your data. If you must restrict user access by focusing on user privileges, or some other method such as the job title that the user in your organization has, you can create a PL/SQL function or procedure to use with a Virtual Private Database policy. See "Controlling Data Access with Oracle Virtual Private Database" on page 6-12 for more information.

The schema for `HR.LOCATIONS` is as follows:

```
Name                                      Null?    Type
----------------------------------------- -------- -------------
LOCATION_ID                               NOT NULL NUMBER(4)
STREET_ADDRESS                                     VARCHAR2(40)
POSTAL_CODE                                        VARCHAR2(12)
CITY                                      NOT NULL VARCHAR2(30)
STATE_PROVINCE                                     VARCHAR2(25)
COUNTRY_ID                                         CHAR(2)
```

You will apply the following labels:

Label	Privileges
`CONFIDENTIAL`	Read access to the cities Munich, Oxford, and Roma
`SENSITIVE`	Read access to the cities Beijing, Tokyo, and Singapore
`PUBLIC`	Read access to all other cities listed in `HR.LOCATIONS`

In this tutorial:

- Step 1: Register Oracle Label Security and Enable the LBACSYS Account
- Step 2: Create a Role and Three Users for the Oracle Label Security Tutorial
- Step 3: Create the ACCESS_LOCATIONS Oracle Label Security Policy
- Step 4: Define the ACCESS_LOCATIONS Policy-Level Components
- Step 5: Create the ACCESS_LOCATIONS Policy Data Labels
- Step 6: Create the ACCESS_LOCATIONS Policy User Authorizations
- Step 7: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table
- Step 8: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data
- Step 9: Test the ACCESS_LOCATIONS Policy
- Step 10: Optionally, Remove the Components for This Tutorial

Step 1: Register Oracle Label Security and Enable the LBACSYS Account

In a default Oracle Database installation, Oracle Label Security is installed. However, you must register Oracle Label Security and then enable the default Oracle Label Security account, which is called `LBACSYS`.

- Registering Oracle Label Security with Oracle Database
- Enabling the Default Oracle Label Security User Account LBACSYS

Registering Oracle Label Security with Oracle Database

After you complete the installation, you must register Oracle Label Security with Oracle Database. You can check if Oracle Label Security is already registered by entering the following `SELECT` statement in SQL*Plus. The `PARAMETER` column is case sensitive, so use the case shown here.

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

If the output is `TRUE`, then Oracle Label Security has been registered. Go to "Enabling the Default Oracle Label Security User Account LBACSYS" on page 6-27. If it is `FALSE`, then register Oracle Label Security.

To register Oracle Label Security with Oracle Database:

1. Stop the database, Database Control console process, and listener.
   - **UNIX:** Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and shut down the database. Then from the command line, stop the Database Control console process and listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> SHUTDOWN IMMEDIATE
     SQL> EXIT

     $ emctl stop dbconsole
     $ lsnrctl stop [listener_name]
     ```

     For Oracle RAC installations, shut down each database instance as follows:

     ```
     $ srvctl stop database -d db_name
     ```

   - **Windows:** Stop the database, Database Control console process, and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.
2. Enable Oracle Label Security as follows:
   - **UNIX:** Run the following commands:

     ```
     $ cd $ORACLE_HOME/rdbms/lib
     $ make -f ins_rdbms.mk lbac_on ioracle
     ```

   - **Windows:** In the *ORACLE_BASE**ORACLE_HOME*\bin directory, rename the `oralbacll.dll.dbl` file to `oralbacll.dll`.
3. Restart the database and listener. (Do not restart the Database Control console process yet.)
   - **UNIX:** Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and restart the database. Then from the command line, restart the listener.

For example:

```
sqlplus sys as sysoper
Enter password: password

SQL> STARTUP
SQL> EXIT

$ lsnrctl start [listener_name]
```

For Oracle RAC installations, restart each database instance as follows:

```
$ srvctl start database -d db_name
```

- **Windows:** Restart the database and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.

4. Start Database Configuration Assistant.
   - **UNIX:** Enter the following command at a terminal window:

     ```
     dbca
     ```

     Typically, `dbca` is in the `$ORACLE_HOME/bin` directory.
   - **Windows:** From the **Start** menu, click **All Programs**. Then, click **Oracle - *ORACLE_HOME***, **Configuration and Migration Tools**, and then **Database Configuration Assistant**.

     Alternatively, you can start Database Configuration Assistant at a command prompt:

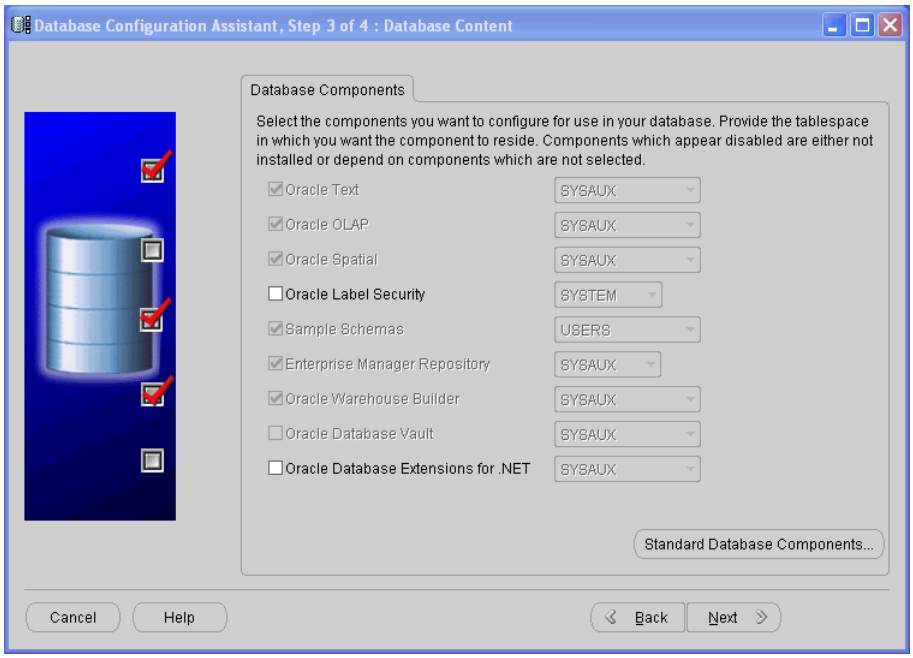
     ```
     dbca
     ```

     As with UNIX, typically, `dbca` is in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\bin` directory.
5. In the Welcome page, click **Next**.

   The Operations page appears.
6. Select **Configure Database Options**, and then click **Next**.

   The Database page appears.
7. From the list, select the database where you installed Oracle Database and then enter the name and password of a user who has been granted the `DBA` role (for example, user `SYS`). Click **Next**.

The Database Content page appears.



8. Select **Oracle Label Security** and then click **Next**.

   The Connection Mode page appears.

9. Select either **Dedicated Server Mode** or **Shared Server Mode** (depending on the selection you made when you created this database), click **Finish**, and then click **OK** in the confirmation prompts.

   Database Configuration Assistant registers Oracle Label Security, and then restarts the database instance.

10. Exit Database Configuration Assistant.

11. Restart the Database Control console process.

    - **UNIX:** Run the following command:

      ```
      $ emctl start dbconsole
      ```

    - **Windows:** Restart the Database Control console process (for example, OracleDBConsoleorcl if the database is named `orcl`) from the Services tool in the Control Panel.

Enabling the Default Oracle Label Security User Account LBACSYS

The Oracle Label Security installation process creates a default user account, `LBACSYS`, who manages the Oracle Label Security features. An administrator can create a user who has the same privileges as this user, that is, `EXECUTE` privileges on the `SA_SYSDBA`, `SA_COMPONENTS`, and `SA_LABEL_ADMIN` PL/SQL packages. By default, `LBACYS` is created as a locked account with its password expired. Your next step is to unlock `LBACYS` and create a new password. Because user `LBACSYS` is using Database Control to create the Oracle Label Security policy, you must grant the `SELECT ANY DICTIONARY` privilege to `LBACSYS`.

To enable the LBACSYS user account:

1. Log in to Database Control as the user `SYS` with the `SYSDBA` privilege.

2. Click **Server** to display the Server subpage.
3. Under Security, select **Users**.

   The Users page appears.
4. Select the **LBACSYS** user, and in the View User page, click **Edit**.

   The Edit User page appears.
5. Next to Status, select **Unlocked**.
6. In the **Enter Password** and **Confirm Password** fields, enter a secure password, according to the guidelines in "Requirements for Creating Passwords" on page 3-8.

   For greater security, do not reuse the same password that was used in previous releases of Oracle Database.
7. Click **Roles** to display the Edit User: LBACSYS page.
8. Click **Edit List**.

   The Modify Roles page appears.
9. In the Available Roles list, select the **SELECT_CATALOG_ROLE** role and then then click **Move** to move it to the Selected Roles list. Then click **OK** to return to the Edit User page.
10. Click **System Privileges**.
11. Click **Edit List**.

    The Modify System Privileges page appears.
12. In the Available System Privileges list, select `SELECT ANY DICTIONARY`, and then click **Move** to move it to the Selected System Privileges list. Then click **OK** to return to the Edit User page.
13. Select **Object Privileges**.
14. In the **Select Object Type** list, select **Package** and then click **Add**.
15. In the Add Package Object Privileges page, do the following:
    a. Under Select Package Objects, select the flashlight icon to display the Select Package Objects window.
    b. Set the Schema to **LBACSYS**.
    c. Enter `%` in the **Search Package Name** field and then click **Go**.
    d. Select all the package objects listed, for both pages of listed objects.
    e. Click **Select** to return to the Add Package Object Privileges window.
    f. Under Available Privileges, move the **EXECUTE** privilege to the Selected Privileges list.
    g. Click **OK**.
16. Click **OK** to return to the Edit User page, and then click **Apply** to apply the changes.

Step 2: Create a Role and Three Users for the Oracle Label Security Tutorial

You are ready to create a role and three users, and then grant these users the role.

- Creating a Role
- Creating the Users

Creating a Role

The `emp_role` role provides the necessary privileges for the three users you will create.

To create the role emp_role:

1. Connect to Database Control as user `SYSTEM`.
2. From the Database Home page, click **Server** to display the Server subpage.
3. In the Security section, click **Roles**.

   The Roles page appears.
4. Click **Create**.

   The Create Role page appears.
5. In the **Name** field, enter `EMP_ROLE` and leave Authentication set to **None**.
6. Select the **Object Privileges** subpage.
7. From the **Select Object Type** list, select **Table**, and then click **Add**.

   The Add Table Object Privileges page appears.
8. Under Select Table Objects, enter `HR.LOCATIONS` to select the `LOCATIONS` table in the `HR` schema, and then under Available Privileges, move `SELECT` to the Selected Privileges list.
9. Click **OK** to return to the Create Role page, and then click **OK** to return to the Roles page.

Creating the Users

The three users you create will have different levels of access to the `HR.LOCATIONS` table, depending on their position. Steven King (`sking`) is the advertising president, so he has full read access to the `HR.LOCATIONS` table. Karen Partners (`kpartner`) is a sales manager who has less access, and Louise Doran (`ldoran`) is a sales representative who has the least access.

To create the users:

1. Ensure that you are logged in to Database Control as `SYSTEM`.

   If you are not already logged in as `SYSTEM`, then select **Logout**, and then select **Login**. In the Login page, enter `SYSTEM` and the password assigned to that account. Set **Connect As** to **Normal**. Select **Login** to log in.

   If you are logged in as `SYSTEM`, click the **Database Instance** link to display the home page.
2. Click **Server** to display the Server subpage.
3. In the Security section, click **Users**.

   The Users page appears.
4. Click **Create**.

   The Create User page appears.
5. Enter the following information:
   - **Name**: `SKING`
   - **Profile**: `DEFAULT`
   - **Authentication**: Password

- **Enter Password** and **Confirm Password**: Enter a password that meets the requirements in "Requirements for Creating Passwords" on page 3-8.
- **Default Tablespace**: `USERS`
- **Temporary Tablespace**: `TEMP`
- **Status:** Set to **Unlocked**.
- **Roles**: Select the **Roles** subpage, and then grant the `emp_role` role to `sking` by selecting **Edit List**. From the Available Roles list, select `emp_role`, and then click **Move** to move it to the Selected Roles list. Click **OK**. In the Create User page, ensure that the **Default** box is selected for both the `CONNECT` and `emp_role` roles.
- **System Privileges**: Select the **System Privileges** subpage and then click **Edit List** to grant the `CREATE SESSION` privileges. Do not grant `sking` the `ADMIN OPTION` option.

6. Click **OK** to return to the Create User page, and then from there, click **OK** to return to the Users page.
7. In the Users page, select `SKING`, set **Actions** to **Create Like**, and then click **Go**.

   The Create User page appears.
8. Create accounts for `kpartner` and `ldoran`.

   Create their names and passwords. (See "Requirements for Creating Passwords" on page 3-8.) You do not need to grant roles or system privileges to them. Their roles and system privileges, defined in the `sking` account, are automatically created.

At this stage, you have created three users who have identical privileges. All of these users have the `SELECT` privilege on the `HR.LOCATIONS` table, through the `EMP_ROLE` role.

Step 3: Create the ACCESS_LOCATIONS Oracle Label Security Policy

Next, you are ready to create the `ACCESS_LOCATIONS` policy.

To create the ACCESS_LOCATIONS policy:

1. Log in to Database Control as user `LBACSYS`.

   Select **Logout**, and then select **Login**. In the Login page, log in as user `LBACSYS`. Set **Connect As** to **Normal**. Select **Login** to log in.
2. Click **Server** to display the Server subpage.
3. In the Security section, click **Oracle Label Security**.

   The Label Security Policies page appears.
4. Click **Create**.
5. In the Create Label Security Policy page, enter the following information:
   - **Name**: `ACCESS_LOCATIONS`
   - **Label Column**: `OLS_COLUMN`

     Later on, when you apply the policy to a table, the label column is added to that table. By default, the data type of the policy label column is `NUMBER(10)`.
   - **Hide Label Column**: Deselect this box so that the label column will not be hidden. (It should be deselected by default.)

Usually, the label column is hidden, but during the development phase, you may want to have it visible so that you can check it. After the policy is created and working, hide this column so that it is transparent to applications. Many applications are designed not to show an another column, so hiding the column prevents the application from breaking.

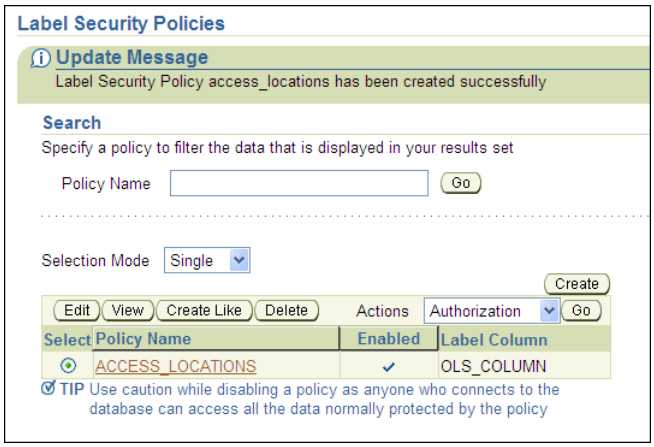
- **Enabled**: Select this box to enable the policy. (It should be enabled by default.)
- **Inverse user's read and write groups (INVERSE_GROUP)**: Do not select this option.
- **Default Policy Enforcement Options**: Select **Apply Policy Enforcements**, and then select the following options:

  For all queries (READ_CONTROL)

  To use session's default label for label column update (LABEL_DEFAULT)

6. Click **OK**.

   The `ACCESS_LOCATIONS` policy appears in the Label Security Policies page.



Step 4: Define the ACCESS_LOCATIONS Policy-Level Components

At this stage, you have the policy and have set enforcement options for it. Next, you are ready to create label components for the policy.

At a minimum, you must create one or more levels, such as `PUBLIC` or `SENSITIVE`; and define a long name, a short name, and a number indicating the sensitivity level. Compartments and groups are optional.

The level numbers indicate the level of sensitivity needed for their corresponding labels. Select a numeric range that can be expanded later on, in case your security policy needs more levels. For example, to create the additional levels `LOW_SENSITIVITY` and `HIGH_SENSITIVITY`, you can assign them numbers 7300 (for `LOW_SENSITIVITY`) and 7600 (for `HIGH_SENSITIVITY`), so that they fit in the scale of security your policy creates. Generally, the higher the number, the more sensitive the data.

Compartments identify areas that describe the sensitivity of the labeled data, providing a finer level of granularity within a level. Compartments are optional.

Groups identify organizations owning or accessing the data. Groups are useful for the controlled dissemination of data and for timely reaction to organizational change. Groups are optional.

In this step, you define the level components, which reflect the names and relationships of the `SENSITIVE`, `CONFIDENTIAL`, and `PUBLIC` labels that you must create for the `ACCESS_LOCATIONS` policy.

To define the label components for the ACCESS_LOCATIONS policy:

1. In the Label Security policies page, select the **ACCESS_LOCATIONS** policy, and then select **Edit**.

   The Edit Label Security Policy page appears.

2. Select the **Label Components** subpage.

3. Under Levels, click **Add 5 Rows**, and then enter a long name, short name, and a numeric tag as follows. (To move from one field to the next, press the **Tab** key.)

Long Name	Short Name	Numeric Tag
SENSITIVE	SENS	3000
CONFIDENTIAL	CONF	2000
PUBLIC	PUB	1000

4. Click **Apply**.

Step 5: Create the ACCESS_LOCATIONS Policy Data Labels

In this step, you create data labels for the policy you created in Step 4: Define the ACCESS_LOCATIONS Policy-Level Components. To create the data label, you must assign a numeric tag to each level. Later on, the tag number will be stored in the security column when you apply the policy to a table. It has nothing to do with the sensitivity of the label; it is only used to identify the labels for the policy.

To create the data labels:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.

2. Select the selection button for the **ACCESS_LOCATIONS** policy.

3. In the Actions list, select **Data Labels**, and then click **Go**.

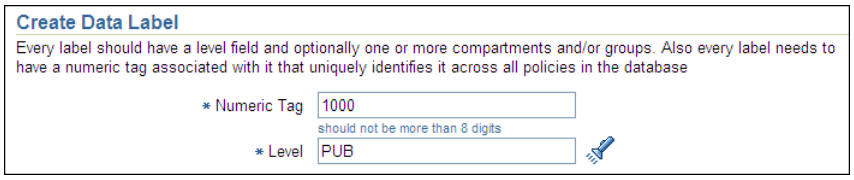
   The Data Labels page appears.

4. Click **Add**.

   The Create Data Label page appears.

5. Enter the following information:

   - **Numeric Tag**: Enter `1000`.
   - **Level**: Enter `PUB`.



6. Click **OK**.

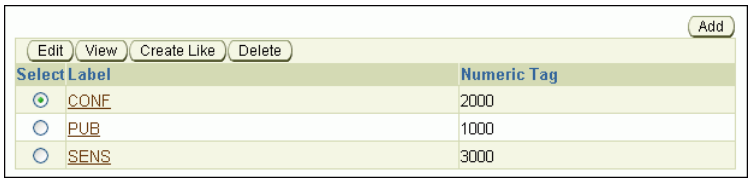
   The data label appears in the Data Labels page.

7. Click **Add** again, and then create a data label for the `CONF` label as follows:

- **Numeric Tag:** Enter `2000`.
- **Level:** Select `CONF` from the list.

8. Click **OK**.
9. Click **Add** again, and then create a data label for the `SENS` label as follows:
   - **Numeric Tag:** Enter `3000`.
   - **Level:** Select `SENS` from the list.
10. Click **OK**.

    At this stage, the `CONF`, `PUB`, and `SENS` labels appear in the Data Labels page.



Later, the tag number will be stored in the security column when you apply the policy to the `HR.LOCATIONS` table. It has nothing to do with the sensitivity of the label; it is only used to identify the labels for the policy.

Step 6: Create the ACCESS_LOCATIONS Policy User Authorizations

Next, you are ready to create user authorizations for the policy.

To create user authorizations for the policy:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.
2. Select the selection button for the `ACCESS_LOCATIONS` policy.
3. In the **Actions** list, select **Authorization**, and then click **Go**.

   The Authorization page appears.
4. Click **Add Users**.

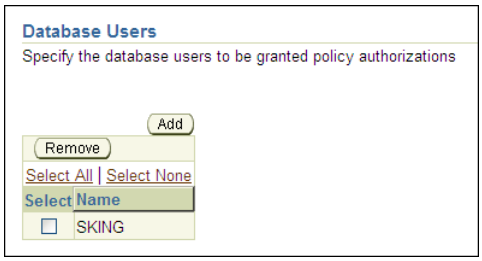
   The Add User: Users page appears.
5. Under Database Users, click **Add**.

   The Search and Select: Userpage appears. Enter `SKING`, and then click **Go**.

   Typically, a database user account already has been created in the database, for example, by using the `CREATE USER` SQL statement.

   The other option is **Non Database Users**. Most application users are considered nondatabase users. A nondatabase user does not exist in the database. This can be any user name that meets the Oracle Label Security naming standards and can fit into the `VARCHAR2(30)` length field. However, be aware that Oracle Database does not automatically configure the associated security information for the nondatabase user when the application connects to the database. In this case, the application must call an Oracle Label Security function to assume the label authorizations of the specified user who is not a database user.
6. Select the check box for user `SKING`, and then click **Select**.

   The Create User page lists user `SKING`.



7. Select the check box for user **SKING** and then click **Next**.

   (You may need to refresh the page to display user `SKING`'s check box.)

8. In the Labels, Compartments and Groups page, enter the following settings:
   - **Maximum Level**: `SENS` (for `SENSITIVE`)
   - **Minimum Level**: `CONF` (for `CONFIDENTIAL`)
   - **Default Level**: `SENS`
   - **Row Level**: `SENS`

9. Click **Next** to go to the Privileges page.

10. In the Privileges page, select **Next** to move to the Audit page.

    Oracle Label Security enforces the policy through the label authorizations. The Privileges page enables the user to override the policy label authorization, so do not select any of its options.

11. In the Audit pane of the Add Users: Audit page, ensure that all of the audit operations are set to `None`, and then click **Next**.

    The Review page appears.

12. Ensure that the settings are correct, and then click **Finish**.

    The Review page lists all the authorization settings you have selected.

13. Repeat Step 4 through Step 12 to create the following authorizations for user `KPARTNER`, so that she can read confidential and public data in `HR.LOCATIONS`.

- **Labels, Compartments And Groups**: Set all four levels to the following:
  - **Maximum Level**: `CONF` (for `CONFIDENTIAL`)
  - **Minimum Level**: `PUB` (for `PUBLIC`)
  - **Default Level**: `CONF`
  - **Row Level**: `CONF`
- **Privileges**: Select no privileges.
- **Audit**: Set all to `None`.

14. Create the following authorizations for user `LDORAN`, who is only allowed to read public data from `HR.LOCATIONS`:
    - **Labels, Compartments And Groups**: Set all four levels to `PUB`.
    - **Privileges**: Select no privileges.
    - **Audit**: Set all to `None`.

Step 7: Apply the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table

Next, you are ready to apply the policy to the `HR.LOCATIONS` table.

To apply the ACCESS_LOCATIONS policy to the HR.LOCATIONS table:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.
2. Select the selection button for the `ACCESS_LOCATIONS` policy.
3. In the **Actions** list, select **Apply**, and then click **Go**.

   The Apply page appears.
4. Click **Create**.

   The Add Table page appears.
5. In the **Table** field, enter `HR.LOCATIONS`.
6. Ensure that the **Hide Policy Column** box is not selected.
7. Ensure that the **Enabled** box is selected.
8. Under Policy Enforcement Options, select **Use Default Policy Enforcement**.

   The default policy enforcement options for `ACCESS_LOCATIONS` are:
   - **For all queries (READ_CONTROL)**
   - **Use session's default label for label column update (LABEL_DEFAULT)**
9. Click **OK**.

   The `ACCESS_LOCATIONS` policy is applied to the `HR.LOCATIONS` table.

Step 8: Add the ACCESS_LOCATIONS Labels to the HR.LOCATIONS Data

After you have applied the `ACCESS_LOCATIONS` policy to the `HR.LOCATIONS` table, you must apply the labels of the policy to the `OLS_COLUMN` in `LOCATIONS`. For the

user HR (the owner of that table) to accomplish this, the user must have FULL access to locations before being able to add the data labels to the hidden OLS_COLUMN column in LOCATIONS.

- Granting HR FULL Policy Privilege for the HR.LOCATIONS Table
- Updating the OLS_COLUMN Table in HR.LOCATIONS

Granting HR FULL Policy Privilege for the HR.LOCATIONS Table

The label security administrative user, LBACSYS, can grant HR the necessary privilege.

To grant HR FULL access to the ACCESS_LOCATIONS policy:

1. Return to the Label Security policies page by selecting the **Label Security Policies** link.
2. Select the selection button for the **ACCESS_LOCATIONS** policy.
3. Select **Authorization** from the **Actions** list, and then click **Go**.

   The Authorization page appears.
4. Click **Add Users**.

   The Add Users page appears.
5. Under Database Users, click **Add**.

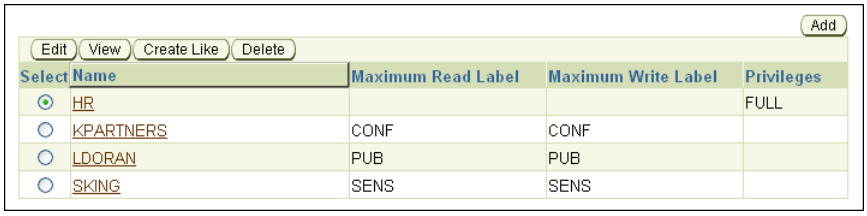
   The Search and Select window appears.
6. Select the box for user HR, and then click **Select**.

   The Create User page lists user HR.
7. Click **Next** to display the Add Users: Levels, Compartments and Groups page, and then click **Next** again to display the Privileges page.
8. Select the **Bypass all Label Security checks (FULL)** privilege, and then click **Next**.

   The Audit page appears.
9. Click **Next**.

   The Review page appears.
10. Click **Finish**.

    At this stage, HR is listed in the Authorization page with the other users.



11. Exit Database Control.

Updating the OLS_COLUMN Table in HR.LOCATIONS

The user HR now can update the OLS_COLUMN column in the HR.LOCATIONS table to include data labels that will be assigned to specific rows in the table, based on the cities listed in the CITY column.

To update the OLS_COLUMN table in HR.LOCATIONS:

1. In SQL*Plus, connect as user HR.

   ```
   CONNECT HR
   Enter password: password
   ```

   If you cannot log in as HR because this account locked and expired, log in as SYSTEM and then enter the following statement. Replace *password* with an appropriate password for the HR account. For greater security, do not reuse the same password that was used in previous releases of Oracle Database. See "Requirements for Creating Passwords" on page 3-8.

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

   After you complete this ALTER USER statement, try logging in as user HR again.

2. Enter the following UPDATE statement to apply the SENS label to the cities Beijing, Tokyo, and Singapore:

   ```
   UPDATE LOCATIONS
   SET ols_column = CHAR_TO_LABEL('ACCESS_LOCATIONS','SENS')
   WHERE UPPER(city) IN ('BEIJING', 'TOKYO', 'SINGAPORE');
   ```

3. Enter the following UPDATE statement to apply the CONF label to the cities Munich, Oxford, and Roma:

   ```
   UPDATE LOCATIONS
   SET ols_column = CHAR_TO_LABEL('ACCESS_LOCATIONS','CONF')
   WHERE UPPER(city) IN ('MUNICH', 'OXFORD', 'ROMA');
   ```

4. Enter the following UPDATE statement to apply the PUB label to the remaining cities:

   ```
   UPDATE LOCATIONS
   SET ols_column = CHAR_TO_LABEL('ACCESS_LOCATIONS','PUB')
   WHERE ols_column IS NULL;
   ```

5. To check that the columns were updated, enter the following statement:

   ```
   SELECT LABEL_TO_CHAR (OLS_COLUMN) FROM LOCATIONS;
   ```

   The following output should appear:

   ```
   LABEL_TO_CHAR(OLS_COLUMN)
   -------------------------------------------------------------------------------
   CONF
   PUB
   SENS
   PUB
   PUB
   PUB
   PUB
   PUB
   PUB
   PUB
   SENS

   LABEL_TO_CHAR(OLS_COLUMN)
   -------------------------------------------------------------------------------
   PUB
   PUB
   SENS
   ```

```
PUB
CONF
PUB
CONF
PUB
PUB
PUB
PUB

LABEL_TO_CHAR(OLS_COLUMN)
--------------------------------------------------------------------------------
PUB

23 rows selected.
```

Note: Using the label column name (OLS_COLUMN) explicitly in the preceding query enables you to see the label column, even if it was hidden.

If the label column is hidden, and you do not specify the label column name explicitly, then the label column is not displayed in the query results. For example, using the SELECT * FROM LOCATIONS query does not show the label column if it is hidden. This feature enables the label column to remain transparent to applications. An application that was designed before the label column was added does not know about the label column and will never see it.

Step 9: Test the ACCESS_LOCATIONS Policy

The ACCESS_LOCATIONS policy is complete and ready to be tested. You can test it by logging in to SQL*Plus as each of the three users and performing a SELECT on the HR.LOCATIONS table.

To test the ACCESS_LOCATIONS policy:

1. In SQL*Plus, connect as user sking.

   ```
   CONNECT sking
   Enter password: password
   ```

2. Enter the following:

   The following commands format the width of the table columns so that you can read them easier. You only need to perform this step once for the entire session (including when kpartner and ldoran log in.)

   ```
   COL city HEADING City FORMAT a25
   COL country_id HEADING Country FORMAT a11
   COL Label format a10
   ```

   Now enter the SELECT statement as follows:

   ```
   SELECT city, country_id, LABEL_TO_CHAR (OLS_COLUMN)
      AS Label FROM hr.locations ORDER BY ols_column;
   ```

   User sking is able to access all 23 rows of the HR.LOCATIONS table. Even though he is only authorized to access rows that are labeled CONF and SENS, he can still read (but not write to) rows labeled PUB.

   ```
   City                      Country     LABEL
   ------------------------- ----------- ----------
   ```

```
Venice                    IT          PUB
Utrecht                   NL          PUB
Bern                      CH          PUB
Geneva                    CH          PUB
Sao Paulo                 BR          PUB
Stretford                 UK          PUB
Mexico City               MX          PUB
Hiroshima                 JP          PUB
Southlake                 US          PUB
South San Francisco       US          PUB
South Brunswick           US          PUB
Seattle                   US          PUB
Toronto                   CA          PUB
Whitehorse                CA          PUB
Bombay                    IN          PUB
Sydney                    AU          PUB
London                    UK          PUB
Oxford                    UK          CONF
Munich                    DE          CONF
Roma                      IT          CONF
Singapore                 SG          SENS
Tokyo                     JP          SENS
Beijing                   CN          SENS

23 rows selected.
```

3. Repeat Steps 1 and 2 for users `kpartner` and `ldoran`.

   User `KPARTNER` can access the rows labeled `CONF` and `PUB`:

```
City                      Country     LABEL
------------------------- ----------- ----------
Venice                    IT          PUB
Utrecht                   NL          PUB
Bern                      CH          PUB
Mexico City               MX          PUB
Hiroshima                 JP          PUB
Southlake                 US          PUB
South San Francisco       US          PUB
South Brunswick           US          PUB
Seattle                   US          PUB
Toronto                   CA          PUB
Whitehorse                CA          PUB
Bombay                    IN          PUB
Sydney                    AU          PUB
London                    UK          PUB
Stretford                 UK          PUB
Sao Paulo                 BR          PUB
Geneva                    CH          PUB
Oxford                    UK          CONF
Munich                    DE          CONF
Roma                      IT          CONF

20 rows selected.
```

   User `LDORAN` can access the rows labeled `PUB`:

```
City                      Country     LABEL
------------------------- ----------- ----------
Venice                    IT          PUB
Hiroshima                 JP          PUB
```

```
Southlake                US          PUB
South San Francisco      US          PUB
South Brunswick          US          PUB
Seattle                  US          PUB
Toronto                  CA          PUB
Whitehorse               CA          PUB
Bombay                   IN          PUB
Sydney                   AU          PUB
London                   UK          PUB
Stretford                UK          PUB
Sao Paulo                BR          PUB
Geneva                   CH          PUB
Bern                     CH          PUB
Utrecht                  NL          PUB
Mexico City              MX          PUB

17 rows selected.
```

4. Exit SQL*Plus.

Step 10: Optionally, Remove the Components for This Tutorial

Remove the components that you created for this tutorial.

To remove the components for this tutorial:

1. In Database Control, connect as user `SYSTEM`.
2. Click **Server** to display the Server subpage.
3. In the Security section, click **Users**.
4. Select user `kpartner`, and then click **Delete**.
5. In the Confirmation page, click **Yes**.
6. Repeat Step 4 and Step 5 for users `ldoran` and `sking`.
7. Click the **Database Instance** link to return to the Database home page.
8. Click **Server** to display the Server subpage.
9. In the Security section, click **Roles**.
10. Select the role `emp_role`, and then click **Delete**.
11. In the Confirmation dialog box, click **Yes**.
12. Log out of Database Control, and then log back in as `LABCSYS`.
13. Click **Server** to display the Server subpage.
14. In the Security section, click **Oracle Label Security**.
15. In the Label Security Policies page, select the `ACCESS_LOCATIONS` policy and then click **Delete**. In the Confirmation page, select the **Drop column** check box and then click **Yes**.

    Deleting the `ACCESS_LOCATIONS` policy also drops the `OLS_COLUMN` column from the `HR.LOCATIONS` table.
16. Log out of Database Control.
17. Optionally, remove Oracle Label Security.

    See *Oracle Label Security Administrator's Guide* for information about removing Oracle Label Security.s

Controlling Administrator Access with Oracle Database Vault

Oracle Database Vault enables you to restrict administrative access to an Oracle database. This helps you address the most difficult security problems remaining today: protecting against insider threats, meeting regulatory compliance requirements, and enforcing separation of duty.

- About Oracle Database Vault
- Tutorial: Controlling Administrator Access to the OE Schema

See Also: *Oracle Database Vault Administrator's Guide*

About Oracle Database Vault

Typically, the main job of an Oracle database administrator is to perform tasks such database tuning, installing upgrades, monitoring the state of the database, and then remedying any problems that he or she finds. In a default Oracle Database installation, database administrators also have the ability to create users and access user data. For greater security, you should restrict these activities only to those users who must perform them. This is called **separation of duty**, and it frees the database administrator to focus on tasks ideally suited to his or her expertise, such as performance tuning.

By restricting administrator access to your Oracle databases, Oracle Database Vault helps you to follow common regulatory compliance requirements, such as the Payment Card Industry (PCI) Data Security Standard (DSS) requirements, Sarbanes-Oxley (SOX) Act, European Union (EU) Privacy Directive, and Healthcare Insurance Portability and Accountability Act (HIPAA). These regulations require strong internal controls on access, disclosure or modification of sensitive information that could lead to fraud, identity theft, financial irregularities and financial penalties.

Oracle Database Vault provides the following ways for you to restrict administrator access to an Oracle database:

- **Group database schemas, objects, and roles that you want to secure.** This grouping is called a **realm**, and all the components of the realm are protected. After you, the Database Vault administrator, create a realm, you designate a user to manage access to the realm. For example, you can create a realm around one table within a schema, or around the entire schema itself.
- **Create PL/SQL expressions to customize your database restrictions.** You create an expression in a **rule**, and for multiple rules within one category, you can group the rules into a **rule set**. To enforce the rules within the rule set, you then associate the rule set with a realm or command rule. For example, if you wanted to prevent access to a database during a maintenance period (for example, from 10 to 12 p.m.), you can create a rule to restrict access only during those hours.
- **Designate specific PL/SQL statements that are accessible or not accessible to users.** These are called **command rules**. A command rule contains a command to be protected and a rule set that determines whether the execution of the command is permitted. You can create a command rule to protect `SELECT`, `ALTER SYSTEM`, database definition language (DDL), and data manipulation language (DML) statements that affect one or more database objects. You can associate a rule set to further customize the command rule.
- **Define attributes to record data such as session users or IP addresses that Oracle Database Vault can recognize and secure.** These attributes are called **factors**. You can use factors for activities such as authorizing database accounts to connect to

the database or creating filtering logic to restrict the visibility and manageability of data. To further customize the factor, you can associate a rule set with it.

- **Design secure application roles that are enabled only by Oracle Database Vault rules.** After you create the secure application role in Oracle Database Vault, you associate a rule set with it. The rule set defines when and how the secure application role is enabled or disabled.

You can create these components by using either Oracle Database Vault Administrator, or by using its PL/SQL packages.

Tutorial: Controlling Administrator Access to the OE Schema

The `OE` schema has several tables that contain confidential data, such as the credit limits allowed for customers and other information. Order Entry tables typically contain sensitive information, such as credit card or Social Security numbers. This type of information must be restricted only to individuals whose job requires access to this information, according to Payment Card Industry (PCI) Data Security Standards (DSS).

In this tutorial, you create a realm around the `OE` schema, which will protect it from administrator access. However, user `SCOTT` needs access to the `OE.CUSTOMERS` table, so you must ensure that he can continue to access this data.

In this tutorial:

- Step 1: Enable Oracle Database Vault
- Step 2: Grant the SELECT Privilege on the OE.CUSTOMERS Table to User SCOTT
- Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT
- Step 4: Create a Realm to Protect the OE.CUSTOMERS Table
- Step 5: Test the OE Protections Realm
- Step 6: Optionally, Remove the Components for This Tutorial

Step 1: Enable Oracle Database Vault

Oracle Database Vault is installed when you perform a default installation of Oracle Database. After you install it, you must register Oracle Database Vault with Oracle Database and then enable the Oracle Database Vault Account Manager user account.

- Checking if Oracle Database Vault Is Enabled
- Registering Oracle Database Vault with Oracle Database
- Enabling Database Access Control for the Database Vault Account Manager Account

Checking if Oracle Database Vault Is Enabled You can check if Oracle Database Vault is enabled by logging in to SQL*Plus and entering the following `SELECT` statement. The `PARAMETER` column is case sensitive, so use the case shown here.

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

If it returns `TRUE`, then Oracle Database Vault is registered. Go to "Enabling Database Access Control for the Database Vault Account Manager Account" on page 6-45. Otherwise, complete the registration process described in the next section.

Registering Oracle Database Vault with Oracle Database In the registration process, Oracle Database Vault is enabled and you are prompted to create its administrative accounts.

To register Oracle Database Vault:

1. Stop the database, Database Control console process, and listener.
   - **UNIX:** Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and shut down the database. Then from the command line, stop the Database Control console process and listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> SHUTDOWN IMMEDIATE
     SQL> EXIT

     $ emctl stop dbconsole
     $ lsnrctl stop [listener_name]
     ```

     For Oracle RAC installations, shut down each database instance as follows:

     ```
     $ srvctl stop database -d db_name
     ```

   - **Windows:** Stop the database, Database Control console process, and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.
2. Enable Oracle Database Vault as follows:
   - **UNIX:** Run the following commands. The `make` command enables both Oracle Database Vault (`dv_on`) and Oracle Label Security (`lbac_on`). You must enable Oracle Label Security before you can use Database Vault.

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

   - **Windows:** In the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\bin` directory, rename the `oradvll.dll.dbl` file to `oradvll.dll`. If Oracle Label Security has not been enabled, then change the name of the `oralbacll.dll.dbl` file to `oralbacll.dll`. You must enable Oracle Label Security before you can use Database Vault.
3. Restart the database and listener. (Do not restart the Database Control console process yet.)
   - **UNIX:** Log in to SQL*Plus as user `SYS` with the `SYSOPER` privilege and restart the database. Then from the command line, restart the listener.

     For example:

     ```
     sqlplus sys as sysoper
     Enter password: password

     SQL> STARTUP
     SQL> EXIT

     $ lsnrctl start [listener_name]
     ```

     For Oracle RAC installations, restart each database instance as follows:

     ```
     $ srvctl start database -d db_name
     ```

   - **Windows:** Restart the database and listener from the Services tool in the Control Panel. The names of Oracle Database services begin with `Oracle`.

4. Start Database Configuration Assistant.
   - **UNIX:** Enter the following command at a terminal window:

     ```
     dbca
     ```

     Typically, `dbca` is in the `$ORACLE_HOME/bin` directory.
   - **Windows:** From the **Start** menu, click **All Programs**. Then, click **Oracle - *ORACLE_HOME***, **Configuration and Migration Tools**, and then **Database Configuration Assistant**.

     Alternatively, you can start Database Configuration Assistant at a command prompt:

     ```
     dbca
     ```

     As with UNIX, typically, `dbca` is in the *`ORACLE_BASE`*`\`*`ORACLE_HOME`*`\bin` directory.
5. In the Welcome page, click **Next**.

   The Operations page appears.
6. Select **Configure Database Options**, and then click **Next**.

   The Database page appears.
7. From the list, select the database where you installed Oracle Database and then enter the name and password of a user who has been granted the `DBA` role. Click **Next**.

   The Database Content page appears.
8. Perform one of the following actions:
   - **If Oracle Label Security is already enabled:** Select the **Oracle Database Vault** option, and then click **Next**.
   - **If Oracle Label Security is not enabled:** Select the **Oracle Label Security** option so that the **Oracle Database Vault** option becomes available for selection. Select the **Oracle Database Vault** option as well, and then click **Next**.

   The Oracle Database Vault Credentials page appears.
9. Specify the name and password for the Database Vault Owner account (for example, `DBVOWNER`) and the Database Vault Account Manager (for example, `DBVACCTMGR`).

   Enter any password that is secure, according to the password guidelines described in "Requirements for Creating Passwords" on page 3-8. Oracle Database Vault has additional password requirements, which are displayed if you try to create an incorrect password.
10. Click **Next**.

    The Connection Mode page appears.
11. Select either **Dedicated Server Mode** or **Shared Server Mode** (depending on the selection you made when you created this database), click **Finish**, and then click **OK** in the confirmation prompts.

    Database Configuration Assistant registers Oracle Database Vault, and then restarts the database instance.
12. Exit Database Configuration Assistant.

13. Restart the Database Control console process.
    - **UNIX:** Run the following command:

      ```
      $ emctl start dbconsole
      ```

    - **Windows:** Restart the Database Control console process (for example, OracleDBConsoleorcl if the database is named `orcl`) from the Services tool in the Control Panel.

Enabling Database Access Control for the Database Vault Account Manager Account The Database Vault Account Manager account must have additional privileges to use Database Control.

To grant the necessary privileges to the Database Vault Account Manager account:

1. Log in to Database Control as the user `SYS`.

   In the Login page, enter `SYS` and the password assigned to `SYS`. Set **Connect As** to **SYSDBA**. Select **Login** to log in. See *Oracle Database 2 Day DBA* for instructions about how to start Database Control.
2. Click **Server** to display the Server subpage.
3. Under Security, select **Users**.

   The Users page appears.
4. Select the Database Vault Account Manager account (for example, `DBVACCTMGR`).

   To quickly find `DBVACCTMGR`, enter `DBV` in the **Object Name** field, and then click **Go**.
5. Select the **DBVACCTMGR** user, and in the View User page, click **Edit**.

   The Edit User page appears.
6. Click **Roles** to display the Edit User: DBVACCTMGR page.
7. Click **Edit List**.

   The Modify Roles page appears.
8. In the Available Roles list, select the **SELECT_CATALOG_ROLE** role and then then click **Move** to move it to the Selected Roles list. Then click **OK** to return to the Edit User page.
9. Click **System Privileges**.
10. Click **Edit List**.

    The Modify System Privileges page appears.
11. In the Available System Privileges list, select `SELECT ANY DICTIONARY`, and then click **Move** to move it to the Selected System Privileges list. Then click **OK**.
12. Click **Apply**.

Step 2: Grant the SELECT Privilege on the OE.CUSTOMERS Table to User SCOTT

To test the tutorial later on, user `SCOTT` must select from the `OE.CUSTOMERS` table. First, you should ensure that he `SCOTT` account is active.

To enable user SCOTT:

1. In Database Control, connect as the Oracle Database Vault Account Manager account with the `Normal` privilege.

After you install Oracle Database Vault, you no longer can use the administrative accounts (such as SYS and SYSTEM) to create or enable user accounts. This is because right out of the box, Oracle Database Vault provides separation-of-duty principles to administrative accounts. From now on, to manage user accounts, you must use the Oracle Database Vault Account Manager account.

However, administrative users still have the privileges they do need. For example, user SYS, who owns system privileges and many PL/SQL packages, can still grant privileges on these to other users. However, user SYS can no longer create, modify, or drop user accounts.

2. Click **Server** to display the Server subpage.
3. Under Security, select **Users**.

   The Users page appears.
4. Select the **SCOTT** user, and in the View User page, click **Edit**.

   The Edit User page appears.
5. Enter the following settings:
   - **Enter Password** and **Confirm Password**: If the SCOTT account password status is expired, then enter a new password. Enter any password that is secure, according to the password guidelines described in "Requirements for Creating Passwords" on page 3-8.
   - **Status**: Click **Unlocked**.
6. Click **Apply**.
7. Click **Logout**.

To grant user SCOTT the SELECT privilege on the OE.CUSTOMERS table:

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

```
sqlplus oe
Enter password: password
Connected.
```

2. Grant user SCOTT the SELECT privilege on the OE.CUSTOMERS table.

```
SQL> GRANT SELECT ON CUSTOMERS TO SCOTT;
```

Step 3: Select from the OE.CUSTOMERS Table as Users SYS and SCOTT

At this stage, both users SYS and SCOTT can select from the OE.CUSTOMERS table, because SYS has administrative privileges and because SCOTT has an explicit SELECT privilege granted by user OE.

To select from OE.CUSTOMERS as users SYS and SCOTT:

1. In SQL*Plus, connect as user SYS using the SYSDBA privilege

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

2. Select from the OE.CUSTOMERS table as follows:

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

The following output should appear

```
COUNT(*)
--------
     319
```

3. Connect as user SCOTT, and then perform the same SELECT statement.

```
CONNECT SCOTT
Enter password: password
Connected.

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

The following output should appear:

```
COUNT(*)
--------
     319
```

Step 4: Create a Realm to Protect the OE.CUSTOMERS Table

To restrict the OE.CUSTOMER table from administrative access, you will create a realm around the OE schema.

To create a realm around the OE schema:

1. Start Oracle Database Vault Administrator.

   In a browser, enter the following URL:

   https://*host_name*:*port*/dva

   Replace *host_name* with the name of the server on which you installed Oracle Database Vault, and *port* with the Oracle Enterprise Manager Console HTTPS port number. In most cases, the name of the server and port number are the same as those used by Database Control.

   If you cannot start Database Vault Administrator, you may need to manually deploy it. See *Oracle Database Vault Administrator's Guide* for more information.

2. In the Login to Database page, enter the following information:

   - **User Name**: Enter the name of the DV_OWNER or DV_ADMIN account (for example, DBVOWNER).
   - **Password**: Enter the password of the user whose name you entered.
   - **Host**: Enter the host name or IP address of the computer where you installed Oracle Database Vault, for example, myserver.us.example.com.
   - **Port**: Enter the port number for the database, for example, 1521.
   - **SID/Service**: Enter either the SID (for example, orcl) of the database, or the service (for example, myserver.us.example.com).

   The Database Instance Administration page appears.

3. Under Database Vault Feature Administration, select **Realms**.

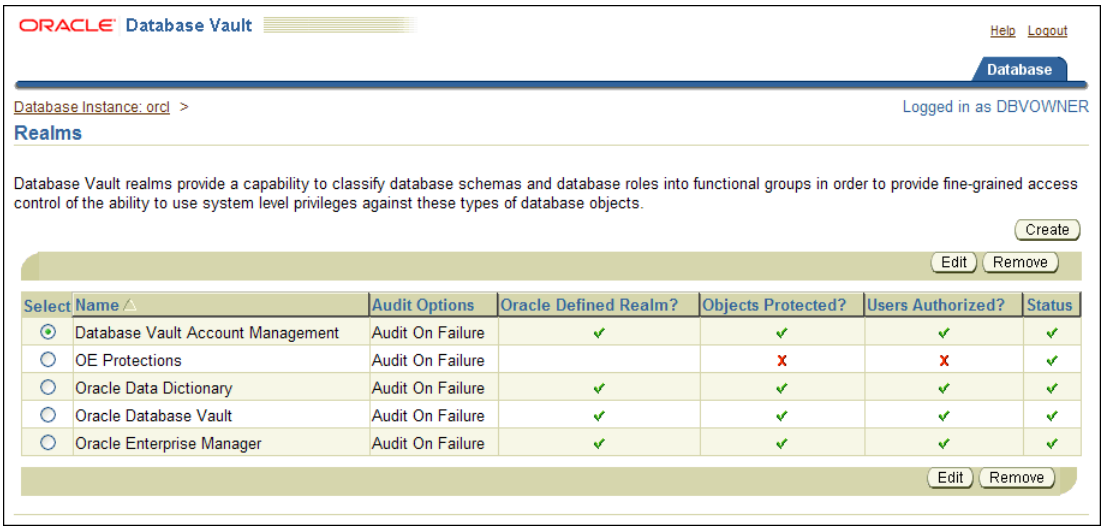
   The Realms page appears.

4. Click **Create**.

   The Create Realm page appears.

5. Enter the following information:
   - **Name**: `OE Protections`
   - **Description**: `Realm to protect the OE schema`
   - **Status**: Click **Enabled**.
   - **Audit Options**: Select **Audit on Failure**.
6. Click **OK**.

   The Realms page appears, with the `OE` schema listed as a realm. However, it has no protected objects or authorized users yet.



7. Select the **OE Protections** realm and then click **Edit**.

   The Edit Realm page appears.
8. Under Realm Secured Objects, click **Create**.

   The Create Realm Secured Object page appears.
9. From the **Object Owner** list, select **OE**.
10. From the **Object Type** list, select **TABLE**.
11. In the **Object Name** field, enter `%` to specify all tables within the `OE` schema, and then click **OK**.

    The Edit Realm page appears.
12. Under **Realm Authorizations**, click **Create**.

    The Create Realm Authorization page appears.
13. From the **Grantee** list, select `OE [USER]`, and then set the **Authorization Type** to **Owner**. Then set **Authorization Rule Set** to **<Non Selected>**.

    This authorizes the `OE` user to manage access to the objects within the `OE` schema. As an Owner, the `OE` user can grant or revoke realm-secured database roles, and access, manipulate, and create objects protected by the `OE` Protections realm.

    The **Authorization Rule Set** list enables to you select a rule that further controls access, such as the time the realm is in effect, and so on.

14. Click **OK** to return to the Edit Realm page, and then click **OK** again to return to the Realms page.
15. Do not exit Oracle Database Vault Administrator.

Step 5: Test the OE Protections Realm

Now that you have created a realm to protect the OE schema, you are ready to test it. You do not need to restart the database session, because any protections you define in Oracle Database Vault take effect right away.

To test the OE Protections realm:

1. Connect to SQL*Plus as user SYS using the SYSDBA privilege.

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

2. Try selecting from the OE.CUSTOMERS table.

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

The following output should appear:

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

The OE Protections realm prevents the administrative user from accessing the OE.CUSTOMERS table. Because you defined the OE Protections realm to protect the entire schema, the administrative user does not have access to any of the other tables in OE, either.

3. Connect as user SCOTT.

```
CONNECT SCOTT
Enter password: password
Connected.
```

4. Try selecting from the OE.CUSTOMERS table.

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

The following output should appear:

```
  COUNT(*)
----------
       319
```

The OE Protections realm does not apply to user SCOTT because user OE has explicitly granted this user the SELECT privilege on the OE.CUSTOMERS table. Oracle Database Vault sets up the protections you need, but does not override the explicit privileges you have defined. SCOTT still can query this table.

Step 6: Optionally, Remove the Components for This Tutorial

After completing this tutorial, you can remove the data structures that you used if you no longer need them.

To revoke the SELECT privilege on OE.CUSTOMERS from user SCOTT:

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

```
CONNECT OE
```

```
Enter password: password
Connected.
```

2. Revoke the `SELECT` privilege from user `SCOTT`.

```
REVOKE SELECT ON CUSTOMERS FROM SCOTT;
```

To drop the OE Protections realm:

1. If you have logged out of Oracle Database Vault Administrator, log back in as the Database Vault Owner account that you created when you installed Oracle Database Vault (for example, `DBVOWNER`).

   See Step 1 in "Step 4: Create a Realm to Protect the OE.CUSTOMERS Table" on page 6-47 for how to start Database Vault Administrator.

   The Administration page appears.

2. Under Database Vault Feature Administration, click **Realms**.

   The Realms page appears.

3. Select **OE Protections** from the list of realms, and then click **Remove**. Then click **Yes** in the Confirmation page.

4. Log out of Oracle Database Vault Administrator.

To disable Oracle Database Vault:

- See *Oracle Database Vault Administrator's Guide* for information about disabling Oracle Database Vault.

7

Auditing Database Activity

This chapter contains:

- About Auditing
- Why Is Auditing Used?
- Where Are Standard Audit Activities Recorded?
- Auditing General Activities Using Standard Auditing
- Tutorial: Creating a Standard Audit Trail
- Guidelines for Auditing
- Initialization Parameters Used for Auditing

See Also:

- *Oracle Database Security Guide* for other ways that you can audit user and database activities
- *Oracle Audit Vault Administrator's Guide* for information about Oracle Audit Vault, which provides advanced auditing features

About Auditing

Auditing is the monitoring and recording of selected user database actions. In standard auditing, you use initialization parameters and the `AUDIT` and `NOAUDIT` SQL statements to audit SQL statements, privileges, and schema objects, and network and multitier activities.

There are also activities that Oracle Database always audits, regardless of whether auditing is enabled. These activities are administrative privilege connections, database startups, and database shutdowns. See *Oracle Database Security Guide* for more information.

Another type of auditing is fine-grained auditing. Fine-grained auditing enables you to audit at the most granular level, data access, and actions based on content, using Boolean measurement, such as `value > 1000`. You can use fine-grained auditing to audit activities based on access to or changes in a column. You can create security policies to trigger auditing when someone accesses or alters specified elements in an Oracle database, including the contents within a specified object. You can create policies that define specific conditions that must take place for the audit to occur. For example, you can audit a particular table column to find out when and who tried to access it during a specified period of time. Furthermore, you can create alerts that are triggered when the policy is violated, and write this data to a separate audit file. *Oracle Database Security Guide* explains how to perform fine-grained auditing.

Why Is Auditing Used?

You typically use auditing to perform the following activities:

- **Enable accountability for actions.** These include actions taken in a particular schema, table, or row, or affecting specific content.
- **Deter users from inappropriate actions based on that accountability.**
- **Investigate suspicious activity.** For example, if a user is deleting data from tables, then a security administrator might decide to audit all connections to the database and all successful and unsuccessful deletions of rows from all tables in the database.
- **Notify an auditor of actions by an unauthorized user.** For example, an unauthorized user could change or delete data, or a user has more privileges than expected, which can lead to reassessing user authorizations.
- **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 do 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
- **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/O operations are performed, or how many concurrent users connect at peak times.

Where Are Standard Audit Activities Recorded?

Oracle Database records audit activities in audit records. Audit records provide information about the operation that was audited, the user performing the operation, and the date and time of the operation. Audit records can be stored in either a data dictionary table, called the **database audit trail**, or in operating system files, called an **operating system audit trail**. Oracle Database also provides a set of data dictionary views that you can use to track suspicious activities. See *Oracle Database Security Guide* for more information about these views.

When you use standard auditing, Oracle Database writes the audit records to either to `DBA_AUDIT_TRAIL` (the `SYS.AUD$` table), the operating system audit trail, or to the `DBA_COMMON_AUDIT_TRAIL` view, which combines standard and fine-grained audit log records.

In addition, the actions performed by administrators are recorded in the `syslog` audit trail when the `AUDIT_SYSLOG_LEVEL` initialization parameter is set.

Auditing General Activities Using Standard Auditing

This section explains how to use standard auditing to audit activities performed on SQL statements, privileges, schema objects, and network or multitier activities.

This section contains:

- About Standard Auditing
- Enabling or Disabling the Standard Audit Trail
- Using Default Auditing for Security-Relevant SQL Statements and Privileges
- Individually Auditing SQL Statements
- Individually Auditing Privileges
- Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment
- Individually Auditing Schema Objects
- Auditing Network Activity
- Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment
- Tutorial: Creating a Standard Audit Trail

> **See Also:** *Oracle Database Security Guide* for detailed information about managing the standard audit trail

About Standard Auditing

In standard auditing, you enable auditing of SQL statements, privileges, schema objects, and network or multitier activities. You can audit a specific schema table if you want. To perform this type of audit, you use Database Control.

You can view the standard audit trail by querying the `DBA_AUDIT_TRAIL` and `DBA_COMMON_AUDIT_TRAIL` data dictionary views.

> **See Also:** *Oracle Database Security Guide* for a roadmap of how and why you can use the different types of audit options available

Enabling or Disabling the Standard Audit Trail

Before you perform the standard auditing procedures described in this section, you must enable standard auditing. When you enable standard auditing, you can create the audit trail in the database audit trail or write the audit activities to an operating system file. If you write to an operating system file, you can create the audit record in text or XML format.

To enable or disable the standard audit trail:

1. Start Database Control.
2. Log in as `SYS` and connect with the `SYSDBA` privilege.
   - **User Name**: `SYS`
   - **Password**: Enter your password.
   - **Connect As**: `SYSDBA`
3. Click **Server** to display the Server subpage.
4. In the Database Configuration section, click **Initialization Parameters**.

   The Initialization Parameters page appears.

5. Click **SPFile** to display the SPFile subpage.

   If the **SPFile** tab does not display in your installation, then you did not install Oracle Database using a server parameters file. Go to the next step.

6. In the **Name** field, enter `audit_trail` to find the `AUDIT_TRAIL` initialization parameter, and then click **Go**.

   You can enter the first few characters of the parameter, for example, `AUDIT_`. Alternatively, you can scroll down the list of parameters to find the `AUDIT_TRAIL` parameter.

7. In the **Value** field, select one of the following values:

   - `DB`: Enables database auditing and directs standard audit records to the database audit trail (`SYS.AUD$`), except for records that are always written to the operating system audit trail. (This value is the default if you created the database using Database Configuration Assistant. Otherwise, the default is `NONE`.)
   - `OS`: Enables database auditing and directs all audit records to an operating system file. Writing the audit trail to operating system files is better for performance instead of sending the audit records to the `SYS.AUD$` table. If the auditor is distinct from the database administrator, you must use the `operating system` setting. Any auditing information stored in the database is viewable and modifiable by the database administrator.

     To specify the location of the operating system audit record file, set the `AUDIT_FILE_DEST` initialization parameter. The first default directory is `$ORACLE_BASE/admin/$ORACLE_SID/adump`.
   - `NONE`: Disables standard auditing.
   - `DB, EXTENDED`: Performs all actions of the `AUDIT_TRAIL=DB` setting and also populates the SQL bind and SQL text CLOB-type columns of the `SYS.AUD$` table, when available. (These two columns are populated only when this parameter is specified.)
   - `XML`: Writes to the operating system audit record file in XML format. Prints all elements of the `AuditRecord` node (as specified by the by the XML schema in `http://xmlns.oracle.com/oracleas/schema/dbserver_audittrail-11_2.xsd` XSD file) except `Sql_Text` and `Sql_Bind` to the operating system XML audit file. This `.xsd` file represents the schema definition of the XML audit file. An XML schema is a document written in the XML Schema language.
   - `EXTENDED`: Specifies `XML, EXTENDED`, which performs all actions of `XML` and also populates the SQL bind and SQL text CLOB-type columns of the `SYS.AUD$` table, wherever possible. (These columns are populated only when this parameter is specified.)

   For a detailed explanation of the `AUDIT_TRAIL` initialization parameter settings, see *Oracle Database Security Guide*.

8. Click **Apply**.

9. Restart the Oracle Database instance:

   a. Click the **Database Instance** link.

   b. Click **Home** to display the Database Control home page.

   c. Under General, click **Shutdown**.

d. In the Startup/Shutdown Credentials page, enter your credentials.

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

e. After the shutdown completes, click **Startup**.

Note the following:

- You do not need to restart the database if you change the auditing of objects. You only need to restart the database if you made a universal change, such as turning on or off *all* auditing or changing the destination of the audit trail operating system files.
- You do not need to set `AUDIT_TRAIL` to enable either fine-grained auditing or `SYS` auditing. (`SYS` auditing enables you to monitor the activities of a system administrator. See *Oracle Database Security Guide* for more information.) For fine-grained auditing, you add and remove fine-grained auditing policies as necessary, applying them to the specific operations or objects you want to monitor. You can use the `AUDIT_SYS_OPERATIONS` parameter to enable and disable `SYS` auditing.

Using Default Auditing for Security-Relevant SQL Statements and Privileges

When you use Database Configuration Assistant (DBCA) to create a new database, Oracle Database configures the database to audit the most commonly used security-relevant SQL statements and privileges. It also sets the `AUDIT_TRAIL` initialization parameter to `DB`. If you decide to use a different audit trail type (for example, `OS` if you want to write the audit trail records to operating system files), then you can do that: Oracle Database continues to audit the privileges that are audited by default. If you disable auditing by setting the `AUDIT_TRAIL` parameter to `NONE`, then no auditing takes place.

Oracle Database audits the following privileges by default:

```
ALTER ANY PROCEDURE   CREATE ANY LIBRARY           DROP ANY TABLE
ALTER ANY TABLE       CREATE ANY PROCEDURE         DROP PROFILE
ALTER DATABASE        CREATE ANY TABLE             DROP USER
ALTER PROFILE         CREATE EXTERNAL JOB          EXEMPT ACCESS POLICY
ALTER SYSTEM          CREATE PUBLIC DATABASE LINK  GRANT ANY OBJECT PRIVILEGE
ALTER USER            CREATE SESSION               GRANT ANY PRIVILEGE
AUDIT SYSTEM          CREATE USER                  GRANT ANY ROLE
CREATE ANY JOB        DROP ANY PROCEDURE
```

Oracle Database audits the following SQL statement shortcuts by default:

```
ROLE                  SYSTEM AUDIT          PUBLIC SYNONYM
DATABASE LINK         PROFILE               SYSTEM GRANT
```

To individually control the auditing of SQL statements and privileges, use the `AUDIT` and `NOAUDIT` statements.

Oracle strongly recommends that you audit the database. 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 catch any activities that may deviate from company policy. Doing so translates into tightly controlled access to your database and the

application software. By enabling auditing by default, you can generate an audit record for audit and compliance personnel.

Note: If your applications use the default audit settings from Oracle Database 10*g* Release 2 (10.2), then you can revert to these audit settings until you modify the applications to use the Release 11*g* audit settings. To do so, run the `undoaud.sql` script.

After you have modified your applications to conform to the Release 11*g* audit settings, then you can manually update your database to use the audit configuration that suits your business needs, or you can run the `secconf.sql` script to apply the Release 11g default audit settings.

The `undoaud.sql` and `secconf.sql` scripts are in the `$ORACLE_HOME/rdbms/admin` directory. The `undoaud.sql` script affects audit settings only, and the `secconf.sql` script affects both audit and password settings. They have no effect on other security settings.

See Also:

- *Oracle Database SQL Language Reference* for detailed information about the SQL statements described in this section
- *Oracle Database Reference* for detailed information about the `AUDIT_TRAIL` initialization parameter

Individually Auditing SQL Statements

The SQL statements that you can audit are in the following categories:

- **DDL statements.** For example, enabling the auditing of tables (`AUDIT TABLE`) audits all `CREATE` and `DROP TABLE` statements
- **DML statements.** For example, enabling the auditing of `SELECT TABLE` audits all `SELECT ... FROM TABLE/VIEW` statements, regardless of the table or view

Statement auditing can be broad or focused, for example, by auditing the activities of all database users or of only a select list of users.

See Also: *Oracle Database Security Guide* for detailed information about auditing SQL statements

Individually Auditing Privileges

Privilege auditing is a way to audit statements that can use a system privilege. For example, you can audit the `SELECT ANY TABLE` privilege if you want to audit all the `SELECT` statements that will use the `SELECT ANY TABLE` privilege. You can audit the use of any system privilege. Similar to statement auditing, privilege auditing can audit the activities of all database users or of only a specified list. As with SQL statement auditing, you use the `AUDIT` and `NOAUDIT` statements to enable and disable privilege auditing. In addition, you must have the `AUDIT SYSTEM` system privilege before you can enable auditing.

Privilege audit options match the corresponding system privileges. For example, the option to audit use of the `DELETE ANY TABLE` privilege is `DELETE ANY TABLE`. For example:

```
AUDIT DELETE ANY TABLE BY ACCESS WHENEVER NOT SUCCESSFUL;
```

To audit all successful and unsuccessful uses of the `DELETE ANY TABLE` system privilege, enter the following statement:

```
AUDIT DELETE ANY TABLE BY ACCESS;
```

To audit all unsuccessful `SELECT`, `INSERT`, and `DELETE` statements on all tables and unsuccessful uses of the `EXECUTE PROCEDURE` system privilege, by all database users, and by individual audited statement, issue the following statement:

```
AUDIT SELECT TABLE, INSERT TABLE, DELETE TABLE, EXECUTE PROCEDURE BY ACCESS
WHENEVER NOT SUCCESSFUL;
```

> **See Also:** *Oracle Database Security Guide* for detailed information about auditing privileges

Using Proxies to Audit SQL Statements and Privileges in a Multitier Environment

You can audit the activities of a client in a multitier environment by specifying a proxy in the Add Audited Statements or Add Audited Privileges page in Database Control. You can use the SQL `AUDIT` statement to audit the activities of a client in a multitier environment. To do so, use the `BY` *`user`* clause in the `AUDIT` statement.

For example, to audit `SELECT TABLE` statements issued by the proxy application user `jackson`:

```
AUDIT SELECT TABLE BY jackson;
```

Afterward, user `jackson` can connect using the `appserve` proxy user as follows:

```
CONNECT appserve[jackson]
Enter password: password
```

The middle tier can also set the user client identity in a database session, enabling the auditing of user actions through the middle-tier application. The user client identity then shows up in the audit trail.

> **See Also:** *Oracle Database Security Guide* for detailed information about auditing in a multitier environment

Individually Auditing Schema Objects

Schema object auditing can audit all `SELECT` and DML statements permitted by object privileges, such as `SELECT` or `DELETE` statements on a particular table. The `GRANT` and `REVOKE` statements that control those privileges are also audited.

> **See Also:** *Oracle Database Security Guide* for detailed information about auditing schema objects

Auditing Network Activity

You can use the `AUDIT` statement to audit unexpected errors in network protocol or internal errors in the network layer. The types of errors uncovered by network auditing are not connection failures, but can have several other possible causes. One possible cause is an internal event set by a database engineer for testing purposes. Other causes include conflicting configuration settings for encryption, such as the network not finding the information required to create or process expected encryption.

To enable network auditing:

1. Start SQL*Plus and log on with administrative privileges, such as `SYSTEM`, or as a security administrator. For example:

   ```
   sqlplus system
   Enter password: password
   ```

   SQL*Plus starts, connects to the default database, and then displays a prompt.

   For detailed information about starting SQL*Plus, see *Oracle Database 2 Day DBA*.

2. Enter the following statement:

   ```
   AUDIT NETWORK;
   ```

   To disable network auditing, enter the following:

   ```
   NOAUDIT NETWORK;
   ```

3. Exit SQL*Plus:

   ```
   EXIT
   ```

> **See Also:** *Oracle Database Security Guide* for detailed information about auditing network activity

Tutorial: Creating a Standard Audit Trail

Suppose you wanted to audit `SELECT` statements on the `OE.CUSTOMERS` table. In this tutorial, you enable standard auditing, enable auditing for the `SELECT` SQL statement, run the `SELECT` SQL statement on the `OE.CUSTOMERS` table, and then check its audit file.

In this tutorial:

- Step 1: Log In and Enable Standard Auditing
- Step 2: Enable Auditing for SELECT Statements on the OE.CUSTOMERS Table
- Step 3: Test the Audit Settings
- Step 4: Optionally, Remove the Components for This Tutorial
- Step 5: Remove the SEC_ADMIN Security Administrator Account

Step 1: Log In and Enable Standard Auditing

First, log in, and, if necessary, enable standard auditing.

To enable standard auditing:

1. Start Database Control.
2. Log in as `SYS`.
   - **User Name**: `SYS`
   - **Password**: Enter your password.
   - **Connect As**: `SYSDBA`
3. Click **Server** to display the Server subpage.
4. In the Database Configuration section, click **Initialization Parameters**.

   The Initialization Parameters page appears.

5. Click **SPFile** to display the SPFile subpage.

   If the **SPFile** tab does not display in your installation, then you did not install Oracle Database using a server parameters file. Go to the next step.

6. In the **Name** field, enter `AUDIT_TRAIL` to find the `AUDIT_TRAIL` parameter, and then click **Go**.

   You can enter the first few characters of the parameter, for example, `AUDIT`. Alternatively, you can scroll down the list of parameters to find the `AUDIT_TRAIL` parameter. To sort the list of parameters in alphabetical order, click the **Name** column.

7. In the **Value** field, make a note of the current setting, and then change it to `DB_EXTENDED`.

   By default, the `AUDIT_TRAIL` parameter is set to `DB`, which enables database auditing and directs all audit records to the database audit trail (`SYS.AUD$`), except for records that are always written to the operating system audit trail. `DB_EXTENDED` has this functionality, plus it records SQL text and SQL bind variables.

8. Click **Apply**.

9. Restart the Oracle Database instance.

   From a command line, enter the following commands:

```
sqlplus sys as sysoper
Enter password: password

SQL> SHUTDOWN IMMEDIATE
SQL> RESTART
```

   At this point, you can check the `AUDIT_TRAIL` setting by entering the following command:

```
SQL> SHOW PARAMETER AUDIT_TRAIL

NAME                     TYPE            VALUE
------------------------ --------------- ---------------------------------\
audit_trail              string          DB_EXTENDED
```

Step 2: Enable Auditing for SELECT Statements on the OE.CUSTOMERS Table

Next, enable auditing for `SELECT` statements on the `OE.CUSTOMERS` table.

To enable auditing of SELECT statements for the OE.CUSTOMERS table:

1. Ensure that the sample user `sec_admin` exists.

   Log on as `SYSTEM`, and then from the Database Control home page, click **Server** to display the Server subpage. Select **Users** under Security, and check the list of accounts for `sec_admin`. "Step 1: Create a Security Administrator Account" on page 4-4 explains how to create the `sec_admin` security administrator account. If you still have Oracle Database Vault enabled, then you must recreate the account using the Database Vault Account Manager account.

2. In SQL*Plus, log in as user `OE` and then grant `sec_admin` the `SELECT` privilege on the `OE.CUSTOMERS` table.

```
sqlplus oe
Enter password: password
Connected.
```

```
SQL> GRANT SELECT ON CUSTOMERS TO sec_admin;
```

3. Log in to Database Control as user `SYS` with the `SYSDBA` privilege.
4. Click **Server** to display the Server subpage.
5. In the Security section, click **Audit Settings**.

   The Audit Settings page appears.
6. Select the **Audited Objects** subpage.
7. Click **Add**.

   The Add Audited Object page appears.
8. Enter the following information:
   - **Object Type**: Select `Table`.
   - **Table**: Enter `OE.CUSTOMERS`.
   - **Available Statements**: Select `SELECT`, and then click **Move** to move it to the Selected Statements list.
9. Click **OK**.
10. Log out of Database Control.

Step 3: Test the Audit Settings

At this stage, auditing is enabled and any `SELECT` statements performed on the `OE.CUSTOMERS` table are written to the to `DBA_AUDIT_TRAIL` data dictionary view. Now, you are ready to test the audit settings.

To test the audit settings:

1. Start SQL*Plus, and connect as user `sec_admin`.

   ```
   sqlplus sec_admin
   Enter password: password
   ```

2. Enter the following `SELECT` statement to create an alert in the audit trail:

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

3. Enter the following statement to view the `DBA_AUDIT_TRAIL` view:

   ```
   SELECT USERNAME, SQL_TEXT, TIMESTAMP
   FROM DBA_AUDIT_TRAIL
   WHERE SQL_TEXT LIKE 'SELECT %';
   ```

   For this `SELECT` statement, enter the text for the `SQL_TEXT` column (`'SELECT %'`) using the same case that you used when you entered the `SELECT` statement in Step 2. In other words, if you entered that `SELECT` statement in lowercase letters, then enter `'select %'` when you query the `DBA_AUDIT_TRAIL` view, not `'SELECT %'`.

   Output similar to the following appears:

   ```
   USERNAME          SQL_TEXT                                TIMESTAMP
   ----------------- --------------------------------------- ------------------
   SEC_ADMIN         SELECT COUNT(*) FROM OE.CUSTOMERS       31-MAR-10
   ```

4. Exit SQL*Plus:

   ```
   EXIT
   ```

Step 4: Optionally, Remove the Components for This Tutorial

Optionally, remove the audit settings that you created earlier.

To remove the audit settings in Database Control:

1. Log in to Database Control as user `SYS` with the `SYSDBA` privilege.
2. Click **Server** to display the Server subpage.
3. In the Security section, click **Audit Settings**.

   The Audit Settings page appears.
4. Select the **Audited Objects** subpage.
5. Under Schema, enter `OE`.
6. Under Object Name, enter `CUSTOMERS`.
7. Click **Search**.
8. Select the box next to the `OE.CUSTOMERS` audited schema, and then click **Remove**.

   A Confirmation dialog box appears.
9. Select **Yes**.
10. Click the **Database Instance** link to return to the Database home page.
11. Select the **Server** subpage, and then under Database Configuration, select **Initialization Parameters**.
12. Select the **SPFile** subpage.
13. Find the `AUDIT_TRAIL` parameter and then set it to the original value. Click **Apply**.
14. Exit Database Control.
15. Restart the database:

```
sqlplus sys as sysoper
Enter password: password

SQL> SHUTDOWN IMMEDIATE
SQL> RESTART
```

Step 5: Remove the SEC_ADMIN Security Administrator Account

This is the last example in this guide. If you no longer need the `sec_admin` administrator account, then you should remove it.

To remove the sec_admin security administrator account:

1. Log in to Database Control as user `SYSTEM`.

   If Oracle Database Vault is enabled, then you must log on as the Database Vault Account Manager.
2. In the Database Control home page, click **Server** to display the Server subpage.
3. In the Security section, click **Users**.

   The Users page appears.
4. In the **Name** field, enter `sec_admin`.
5. Click **Search**.

6. Select the selection box next to the `sec_admin` user account, and then click **Delete**.

   A Confirmation dialog box appears.

7. Select **Yes**.

8. Exit Database Control.

Guidelines for Auditing

This section contains the following topics:

- Guideline for Using Default Auditing of SQL Statements and Privileges
- Guidelines for Managing Audited Information
- Guidelines for Auditing Typical Database Activity
- Guidelines for Auditing Suspicious Database Activity

Guideline for Using Default Auditing of SQL Statements and Privileges

When you create a new database, you can enable the auditing of a select set of SQL statements and privileges. Oracle recommends that you enable default auditing. Auditing is an effective method of enforcing strong internal controls so that your site meets its regulatory compliance requirements. See "Using Default Auditing for Security-Relevant SQL Statements and Privileges" on page 7-5 for more information about default auditing.

Guidelines for Managing Audited Information

Although auditing has a minimal impact on database performance, 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 understand 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 purpose 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. 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.

Guidelines for Auditing Typical Database Activity

When your purpose for auditing is to gather historical information about particular database activities, follow these guidelines:

1. **Audit only pertinent actions.**

   To avoid cluttering meaningful information with useless audit records and to reduce the amount of audit trail administration, audit only the targeted database activities. You can audit specific actions by using fine-grained auditing. *Oracle Database Security Guide* describes fine-grained auditing in detail.

2. **Archive audit records and purge the audit trail.**

   After you collect the required information, archive the audit records of interest, and purge the audit trail of this information.

   To archive audit records, you copy the relevant records to a database table, for example, using `INSERT INTO table SELECT ... FROM SYS.AUD$ ...` for the standard audit trail. (Fine-grained audit records are in the `SYS.FGA_LOG$` table.) Alternatively, you can export the audit trail table to an operating system file. *Oracle Database Utilities* explains how to export tables by using Oracle Data Pump.

   To purge audit records, you delete standard audit records from the `SYS.AUD$` table and fine-grained audit records from the `SYS.FGA_LOG$` table. For example, to delete *all* audit records from the standard audit trail, enter the following statement:

   ```
   DELETE FROM SYS.AUD$;
   ```

   Alternatively, to delete all audit records from the standard audit trail generated as a result of auditing the table `emp`, enter the following statement:

   ```
   DELETE FROM SYS.AUD$ WHERE OBJ$NAME='EMP';
   ```

3. **Follow the privacy considerations of your company.**

   Privacy regulations often lead to additional business privacy policies. Most privacy laws require businesses to monitor access to personally identifiable information (PII), and this type of 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.

Guidelines for Auditing Suspicious Database Activity

When you audit to monitor suspicious database activity, follow these guidelines:

1. **Audit general information, and then audit specific information.**

   When you start to audit for suspicious database activity, often not much information is available to target specific users or schema objects. Therefore, set audit options more generally at first, that is, by using the standard audit options described in "Auditing General Activities Using Standard Auditing" on page 7-3.

   After you have recorded and analyzed the preliminary audit information, disable general auditing, and then audit specific actions. You can use fine-grained auditing, described in *Oracle Database Security Guide*, to audit specific actions.

Continue this process until you gather enough evidence to draw conclusions about the origin of the suspicious database activity.

2. **Protect the audit trail.**

   When auditing for suspicious database activity, protect the audit trail so that audit information cannot be added, changed, or deleted without being audited. You audit the standard audit trail by using the AUDIT SQL statement. For example:

```
sqlplus sys as sysdba
Enter password: password

SQL> AUDIT SELECT ON SYS.AUD$ BY ACCESS;
```

Initialization Parameters Used for Auditing

Table 7–1 lists initialization parameters that you can use to secure auditing.

Table 7–1 Initialization Parameters Used for Auditing

Initialization Parameter	Default Setting	Description
AUDIT_TRAIL	DB	Enables or disables auditing. See "Enabling or Disabling the Standard Audit Trail" on page 7-3 for detailed information. For a full listing of the AUDIT_TRAIL parameters and an example of setting them, see *Oracle Database Security Guide*.
AUDIT_FILE_DEST	*ORACLE_BASE*/admin/*ORACLE_SID*/adump or *ORACLE_HOME*/rdbms/audit	Specifies the operating system directory into which the audit trail is written when the AUDIT_TRAIL initialization parameter is set to OS, XML, or XML, EXTENDED. Oracle Database writes the audit records in XML format if the AUDIT_TRAIL initialization parameter is set to XML. Oracle Database also writes mandatory auditing information to this location, and if the AUDIT_SYS_OPERATIONS initialization parameter is set, writes audit records for user SYS.
AUDIT_SYS_OPERATIONS	FALSE	Enables or disables the auditing of top-level operations directly issued by user SYS, and users connecting with SYSDBA or SYSOPER privilege. Oracle Database writes the audit records to the audit trail of the operating system. If you set the AUDIT_TRAIL initialization parameter to XML or XML, EXTENDED, it writes the audit records in XML format. On UNIX systems, if you have also set the AUDIT_SYSLOG_LEVEL parameter, then it overrides the AUDIT_TRAIL parameter, which writes the SYS audit records to the system audit log using the SYSLOG utility.
AUDIT_SYSLOG_LEVEL	No default setting	On UNIX systems, writes the SYS and standard OS audit records to the system audit log using the SYSLOG utility.

To modify an initialization parameter, see "Modifying the Value of an Initialization Parameter" on page 2-6. For detailed information about initialization parameters, see *Oracle Database Reference* and *Oracle Database Administrator's Guide*.

Index

A

access control
  data encryption, 6-2
  enforcing, 5-1
  Oracle Label Security, 6-22
administrative accounts
  about, 3-2
  access, 5-2
  passwords, 3-10
  predefined, listed, 3-2
administrators
  privileges for listener.ora file, 5-2
  restricting access of, 6-41
  separation of duty, 6-41
ANONYMOUS user account, 3-2
ANY system privilege, protecting data dictionary, 2-3
APEX_PUBLIC_USER user account, 3-5
application contexts
  Oracle Virtual Private Database, used with, 6-13
ASMSNMP user account, 3-5
audit files
  archiving and purging, 7-13
  operating system file, writing to, 7-4
audit records
  types, 7-2
  viewing, 7-2
audit trail
  DB setting, 7-4
  XML file output, 7-4
auditing
  about, 7-1
  DDL statements, 7-6
  default security setting, modified by, 7-5
  DML statements, 7-6
  fine-grained auditing, 7-1
  guidelines, security, 7-12
  historical information, 7-13
  keeping information manageable, 7-12
  monitoring user actions, 7-1
  privilege audit options, 7-6
  reasons to audit, 7-2
  Sarbanes-Oxley Act requirements, 7-5
  suspicious activity, 7-13
  viewing audit records, 7-2
  where recorded, 7-2
authentication
  client, 5-1
  remote, 5-1, 5-2
  strong, 3-11
AUTHID CURRENT USER invoker's rights clause, 4-9

B

BFILE files
  restricting access, 2-5
BI user account, 3-7

C

client connections
  stolen, 5-1
client guidelines, 5-1
configuration files
  listener.ora
    administering listener remotely, 5-3
    sample, 5-3
CONNECT role, privilege available to, 4-2
connections
  securing, 5-1
  SYS user, 4-2
CREATE ANY TABLE statement, 4-2
CREATE DATABASE LINK statement, 4-2
CREATE EXTERNAL JOB privilege
  default security setting, modified by, 2-1
CREATE SESSION statement, 4-2
CREATE TABLE statement, auditing, 7-6
CTXSYS user account, 3-2

D

data definition language
  auditing, 7-6
data dictionary
  about, 2-2
  securing, 2-3
data dictionary views
  DBA_USERS, 3-11
  DBA_USERS_WITH_DEFPWD, 3-9
data files
  restricting access, 2-5
data manipulation language, auditing, 7-6

database
checking compatibility, 6-5
database accounts
See user accounts
Database Configuration Assistant
auditing by default, 7-5
default passwords, changing, 3-10
Database Control
See Oracle Enterprise Manager Database Control
DBA_USERS data dictionary view, 3-11
DBA_USERS_WITH_DEFPWD data dictionary view, 3-9
DBCA
See Database Configuration Assistant
DBSNMP user account
about, 3-2
passwords, default, 3-10
default passwords
administrative accounts, using with, 3-10
importance of changing, 3-9
default permissions, 2-4
default security settings
about, 2-1
Denial of Service (DoS) attacks
networks, addressing, 5-4
See also security attacks
DIP user account, 3-5
disabling unnecessary services, 5-4
DROP TABLE statement, auditing, 7-6

E

encryption
about, 6-2
algorithms, described, 5-7
components, 6-2
data transfer, 5-3
network, 5-5 to 5-7
network traffic, 5-4
reasons not to encrypt, 6-2
reasons to encrypt, 6-2
Enterprise Edition, 3-11
errors
checking trace files, 4-10
WHEN NO_DATA_FOUND exception example, 4-10
examples
user session information, retrieving with SYS_CONTEXT, 6-18
See also tutorials
exceptions
WHEN NO_DATA_FOUND example, 4-10
EXECUTE privilege, 4-2
EXFSYS user account, 3-3
external tables, 2-5

F

files
audit
archiving, 7-13
DoS attacks, recommendations, 7-4
configuration, 5-3
listener.ora, 5-3
restrict listener access, 5-3
restricting access, 2-5
symbolic links, restricting, 2-5
fine-grained auditing, 7-1
firewalls
Axent, 5-4
CheckPoint, 5-4
Cisco, 5-4
database server, keeping behind, 5-4
Firewall-1, 5-4
Gauntlet, 5-4
guidelines, 5-3
Network Associates, 5-4
PIX Firewall, 5-4
Raptor, 5-4
supported
packet-filtered, 5-4
proxy-enabled, 5-4
FLOWS_30000 user account, 3-6
FLOWS_FILES user account, 3-6
FTP service
disabling, 5-4

G

GRANT ALL PRIVILEGES privilege, 2-4
guidelines for security
auditing
audited information, managing, 7-12
database activity, typical, 7-13
default auditing, 7-12
client connections, 5-1
database activity, suspicious, 7-13
network connections, 5-2
operating access to database, 2-4
operating system accounts, limiting privileges, 2-4
operating system users, limiting number of, 2-4
Oracle home default permissions, disallowing modifying of, 2-4
Oracle Label Security policies, planning, 6-23
passwords
administrative, 3-10
creating, 3-8
management, enforcing, 3-11
privileges, granting, 4-1
PUBLIC user group, privileges, 4-2
roles, granting to users, 4-2
run-time facilities, granting permissions to, 2-5
symbolic links, restricting, 2-5

H

HR user account, 3-7

I

identity theft
See security attacks
initialization parameters

AUDIT_FILE_DESTINATION, 7-14
AUDIT_SYS_OPERATIONS, 7-14
AUDIT_SYSLOG_LEVEL, 7-14
AUDIT_TRAIL, 7-14
configuration related, 2-5
default security, modified by, 2-2
FAILED_LOGIN_ATTEMPTS, 3-12
installation related, 2-5
MAX_ENABLED_ROLES, 4-12
modifying, 2-6
O7_DICTIONARY_ACCESSIBILITY
about, 2-5
data dictionary, protecting, 2-3
default setting, 2-4
setting in Database Control, 2-4
OS_AUTHENT_PREFIX, 5-8
OS_ROLES, 4-12
PASSWORD_GRACE_TIME, 3-12
PASSWORD_LIFE_TIME, 3-12
PASSWORD_LOCK_TIME, 3-12
PASSWORD_REUSE_MAX, 3-12
PASSWORD_REUSE_TIME, 3-12
REMOTE_LISTENER, 5-8
REMOTE_OS_AUTHENT, 5-2, 5-8
REMOTE_OS_ROLES, 4-12, 5-8
SEC_CASE_SENSITIVE_LOGIN, 3-12
SEC_MAX_FAILED_LOGIN_ATTEMPTS, 3-12
SEC_RETURN_SERVER_RELEASE_BANNER, 2-5
SQL92_SECURITY, 4-12
invoker's rights, 4-9
IP addresses
falsifying, 5-4
IX user account, 3-7

K

Kerberos authentication
password management, 3-11

L

LBACSYS user account, 3-3
least privilege principle, 4-1
listener
not an Oracle owner, 5-3
preventing online administration, 5-2
restrict privileges, 5-3
secure administration, 5-4
listener.ora file
administering remotely, 5-3
online administration, preventing, 5-2
log files
restricting access, 2-5

M

MDDATA user account, 3-6
MDSYS user account, 3-3
MGMT_VIEW user account, 3-3
monitoring
See auditing
multiplex multiple-client network sessions, 5-4
multitier environments, auditing, 7-7
My Oracle Support
user account for logging service requests, 3-6

N

Net8 network utility
See Oracle Net
network activity
auditing, 7-7
network authentication services, 3-11
smart cards, 3-11
token cards, 3-11
X.509 certificates, 3-11
network encryption
about, 5-5
components, 5-5
configuring, 5-5
network IP addresses, 5-4
network security
Denial of Service attacks, addressing, 5-4
guidelines for clients, 5-1
nondatabase users, 6-14

O

object privileges, 4-2
OE user account, 3-7
OLAPSYS user account, 3-3
operating system access, restricting, 2-4
operating system account privileges, limiting, 2-4
operating system users, limiting number of, 2-4
operating systems
compromised, 5-1
default permissions, 2-4
Oracle Advanced Security
authentication protection, 3-11
network traffic encryption, 5-4
Oracle Connection Manager
firewall configuration, 5-4
Oracle Database Vault
about, 6-41
components, 6-41
registering with database, 6-42
regulatory compliances, how meets, 6-41
tutorial, 6-42 to 6-50
Oracle Enterprise Manager Database Control
about, 1-2
starting, 2-3
Oracle home
default permissions, disallowing modifying of, 2-4
Oracle Java Virtual Machine (OJVM), 2-5
Oracle Label Security (OLS)
about, 6-22
compared with Oracle Virtual Private Database, 6-11
components, 6-22
guidelines in planning, 6-23
how it works, 6-22
registering with Oracle Database, 6-25

tutorial, 6-24 to 6-40
used with Oracle Virtual Private Database, 6-12
Oracle Net
encrypting network traffic, 5-5
firewall support, 5-4
Oracle Virtual Private Database (VPD)
about, 6-13
advantages, 6-14
application contexts, 6-13
compared with Oracle Label Security, 6-11
components, 6-13
tutorial, 6-14 to 6-21
used with Oracle Label Security, 6-12
ORACLE_OCM user account, 3-6
ORDDATA user account, 3-3
ORDPLUGINS user account, 3-3
ORDSYS user account, 3-3
OUTLN user account, 3-4
OWBSYS user account, 3-3

P

passwords
administrative, 3-10
administrative user, 3-10
changing, 3-10
complexity, 3-11
default security setting, modified by, 2-1
default user account, 3-9
history, 3-11
length, 3-11
management, 3-11
management rules, 3-11
SYS user, 3-10
SYSTEM user, 3-10
passwords for security
requirements, 3-8
permissions
default, 2-4
run-time facilities, 2-5
PM user account, 3-7
principle of least privilege, 4-1
privileges
about, 4-1
audited when default auditing is enabled, 7-5
auditing, 7-6
CREATE DATABASE LINK statement, 4-2
system
ANY, 2-3
SYSTEM and OBJECT, 4-2
using proxies to audit, 7-7
PUBLIC user group
revoking unnecessary privileges and roles, 4-2

R

remote authentication, 5-1, 5-2
REMOTE_OS_AUTHENT initialization
parameter, 5-2
roles
CONNECT, 4-2
create your own, 4-2
job responsibility privileges only, 4-2
root file paths
for files and packages outside the database, 2-5
run-time facilities, restricting permissions, 2-5

S

Sarbanes-Oxley Act
auditing requirements, 7-5
schema objects, auditing, 7-7
SCOTT user
about, 3-7
restricting privileges of, 4-2
sec_admin example security administrator
creating, 4-4
removing, 7-11
secure application roles
about, 4-3
advantages, 4-3
components, 4-3
invoker's rights, 4-9
tutorial, 4-3 to 4-12
user environment information from SYS_
CONTEXT SQL function, 4-9
Secure Sockets Layer (SSL)
administering listener remotely, 5-2
security administrator
example of creating, 4-4
removing sec_admin, 7-11
security attacks
applications, 5-1
client connections, 5-1
Denial of Service, 5-4
eavesdropping, 5-2
falsified IP addresses, 5-1
falsified or stolen client system identities, 5-1
network connections, 5-2
security tasks, common, 1-2
SELECT ANY DICTIONARY privilege
GRANT ALL PRIVILEGES privilege, not included
in, 2-4
sensitive data
Oracle Label Security, 6-22
Oracle Virtual Private Database, 6-13
secure application roles, 4-3
separation of duty concepts, 4-4
separation-of-duty principles
about, 6-41
Oracle Database Vault, 6-46
session information, retrieving, 6-13
SH user account, 3-7
SI_INFORMTN_SCHEMA user account, 3-4
smart cards, 3-11
SPATIAL_CSW_ADMIN_USR user account, 3-6
SPATIAL_WFS_ADMIN_USR user account, 3-6
SQL statements
audited when default auditing is enabled, 7-5
auditing, 7-6
using proxies to audit, 7-7
SQL*Net network utility, 5-4
standard auditing

about, 7-3
auditing by default, 7-5
enabling or disabling audit trail, 7-3
in multitier environment, 7-7
network activity, 7-7
privileges, 7-6
proxies, 7-7
schema objects, 7-7
SQL statements, 7-6
tutorial, 7-8 to 7-12
strong authentication, 3-11
symbolic links, restricting, 2-5
SYS user account
about, 3-4
password use, 3-10
SYS_CONTEXT SQL function
example, 6-18
validating users, 4-9
SYS.AUD$ database audit trail table
about, 7-4
DB (database) option, 7-9
DB, EXTENDED option, 7-4
XML, EXTENDED option, 7-4
SYSMAN user account
about, 3-4
password use, 3-10
passwords, default, 3-10
SYS-privileged connections, 4-2
system administrator
See administrative accounts, security administrator
system identities, stolen, 5-1
system privileges, 4-2
ANY, 2-3
SYSTEM user account
about, 3-4
password use, 3-10

T

tablespaces
encrypting, 6-8
TCP ports
closing for ALL disabled services, 5-4
TCPS protocol
Secure Sockets Layer, used with, 5-2
TDE
See transparent data encryption
TELNET service, disabling, 5-4
TFTP service
disabling, 5-4
token cards, 3-11
trace files
checking for errors, 4-10
restricting access, 2-5
transparent data encryption
about, 6-3
advantages, 6-3
components, 6-3
configuring, 6-4
how it works, 6-3
performance effects, 6-4
storage space, 6-4
table columns
checking in database instances, 6-10
checking individual tables, 6-10
encrypting, 6-6
tablespaces
checking, 6-11
tablespaces, encrypting, 6-8
wallets, 6-5
troubleshooting
checking trace files, 4-10
tutorials
Oracle Database Vault, 6-42 to 6-50
Oracle Label Security, 6-24 to 6-40
Oracle Virtual Private Database, 6-14 to 6-21
secure application roles, 4-3 to 4-12
standard auditing, 7-8 to 7-12

U

UDP ports
closing for ALL disabled services, 5-4
user accounts
about, 3-1
administrative user passwords, 3-10
default, changing password, 3-9
expiring, 3-7
finding information about, 3-11
locking, 3-7
password requirements, 3-8
predefined
administrative, 3-2
non-administrative, 3-5
sample schema, 3-6
securing, 3-1 to 3-12
unlocking, 3-7
user accounts, predefined
ANONYMOUS, 3-2
APEX_PUBLIC_USER, 3-5
ASMSNMP, 3-5
BI, 3-7
CTXSYS, 3-2
DBSNMP, 3-2
DIP, 3-5
EXFSYS, 3-3
FLOWS_30000, 3-6
FLOWS_FILES, 3-6
HR, 3-7
IX, 3-7
LBACSYS, 3-3
MDDATA, 3-6
MDSYS, 3-3
MGMT_VIEW, 3-3
OE, 3-7
OLAPSYS, 3-3
ORACLE_OCM, 3-6
ORDDATA, 3-3
ORDPLUGINS, 3-3
ORDSYS, 3-3
OUTLN, 3-4
OWBSYS, 3-3

PM, 3-7
SCOTT, 3-7, 4-2
SH, 3-7
SI_INFORMTN_SCHEMA, 3-4
SPATIAL_CSW_ADMIN_USR, 3-6
SPATIAL_WFS_ADMIN_USR, 3-6
SYS, 3-4
SYSMAN, 3-4
SYSTEM, 3-4
WK_TEST, 3-4
WKPROXY, 3-5
WKSYS, 3-4
WMSYS, 3-5
XDB, 3-5
XS$NULL, 3-6
user session information, retrieving, 6-13

V

valid node checking, 5-4
views
See data dictionary views
Virtual Private Database
See Oracle Virtual Private Database
VPD
See Oracle Virtual Private Database
vulnerable run-time call, 2-5
made more secure, 2-5

W

wallets
closing, 6-6
creating, 6-4
with transparent data encryption, 6-5
WK_TEST user account, 3-4
WKPROXY user account, 3-5
WKSYS user account, 3-4
WMSYS user account, 3-5

X

X.509 certificates, 3-11
XDB user account, 3-5
XS$NULL user account, 3-6