Oracle® Database

2 Day + Security Guide

11*g* Release 1 (11.1)

B28337-01

July 2007

ORACLE®

Oracle Database 2 Day + Security Guide, 11*g* Release 1 (11.1)

B28337-01

Primary Author: Patricia Huey

Contributors: Nina Lewis, Paul Needham, Deborah Owens, Ashwini Surpur, Kamal Tbeileh, Mark Townsend, Peter Wahl, Peter M. Wong

Contents

4 Managing User Privileges

5 Securing the Network

6 Securing Data

7 Auditing Database Activity

Index

List of Tables

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.

The contents of this preface are as follows:

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

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

Accessibility of Code Examples in Documentation

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

Accessibility of Links to External Web Sites in Documentation

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

TTY Access to Oracle Support Services

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

Related Documents

For more information, 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*
- *Oracle Audit 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 already 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/`

Oracle Store

Printed documentation is available for sale in the Oracle Store at:

`http://oraclestore.oracle.com/`

OracleMetaLink

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

`http://metalink.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

As an Oracle database security administrator, you are responsible for day-to-day security tasks.

This chapter contains the following topics:

- 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 need to 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 comparable security options to those presented 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 1 (11.1) Enterprise Edition**

  Oracle Database 11*g* Release 1 (11.1) 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 1 (11.1) 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 how to secure your database, you follow these general 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
   - Why you need to encrypt passwords in Oracle Database tables

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. **Encrypt sensitive 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

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 describes how you can secure your Oracle Database installation and configuration.

This chapter contains the following topics:

- About Securing the Database Installation and Configuration
- Enabling the Default Security Settings
- Securing the Oracle Data Dictionary
- Restricting Operating System Access
- Restricting Permissions on 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

Enabling the Default Security Settings

When you create a new database or modify an existing database, you can use the Security Settings window in Database Configuration Assistant (DBCA) to enable or disable the default security settings. Oracle recommends that you enable these settings. These settings enable 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 users who need it.
- **Modifies initialization parameter settings.** Table 2–1 lists the modified initialization parameter settings.

Table 2–1 Default Security Settings for Initialization Parameters

Setting	Previous Setting	New Setting
AUDIT_TRAIL	NONE	DB
07_DICTIONARY_ACCESSIBILITY	TRUE	FALSE
PASSWORD_GRACE_TIME	UNLIMITED	7
PASSWORD_LOCK_TIME	UNLIMITED	1
PASSWORD_LOGIN_FAILURES	10	10
PASSWORD_LIFE_TIME	UNLIMITED	180
PASSWORD_REUSE_MAX	UNLIMITED	UNLIMITED
PASSWORD_REUSE_TIME	UNLIMITED	UNLIMITED
REMOTE_OS_ROLES	TRUE	FALSE

To enable the default profile security settings using Database Configuration Assistant:

1. 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***, then **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.
2. In the Welcome window, click **Next**.

   The Operations window appears.
3. Select **Configure Database Options**, and then click **Next**.

   The Database window appears.
4. Select the database that you want to configure, and then click **Next**.

   The Security Settings window appears.
5. Select the **Keep the enhanced 11g default security settings (recommended). These settings include enabling auditing and a new default password profile** option.
6. Click **Next**.

The Database Components window appears.

7. Select any additional options, and then click **Next**. Answer the remaining questions as necessary.
8. Click **Finish**.

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, default values, and so on.

This section describes the following topics:

- About the Oracle Data Dictionary
- Enabling Data Dictionary Protection

About the Oracle Data Dictionary

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

- 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
- The names of Oracle Database users
- Privileges and roles granted to each user
- Auditing information, such as who has accessed or updated various schema objects
- Other general database information

The data dictionary is structured in tables and views, just like other database data. All the data dictionary tables and views for a given database are stored in the `SYSTEM` tablespace for that database. The data dictionary central to every Oracle database, and it is an important tool for all users, from end users to application designers and database administrators.

You can use SQL statements to access the data dictionary. Because the data dictionary is read only, you can issue only queries (`SELECT` statements) against its tables and views. *Oracle Database Reference* provides a list of database views that you can query to find information about the data dictionary.

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
Connected.
SQL> SELECT * FROM DICTIONARY;
```

Enabling Data Dictionary Protection

You can protect the data dictionary by enabling the O7_DICTIONARY_ACCESSIBILITY initialization parameter. 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.

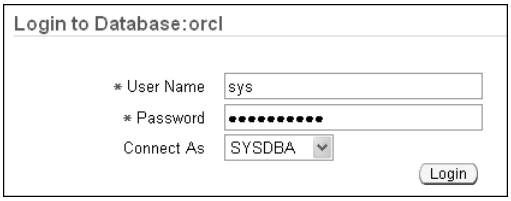
To enable data dictionary protection:

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.

   For example:



   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 **SFFile** 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**.

After you set the O7_DICTIONARY_ACCESSIBILTY parameter to FALSE, only users who have the SELECT ANY DICTIONARY privilege and those users authorized to make DBA-privileged (for example CONNECT / AS SYSDBA) connections can use the ANY system privilege on the data dictionary. If the O7_DICTIONARY_ACCESSIBILITY parameter is not set to FALSE, then any user with a DROP ANY

TABLE (for example) system privilege can drop parts of the data dictionary. However, if a user *needs* view access to the data dictionary, then you can grant that user the SELECT ANY DICTIONARY system privilege.

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 "Granting Roles to Users" on page 4-2 and *Oracle Database 2 Day DBA*.

Restricting Operating System Access

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.

Restricting Permissions on Run-Time Facilities

Many Oracle Database products use run-time facilities such as Oracle Java Virtual Machine (OJVM). Do not assign all permissions to a database run-time facility. Instead, grant specific permissions to the explicit document 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 individual files are specified:

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

Here is an example of a better (more secure) run-time call, which specifies a directory path instead:

```
call dbms_java.grant_permission('wsmith', 'SYS:java.io.FilePermission','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 2-4 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 **SFFile** 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

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.

This chapter contains the following topics:

- 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
- Changing the Default Administrative User Passwords
- Enforcing Password Management
- Initialization 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

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 for your site, 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, lock, and unlock database accounts.
- **Managing passwords.** You can manage and protect passwords by using the tools provided with Oracle Database, such as initialization parameters.

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 locks and expires 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.	Locked and expired
`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*.	Locked and expired
`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*.	Locked and expired

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*.	Locked and expired
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 6-20 and *Oracle Label Security Administrator's Guide*.	Locked and expired
MDSYS	The Oracle Spatial and Oracle Multimedia Locator administrator account. See *Oracle Spatial Developer's Guide*.	Locked and expired
MGMT_VIEW	An account used by Oracle Enterprise Manager Database Control.	Open
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*.	Locked and expired
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*.	Locked and expired
ORDSYS	The Oracle Multimedia administrator account. See *Oracle Multimedia User's Guide*, *Oracle Multimedia Reference*, and *Oracle Multimedia DICOM Developer's Guide*.	Locked and expired
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*.	Locked and expired
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*.	Locked and expired

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

User Account	Description	Status After Installation
SYS	An account used to perform database administration tasks. See *Oracle Database 2 Day DBA*.	Open
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
SYSTEM	An account used to perform database administration tasks. See *Oracle Database 2 Day DBA*.	Open
WMSYS	The account used to store Ultra Search system dictionaries and PL/SQL packages. 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*.	Locked and expired
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*.	Locked and expired

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*.	Locked and expired
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*.	Open
FLOWS_020200	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*.	Locked and expired
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*.	Locked and expired
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*.	Locked and expired
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 Oracle*MetaLink*. Then when you log a service request, it is associated with the database instance configuration information. See *Oracle Database Installation Guide* for your platform.	Locked and expired
PUBLIC	Account used for the PUBLIC user group. Oracle Universal Installer does not lock or expire this account upon installation. Its status is OPEN. See *Oracle Database Security Guide*.	Locked and expired

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

User Account	Description	Status After Installation
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*.	Locked and expired
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*.	Locked and expired
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.	Locked and expired

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*.	Locked and expired
HR	The account used to manage the HR (Human Resources) schema. This schema stores information about the employees and the facilities of the company.	Locked and expired
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.	Locked and expired
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.	Locked and expired
IX	The account used to manage the IX (Information Exchange) schema. This schema manages shipping through business-to-business (B2B) applications.	Locked and expired
SH	The account used to manage the SH (Sales) schema. This schema stores business statistics to facilitate business decisions.	Locked and expired

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

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.

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, as well as 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.

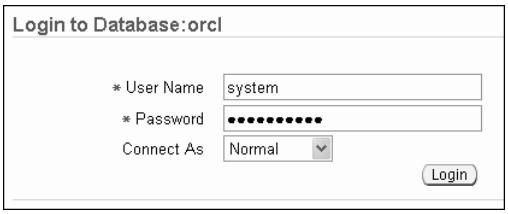
To expire and lock a database account:

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.

Oracle Database Security Guide provides guidelines for securing passwords. At a minimum, passwords must meet the following requirements:

- The password contains no fewer than eight characters.
- The password is not the same as the user name, nor is it the user name spelled backward or with numeric characters appended.
- The password is not the same as the server name or the server name with the numbers 1–100 appended.
- The password is not too simple, for example, `welcome1`, `database1`, `account1`, `user1234`, `password1`, `oracle`, `oracle123`, `computer1`, `abcdefg1`, or `change_on_install`.
- The password includes at least 1 numeric and 1 alphabetic character.
- The password differs from the previous password by at least 3 letters.

See Also:

- "Finding and Changing Default Passwords" on page 3-8 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
- *Oracle Database Security Guide* for additional guidelines on choosing secure passwords

Finding and Changing Default Passwords

In Oracle Database 11*g* Release 1 (11.1), database user accounts, including administrative accounts, are installed without default passwords. During installation, you either create a password the account (always an administrative account), or Oracle Database installs the default accounts, such as those in the sample schemas, locked with their passwords expired.

If you have upgraded from a previous release of Oracle Database, you may have database accounts that have default passwords. These are default accounts that are created when you create a database, such as the `HR`, `OE`, and `SCOTT` accounts.

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_WTIH_DEFWD;
   ```

   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.

   ```
   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.

   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

Changing 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.

Similarly, for production environments, do not use default passwords for any administrative accounts, including SYSMAN and DBSNMP. Oracle Database 11*g* Release 1 (11.1) 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 3-8.

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.

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 3-11 lists initialization parameters that you can set to enforce password management.

You can find information about user accounts by querying the DBA_USERS view. This view contains a column for passwords, but for stronger security, Oracle Database encrypts (disguises) the data in this column. The DBA_USERS view provides useful information such as the user account status, whether or not the account is locked, and password versions. You can query DBA_USERS as follows:

```
sqlplus SYSTEM
Enter password: password
Connected.
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

Initialization Parameters Used to Secure User Accounts

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

Table 3–4 Initialization Parameters Used for User Account Security

Initialization 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 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	Sets the number of days before which a password cannot be reused.
PASSWORD_REUSE_TIME	UNLIMITED	Sets the number of password changes required before the current password can be reused.

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

A **privilege** refers to the rights of a user to perform an action. This chapter describes how to manage user privileges.

This chapter contains the following topics:

- About Privilege Management
- Granting Necessary Privileges Only
- Revoking Privileges from the PUBLIC User Group
- Granting Roles to Users
- 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 authenticate users based on conditions defined in a PL/SQL package. For example, a secure application role can be used to check the session ID of a user before being allowed to log in to an application.

Granting Necessary Privileges Only

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.

Revoking Privileges from 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.

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.

For example, it is imperative to strictly limit the privileges of user SCOTT, because this is a well known default user account that may be vulnerable to intruders. Because the CREATE DBLINK privilege allows access from one database to another, drop its privilege for SCOTT. Then, drop the entire role for the user, because privileges acquired through a role cannot be dropped individually. Recreate your own role with only the privileges needed, and grant that new role to that user. Similarly, for even better security, drop the CREATE DBLINK privilege from all users who do not require it.

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 includes the following topics:

- About Secure Application Roles
- Example: 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 security administrator.

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.

  Do not grant the role directly to the user; the PL/SQL package will do that for you. However, if the policy for your site is to grant roles to users, you can grant the secure application role to the user if you alter the user account to not have any default roles. For example:

  ```
  ALTER USER psmith DEFAULT ROLE NONE;
  ```

- **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. Invoker's rights enable the user to have `EXECUTE` privileges on all objects that the package accesses. An invoker's right procedure executes with the privileges of the current user, that is, the user who invokes the procedure. These procedures are not bound to a particular schema. They can be run by a variety of users and 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 `DBMS_SESSION.SET_ROLE` call to enable (or disable) the role for the user.

  After you create the PL/SQL package, you need to grant the appropriate users `EXECUTE` privileges on the package.

- **A way to execute the PL/SQL package when the user logs on.** You can 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. You can include auditing checks in the security policy to record this information, which is a way to track potential intruders.

Example: Creating a Secure Application Role

This example 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`.

Follow these steps to complete this example:

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

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 examples 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**.

   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**: `fussy2all`
- **Default Tablespace**: `SYSTEM`
- **Temporary Tablespace**: `TEMP`
- **Status**: `Unlocked`

7. Click **System Privileges** to display the System Privileges subpage.
8. Click **Edit List**.

   The Modify System Privileges page appears.
9. 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`
   - `DROP ANY PROCEDURE`
   - `DROP ANY ROLE`
   - `SELECT ANY DICTIONARY`
10. Click **OK**.
11. Under Admin Option, do not select the check boxes.
12. Click **OK**.

Step 2: Create User Accounts for This Example

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 Database Control, select the **Database Instance** link to display the Database Home page.

   If you are not logged in to Database Control, see *Oracle Database 2 Day DBA* for instructions about how to start Database Control. In the Login page, enter an administrator user name (for example, `SYSTEM`) and password, and then click **Login**.
2. Click **Server** to display the Server subpage.
3. Under Security, select **Users**.

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

   The Create User page appears.
5. Enter the following information:

- **Name**: `mweiss` (to create the user account for Matthew Weiss)
- **Profile**: `DEFAULT`
- **Authentication**: `Password`
- **Enter Password** and **Confirm Password**: `mw2work_now`
- **Default Tablespace**: `USERS`
- **Temporary Tablespace**: `TEMP`
- **Status**: `Unlocked`

6. Click **System Privileges** to display the System Privileges subpage.
7. Click **Edit List**.

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

   The Create User page appears, with `CREATE SESSION` listed as the system privilege for user `mweiss`.
10. Ensure that the Admin Option for `CREATE SESSION` is not selected, and then click **OK**.

    The Users page appears.
11. Select **MWEISS** from the list of users, and then from the **Actions** list, select **Create Like**. Then, click **Go**.
12. 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**: `wt4today_always`
13. Click **OK**.

    You do not need to grant `wtaylor` the `CREATE SESSION` privilege, because the **Create Like** action has done of this for you.
14. Exit 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 need to 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: fussy2all
   ```

   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. The password `ready2go` is an example, but you can 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: sys_password
ALTER USER OE ACCOUNT UNLOCK IDENTIFIED BY ready2go;
CONNECT oe
Enter password: ready2go
```

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. If your site requires that you directly grant users the role, then you *must* disable the role for that user. This is because the role needs to be initially disabled before the security policies in the package can begin performing their checks. For example, to disable the role for user `wsmith` (assuming `wsmith` was granted the role in the first place), enter the following statement:

```
ALTER USER wsmith DEFAULT ROLE NONE;
```

Step 4: Create a Lookup Table

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. To get around this problem, you can create a lookup table that only contains the employee names, employee IDs, and their manager IDs.

To create the HR.HR_VERIFY lookup table:

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. 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: sys_password
ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password;
CONNECT hr
Enter password: password
```

2. Enter the following CREATE TABLE SQL statement to create the lookup table:

```
CREATE table hr_verify AS
SELECT employee_id, first_name, last_name, email, manager_id
FROM employees;
/
```

3. Grant EXECUTE privileges for this table to mweiss and wtaylor 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 Package 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 example 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, whose password is fussy2all.

```
CONNECT sec_admin
Enter password: fussy2all
```

2. Enter the following CREATE PROCEDURE statement to create the secure application role procedure:

```
SQL> 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     DBMS_SESSION.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 the write 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` table for the manager ID of the current user.

- **Lines 9–13**: Use an `IF` condition to test whether or not 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 using `DBMS_SESSION.SET_ROLE`. Otherwise, its `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 into a buffer so that it is readily available.

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

At this stage, Matthew and Winston can try to access the `OE.ORDERS` table, but they are not able to, even if they should. The next step is to grant them `EXECUTE` privileges 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`, whose password is `mw2work_now`.

```
CONNECT mweiss
Enter password: mw2work_now
```

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;
```

   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`, whose password is `wt4today_always`.

```
CONNECT wtaylor
Enter password: wt4today_always
```

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 Example

Remove the components that you created for this example.

To remove the components:

1. In SQL*Plus, connect as SYSTEM.

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

2. As user SYSTEM, enter the following DROP statements:

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

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

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

```
CONNECT sec_admin
Enter password: fussy2all
```

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 table.

```
CONNECT HR
Enter password: hr
DROP TABLE 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 2-4 for more information about this parameter.
OS_ROLES	FALSE	Determines whether Oracle or the operating system identifies and manages the roles of each user name.
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 or not 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 or not 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 describes how you can secure the network for Oracle Database.

This chapter contains the following topics:

- 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 improve security for the client connection to ensure thorough protection. Using SSL is an essential element in these lists, enabling strict security for authentication and communications.

These guidelines are as follows:

- Guidelines for Securing Client Connections
- Securing the Network Connection
- Securing a Secure Sockets Layer Connection

Guidelines for Securing Client Connections

Because authenticating client computers is problematic over the Internet, 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 clients connecting to an Oracle database.

   Do not alter the default setting of the `REMOTE_OS_AUTHENT` initialization parameter, which is `FALSE`.

   Setting this parameter to `FALSE` does not mean that users cannot connect remotely. It means that the database will not trust that the client has already authenticated, and will apply its standard authentication processes.

2. **Configure the connection to use Secure Sockets Layer (SSL).**

   Using SSL communication makes eavesdropping difficult and enables the use of certificates for user and server authentication. To learn how to configure SSL, see *Oracle Database Advanced Security Administrator's Guide*.

3. **Set up certificate authentication for clients and servers.**

   See *Oracle Database Advanced Security Administrator's Guide* for more information about ways to manage certificates.

4. **Monitor the users who access your systems.**

   Authenticating client computers over the Internet is problematic. Perform user authentication instead, which avoids client system issues that include falsified IP addresses, hacked operating systems or applications, and falsified or stolen client system identities. The following steps improve client computer security:

   a. Configure the connection to use Secure Sockets Layer (SSL). Using SSL communication makes eavesdropping unfruitful, and enables the use of certificates for user and server authentication. To learn how to configure SSL, see *Oracle Database Advanced Security Administrator's Guide*.

   b. Set up certificate authentication for clients and servers so that:

      – The organization is identified by unit and certificate issuer, and the user is identified by distinguished name and certificate issuer.

      – Applications test for expired certificates.

      – Certificate revocation lists are audited.

See *Oracle Database Advanced Security Administrator's Guide* for more information about ways to manage certificates.

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 that impinge on each path and node. Then, take steps to lessen or eliminate those threats and the consequences of a breach of security. In addition, monitor and audit to detect either increased threat levels or successful penetration.

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. **Use Secure Sockets Layer (SSL) when administering the listener.**

   See "Securing a Secure Sockets Layer Connection" on page 5-5 for more information.

2. **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.

3. **Prevent online administration by requiring the administrator to have write privileges on the listener.ora file and the listener password:**

   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 = ed-pdsun1.us.oracle.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 = ed-pdsun1.us.oracle.com)
         (PORT = 8281))
       )
     )
   ```

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

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

   Ensure that the password has not been set in the `listener.ora` file. The local operating system authentication secures the listener administration. The remote listener administration is disabled when the password has not been set.

5. **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.

6. **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*.

7. **Because you cannot protect physical addresses when transferring data over the Internet, use encryption when this data needs to be secure.**

   See "Protecting Data on the Network by Using Network Encryption" on page 5-7 to learn about how to protect Oracle data over the network. *Oracle Database Advanced Security Administrator's Guide* describes network encryption in detail.

8. **Use a firewall.**

   Appropriately placed and configured firewalls can prevent outside access to your intranet when you allow internal users to have Internet access.

   - 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.)

9. **Prevent unauthorized administration of the Oracle listener.**

   Establish a well-formed password for the Oracle listener to prevent remote configuration of the Oracle listener. See "Requirements for Creating Passwords" on page 3-8 for advice on the best types of passwords to choose. For more information about the listener, see *Oracle Database Net Services Administrator's Guide*.

10. **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.

11. **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 5-7. For detailed information about network encryption, see *Oracle Database Advanced Security Administrator's Guide*.

12. **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.

Securing a Secure Sockets Layer Connection

Secure Sockets Layer (SSL) is the Internet standard protocol for secure communication, providing mechanisms for data integrity and data encryption. These mechanisms can protect the messages sent and received by you or by applications and servers, supporting secure authentication, authorization, and messaging through certificates and, if necessary, encryption. Good security practices maximize protection and minimize gaps or disclosures that threaten security. The following list illustrates the cautious attention to detail necessary for the successful use of SSL. For detailed information about Oracle SSL configuration, see *Oracle Database Advanced Security Administrator's Guide*.

1. **Ensure that configuration files (for example, as for clients and listeners) use the correct port for SSL, which is the port configured upon installation.**

   You can run HTTPS on any port, but the standards specify port 443, where any HTTPS-compliant browser looks by default. The port can also be specified in the URL, for example,

   ```
   https://secure.server.com:4445/
   ```

   If a firewall is in use, then it too must use the same ports for secure (SSL) communication.

2. **Ensure that TCPS is specified as the PROTOCOL in the ADDRESS parameter in the tnsnames.ora file (typically on the client or in the LDAP directory).**

   An identical specification must appear in the `listener.ora` file (typically in the `$ORACLE_HOME/network/admin` directory).

3. **Ensure that the SSL mode is consistent for both ends of every communication. For example, the database (on one side) and the user or application (on the other) must have the same SSL mode.**

   The mode can specify either client or server authentication (one-way), both client and server authentication (two-way), or no authentication.

4. **Ensure that the server supports the client cipher suites and the certificate key algorithm in use.**

5. **Enable DN matching for both the server and client. This prevents the server from falsifying its identity to the client during connections.**

   This setting ensures that the server identity is correct by matching its global database name against the DN from the server certificate.

   You can enable DN matching in the `tnsnames.ora` file. For example:

   ```
   set:SSL_SERVER_CERT_DN="cn=finance,cn=OracleContext,c=us,o=acme"
   ```

   Otherwise, a client application would not check the server certificate, which could allow the server to falsify its identity.

6. **Do not remove the encryption from your RSA private key inside your server.key file, which requires that you enter your pass phrase to read and parse this file.**

   Note: A server without SSL does not require a pass phrase.

   If you decide your server is secure enough, you can remove the encryption from the RSA private key while preserving the original file. This enables system startup scripts to start the database server, because no pass phrase is needed. Ideally, restrict permissions to the root user only, and have the Web server start as `root`, but then run as another user. Otherwise, anyone who gets this key can impersonate you on the Internet or decrypt the data that was sent to the server.

   See Also:

   - *Oracle Database Advanced Security Administrator's Guide* for general SSL information, including configuration
   - *Oracle Database Net Services Reference* for TCP-related parameters in `sqlnet.ora`

Protecting Data on the Network by Using Network Encryption

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

This section explores the following topics:

- 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 even though you have carefully encrypted it in the database. 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 disguise 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 need to 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

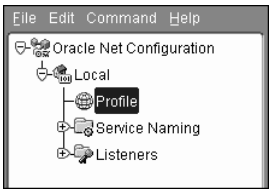
You can configure network encryption by using either Oracle Net Manager or by editing the `sqlnet.ora` file. This guide explains how to use Oracle Net Manager to configure network encryption.

To configure network encryption:

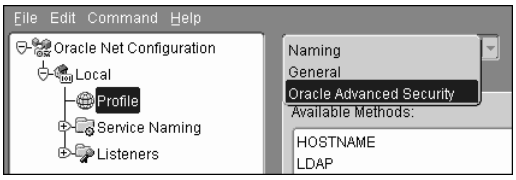
1. On the server computer, start Oracle Net Manager.
   - **UNIX**: From `$ORACLE_HOME/bin`, enter the following at the command line:

     ```
     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**.

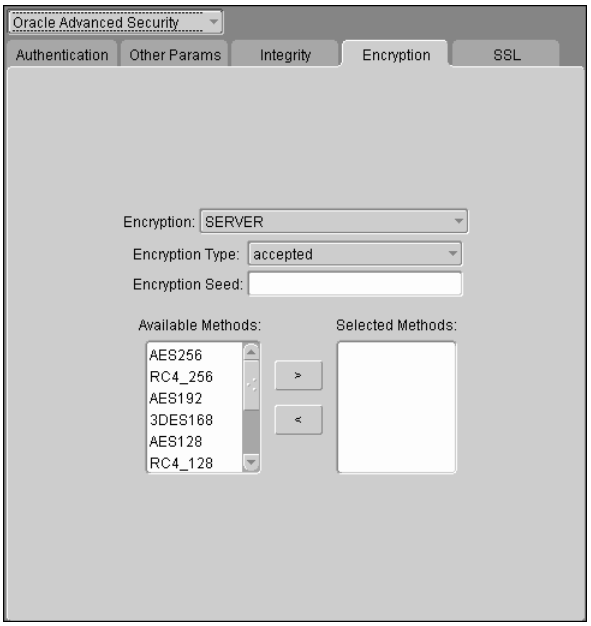


3. From the list, select **Oracle Advanced Security**.



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 other side; it will otherwise be inactive.
     - **rejected**: Service must not be active, and the connection will fail if the other side requires.
     - **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; it will otherwise be 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.
  - **DES40**: Enables you to use DES to encrypt up to 40 bits of data.

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 authenticate 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. The default value of this parameter is OPS$ for backward compatibility with previous versions. However, you can set the prefix value to "" (a null string), thereby eliminating the addition of any prefix to operating system account names.
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 describes three ways that you can secure data: by using transparent data encryption, Oracle Virtual Private Database, and Oracle Label Security.

This chapter contains the following topics:

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

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 (disguises) data in one or more database table columns, or it can encrypt an entire tablespace. This method 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 5-7 explains how.

- **Oracle Virtual Private Database (VPD).** This feature restricts data access by creating a policy that enforces a predicate `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 the needs of your site. You then create a security authorization for users based on the OLS labels.

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 explores the following topics:

- 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

Encryption is the practice of disguising (encrypting) data in a way that only its recipient can undisguise (decrypt) and read. You use encryption to protect data in a potentially unprotected environment, such as the network.

Encrypted data has the following components:

- **An algorithm to encrypt the data.** The encryption algorithm is a formula that Oracle Database uses to disguise data. It translates the clear text (that is, human-readable) version of the data into a format that only can be undisguised by another algorithm to decrypt the data. Oracle Database supports several industry-standard encryption and hashing algorithms, including the Advanced Encryption Standard (AES) encryption algorithm. AES has been approved by the National Institute of Standards and Technology (NIST) to replace the Data Encryption Standard (DES).
- **An algorithm to decrypt the data.** The decryption algorithm performs the task of the encryption algorithm in reverse: it takes the disguised data and translates it back into clear text.
- **A key to encrypt the data for the sender and to decrypt the data for the receiver.** The encryption key determines whether encrypted data is decrypted. When you encrypt data, Oracle Database uses the encryption key to apply the encryption algorithm to the data. Conversely, when you decrypt data, the encryption key applies the decryption algorithm to the 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.

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 applications from database administrators. You can get around this problem, and address it more efficiently, by using Oracle Database Vault to control the access your database administrators have to sensitive data.

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. When you encrypt an entire tablespace, the amount of storage space increases significantly.

> **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. After the selection, the data is reencrypted.

Transparent data encryption helps protect data stored on media in case the storage media or data file gets stolen, because it stores the encryption keys in a security module (that is, a wallet) external to the database. Protecting data from this type of theft is required for most compliance regulations. The benefit to using transparent data encryption is that it requires little coding and is quick and easy to implement.

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

- **A wallet to store the encryption key.** The wallet is a storage space in the form of a binary file. This file is created outside the database and is accessible only to the security administrator. For this external security module, Oracle Database uses an Oracle wallet as described in this section. Storing the master encryption key in this way prevents unauthorized use. To create the wallet, you can use the `ALTER SYSTEM` SQL statement, which enables you to specify the wallet password. The encryption key to open the wallet has an associated password and encryption algorithm. After you create the wallet, you need to open the wallet, which you can do in Database Control or in SQL*Plus.
- **A location for the wallet.** You can specify the wallet location by modifying the `sqlnet.ora` file.
- **A mechanism for encrypting the data.** You can use SQL*Plus to designate one or more columns or the tablespace to encrypt. If you decide that the data does not need to be encrypted, you can decrypt it in SQL*Plus.

Afterward, when a user enters data into an encrypted column, Oracle Database performs the following steps:

1. Retrieves the master key from the wallet.
2. Decrypts the encryption key of the table from the data dictionary.
3. Uses the encryption key to encrypt the data the user entered into the encrypted column.
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, displays it in clear text format, and then reencrypts it afterward.

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 gets stolen.
- Implementing transparent data encryption helps you address security-related regulatory compliance issues.
- You do not need to create triggers or views to decrypt data. Data from tables is transparently decrypted for the database user.
- 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 affects performance only when data is retrieved from or inserted into an encrypted column. No reduction in performance occurs 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.

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

   For example, create a directory called `ORA_WALLETS` in the `C:\oracle\product\11.1.0\db_1` 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\11.1.0\db_1\ORA_WALLETS)))
```

4. Save and close the `sqlnet.ora` file.
5. Start SQL*Plus and then log on as `SYS`, connecting `AS SYSOPER`.

```
SQLPLUS "SYS/AS SYSOPER"
```

```
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*.

6. Enter the following SQL statements to shut down and then restart the database:

```
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 that might be 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 who knows the wallet password.

To create the wallet:

1. In SQL*Plus, connect as a user with administrative privileges, such as `SYSTEM`, 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 assign to the encryption key:

```
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 will 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 assigned to the encryption key:

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

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. However, the unencrypted data is still available. 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:

```
ALTER SYSTEM SET ENCRYPTION WALLET CLOSE;
```

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 private standards used by companies such as MasterCard or VISA. 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 6-5.

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`**.

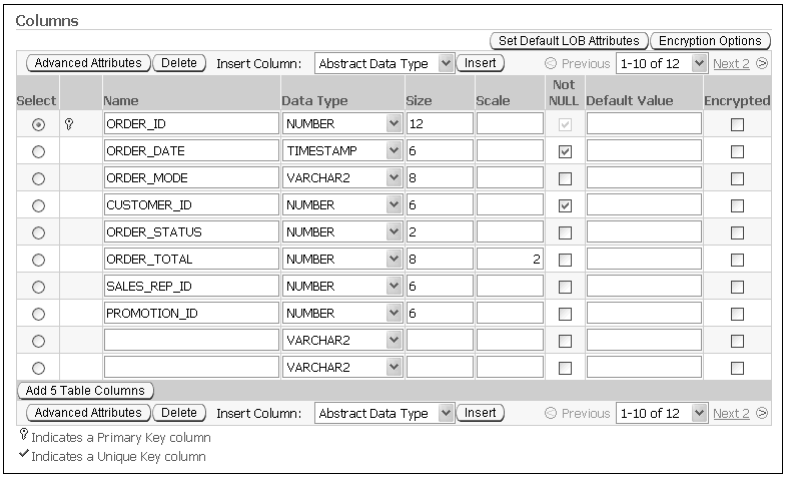
   The Tables page appears.

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 any indexed columns or columns that use a foreign key restraint (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 repetition of text in the clear 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 check box under **Encrypted**.

8. Click **Continue**.

The Create Table (or Edit Table) page appears.

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. When a table is updated, read access is still possible. 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 6-5.

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** check 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 5-7 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 the following topics:

- 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, run 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\11.1.0\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;
  ```

  The tables in the current database instance that contain encrypted columns are listed. 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$ENCRYPTED_TABLESPACE	Displays information about the tablespaces that are encrypted. For example:
	`SELECT * FROM V$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.

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

Controlling Data Access with Oracle Virtual Private Database

Oracle Virtual Private Database (VPD) enables you to dynamically embed 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 credentials of a user.

This section contains the following topics:

- About Oracle Virtual Private Database
- Example: 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 credentials pass 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. 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 Database Control or 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.

Example: 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 have been created, but other employees should not. In this example, 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 example, 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 ID of the customer.

You may want to store VPD policies in a database account separate from the database administrator and from application accounts. In this example, you will use the sec_admin account, which was created in "Example: Creating a Secure Application Role" on page 4-4, 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 6-20 for more information.

Follow these steps to complete this example:

- Step 1: If Necessary, Create the Security Administrator Account
- Step 2: Update the Security Administrator Account

- Step 3: Create User Accounts for This Example
- 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 Example

Step 1: If Necessary, Create the Security Administrator Account

In "Example: Creating a Secure Application Role" on page 4-4, you created a security administrator account called `sec_admin` for that example. You can use that account for this example. 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 needs to 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 **SEC_ADMIN** and then 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. Select **Table Objects**, and then 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.
15. Click **Apply**.

Step 3: Create User Accounts for This Example

You are ready to create accounts for the employees who need to 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**: `too_much2do`
   - **Default Tablespace**: `USERS`
   - **Temporary Tablespace**: `TEMP`
   - **Status**: `Unlocked`
4. Click **OK**.

   The Users page appears, with `LDORAN` listed as a new user.
5. Select **LDORAN** from the Users page.

   The Edit User page appears.
6. Select **Object Privileges** to display 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. In the **Select Table Objects** field, enter the following text:

   ```
   OE.ORDERS
   ```

   Do not include spaces in this text.
9. 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.
10. Click **Apply**.

11. Select **LDORAN**, and from the **Actions** list, select **Create Like**. Then, click **Go**.

    The Create User page appears.

12. Enter the following information:

    - **Name**: `LPOPP` (to create the user account for Finance Manager Luis Popp.)
    - **Enter Password** and **Confirm Password**: `shop2drop`

13. Click **OK**.

Both employee accounts have been created, and they have identical privileges. If either 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`, whose password is `fussy2all`.
3. Click **Schema** to display the Schema subpage.
4. Under Programs, select **Functions**.

   The Functions page appears.

5. Click **`Create`**.

   The Create Function page appears.

6. Enter the following information:

   - **Name**: `F_POLICY_ORDERS`
   - **Schema**: `SEC_ADMIN`
   - **Source**: 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.

     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 need to 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. The `INSERT`, `UPDATE`, and `DELETE` statements are impossible to use, because the user was not granted permissions.

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 Policies**.

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

   The Create Policy page appears.
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 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** check 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 check box to return a predicate with a length of up to 32K bytes. By not selecting this check box, Oracle Database limits the predicate to 4000 bytes.
7. Under Enforcement, select **SELECT**.
8. Click **OK**.

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: too_much2do
   ```

   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: shop2drop
   ```

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.

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

5. Exit SQL*Plus:

   ```
   EXIT
   ```

Step 7: Optionally, Remove the Components for This Example

After completing this example, 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`, whose password is `fussy2all`.
2. Click **Server** to display the Server subpage.
3. Under Security, select **`Virtual Private Database Policies`**.

   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 who created the user accounts and roles used in this example.
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 examples 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 includes the following topics:

- About Oracle Label Security
- Guidelines for Planning an Oracle Label Security Policy
- Example: 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.

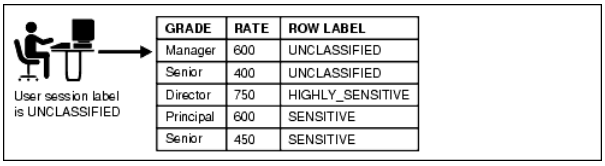
After you create and assign the labels, you can use Oracle Label Security to assign specific users authorization for specific rows, based on these labels. Afterward, Oracle Label Security automatically compares the label of the data row with the security clearance of the user to determine whether or not 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 to 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 or not 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 need to 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.

Example: Applying Security Labels to the HR.LOCATIONS Table

This example 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 need to restrict user access by focusing on user privileges, or some other method such

as the job title the user has in your organization, 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 6-11 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

Follow these steps to complete this example:

- Step 1: Install Oracle Label Security and Enable User LBACSYS
- Step 2: Create a Role and Three Users for the Oracle Label Security Example
- 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 Example

Step 1: Install Oracle Label Security and Enable User LBACSYS

In a default Oracle Database installation, Oracle Label Security is not installed, but it is part of the products available in Oracle Database. You can install it in an existing database by using Oracle Universal Installer, and then Database Configuration Assistant (DBCA) to register it. Oracle Label Security provides its own user account, LBACSYS, which you will need to enable after the installation.

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

Installing Oracle Label Security

This procedure explains how to install Oracle Label Security in an existing database.

To install Oracle Label Security:

1. Shut down the database instance in which you plan to install Oracle Label Security.

   Log in to SQL*Plus as `SYS`, connecting with the `SYSDBA` privilege. At the SQL prompt, enter the following command:

   ```
   SHUTDOWN IMMEDIATE
   ```

2. Exit SQL*Plus.

   ```
   EXIT
   ```

3. Stop the Oracle Database processes.

   - **UNIX**: Go to the `$ORACLE_HOME/bin` directory and run the following commands to stop the Database Console and the listener:

     ```
     ./emctl stop dbconsole
     ./lsnrctl stop
     ```

   - **Windows**: In the Windows Services tool, right-click the Oracle listener, console, and database service services, and then from the menu, select **Stop**. The names of these services begin with Oracle and include the name of the database instance. For example, assuming the database instance is `orcl`, the names would be similar to the following:
     - OracleDBConsoleorcl
     - OracleJobSchedulerORCL
     - OracleOraDB1g-home1TNSListener
     - OracleServiceORCL

4. Run Oracle Universal Installer from the installation media.

   - **UNIX**: Use the following command:

     ```
     /mnt/cdrom/runInstaller
     ```

   - **Windows**: Double-click the file, `setup.exe`, on the installation media.

5. Select **Advanced Installation**, and then click **Next**.

   The Select Installation Type window appears.

6. Select **Custom**, and then click **Next**.

   The Specify Home Details screen appears.

7. Select the Oracle base directory and the Oracle home directory in which you want to install Oracle Label Security. Click **Next**.

   (By default, Oracle Universal Installer offers to create a new Oracle home for you, so ensure that you select the correct existing Oracle home.) Oracle Universal Installer verifies that your system meets the minimum requirements. Next, the Available Product Components window is displayed.

8. Select the check box corresponding to **Oracle Label Security**.

   You can find this option under Oracle Database 11g, Enterprise Edition Options. Click **Next**.

   The Summary window is displayed.

9. Review your choices and then click **Install**.

The progress window is displayed. When the installation completes, Oracle Universal Installer displays the End of Installation window.

10. Click **Exit**, and then click **Yes** to confirm the exit.
11. Restart the services and the database instance in which you installed Oracle Label Security.
    - **UNIX**: Go to the `$ORACLE_HOME/bin` directory and run the following commands to start the Database Console and the listener:

      ```
      ./emctl start dbconsole
      ./lsnrctl start
      ```

      Start SQL*Plus and then restart the database instance:

      ```
      sqlplus "sys/as sysoper"
      Enter password: password
      Connected to an idle instance
      SQL> STARTUP
      ```

    - **Windows**: In the Windows Services tool, right-click the Oracle listener, console, and database service services, and then from the menu, select **Start**. The names of these services begin with Oracle and include the name of the database instance. For example, assuming the database instance is `orcl`, the names would be similar to the following:
      - OracleDBConsoleorcl
      - OracleJobSchedulerORCL (Optional; you do not need to start it for the examples in this guide.)
      - OracleOraDB1g-home1TNSListener
      - OracleServiceORCL (This service starts when you start OracleDBConsole.)

Registering Oracle Label Security with Oracle Database

After you complete the installation, you must register Oracle Label Security with Oracle Database.

To register Oracle Label Security with Oracle Database:

1. 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.
2. In the Welcome page, click **Next**.

   The Operations page appears.

3. Select **Configure Database Options**, and then click **Next**.

   The Database page appears.

4. From the list, select the database where you installed Oracle Label Security and then click **Next**.

   The Management Options page appears.

5. Select **Keep the database configured with Database Control**.

   The Security Settings page appears.

6. Select the security option you prefer, and then click **Next**.

   Oracle recommends that you take advantage of the enhanced security settings for this release.

   The Database Components page appears.

7. Select **Oracle Label Security**, and then click **Next**.

   The Connection Mode page appears.

8. 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.

9. Exit Database Configuration Assistant.

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 unlock LBACSYS, create a new password, and grant it SELECT ANY DICTIONARY privileges:

1. Log in to Database Control as the user `SYSTEM`.

   In the Login page, enter `SYSTEM` and the password assigned to `SYSTEM`. Set **Connect As** to **Normal**. Select **Login** to log in.

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

3. Under Users & Privileges, select **`Users`**.

   The Users page appears.

4. Select user `LBACSYS`.

   To quickly find `LBACSYS`, enter `lba` in the **Object Name** field, and then click **Go**.

5. With `LBACSYS` selected, click **Edit**.

   The Edit User page appears.

6. Next to Status, select **Unlocked**.

7. In the **Enter Password** and **Confirm Password** fields, enter `steplively2day` to create the password.
8. Click **System Privileges** to display the Edit User: LBACSYS page.
9. Click **Edit List**.

   The Modify System Privileges page appears.
10. 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**.
11. Click **Apply**.

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

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. 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 **Schema** to display the Schema subpage.
3. In the Users & Privileges 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**: kingpin2all
   - **Default Tablespace**: USERS
   - **Temporary Tablespace**: TEMP
   - **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** check 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**.

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, with eager2please as the password for kpartner and too_much2do as the password for ldoran.

   Create their names and passwords. 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 SELECT privileges on the HR.LOCATIONS table.

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, enter `LBACSYS` and `steplively2day` for the password. 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 check box so that the label column will not be hidden. (It should be deselected by default.)

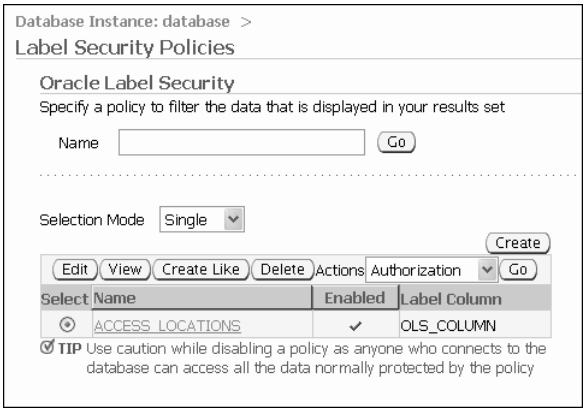
     Usually, the label column is hidden, but during the development phase, you may want to have it visible so that you can check it. After the policy is created and working, hide this column so that it is transparent to applications.
   - **Enabled**: Select this check box to enable the policy. (It should be enabled by default.)
   - **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 need to 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 need to 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 **ACCESS_LOCATIONS** policy.

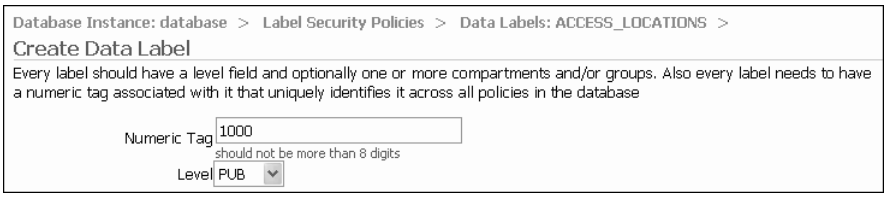
3. In the Actions list, select **Data Labels**, and then click **Go**.

   The Data Labels page appears.

4. Click **Add**.

The Create Data Label page appears.

5. Enter the following information:
   - **Numeric Tag**: Enter `1000`.
   - **Level**: From the list, select **PUB**. (To use the keyboard to select an item, enter the first letter of its name. For example, enter `P` to select **PUB**.)

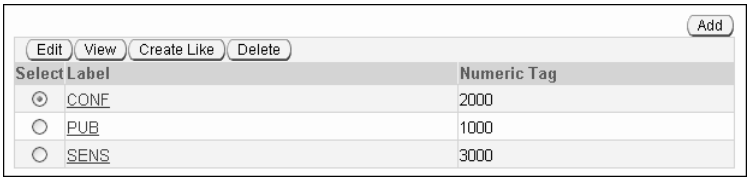


6. Click **OK**.

   The data label appears in the Data Labels page.

7. Click **Add** again, and then create a data label for the `CONF` level. For the numeric tag, enter `2000`.
8. Click **OK**.
9. Click **Add** again, and then create a data label for the `SENS` level. For the numeric tag, enter `3000`.
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 `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**.

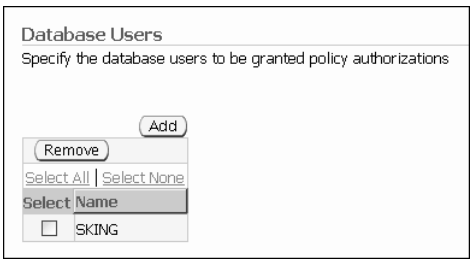
   The Search and Select: Userpage appears. Enter `SKING`, and then click **Go**.

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 needs to 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`.

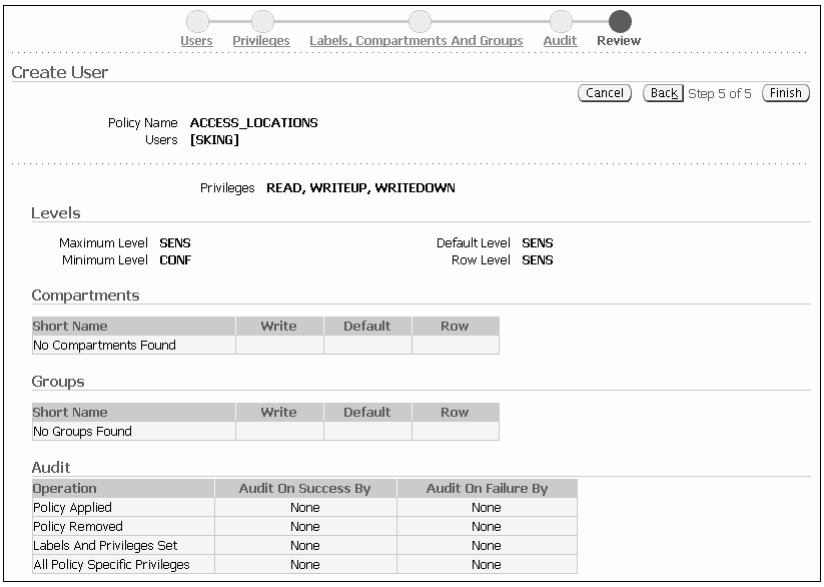


7. Click **Next**.
8. In the Privileges page, select **Next**.

   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.

9. In the Labels, Compartments and Groups page, use the flashlight icon to select data to enter for the following fields, so that user `SKING` will be able to read sensitive and confidential data in `HR.LOCATIONS`:
   - **Maximum Level**: `SENS` (for `SENSITIVE`)
   - **Minimum Level**: `CONF` (for `CONFIDENTIAL`)
   - **Default Level**: `SENS`
   - **Row Level**: `SENS`
10. Click **Next**.
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`.

    - **Privileges**: Select no privileges.
    - **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`
    - **Audit**: Set all to `None`.

14. Create the following authorizations for user `LDORAN`, who is only allowed to read public data from `HR.LOCATIONS`:

    - **Privileges**: Select no privileges.
    - **Labels, Compartments And Groups**: Set all four levels to `PUB`.
    - **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 ACCESS_LOCATIONS policy.
3. In the Actions list, select **Apply**, and then click **Go**.

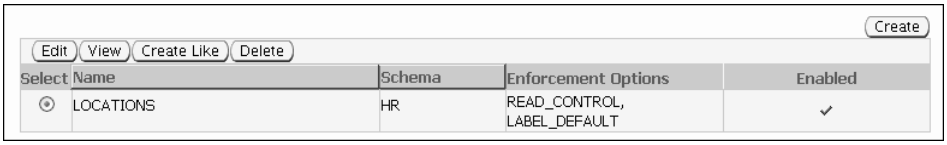
   The Apply page appears.
4. Click **Create**.

   The Add Table page appears.
5. In the **Table** field, enter HR.LOCATIONS.
6. Ensure that the **Hide Policy Column** check box is not selected.
7. Ensure that the **Enabled** check 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 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 **ACCESS_LOCATIONS** policy.
3. Select **Authorization** from the Actions list, and then click **Go**.

   The Authorization page appears.

4. Click **Add Users**.

   The Add User page appears.

5. Under Database Users, click **Add**.

   The Search and Select window appears.

6. Select the check box for user HR, and then click **Select**.

   The Create User page lists user HR.

7. Click **Next**.

   The Privileges step appears.

8. Select the **Bypass all Label Security checks (FULL)** privilege, and then click **Next**.

   The Labels, Compartments, and Groups page appears.

9. Click **Next**.

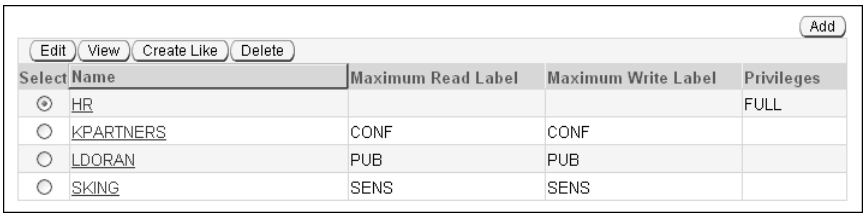
   The Audit step appears.

10. Click **Next**.

    The Review step appears.

11. Click **Finish**.

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



12. 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, whose default password is hr.

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

   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 example, 2_much_fun.

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

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;
```

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, whose password is kingpin2all.

```
CONNECT sking
Enter password: kingpin2all
```

2. Enter the following statement:

```
COL city HEADING City FORMAT a25
COL country_id HEADING Country FORMAT a11
COL Label format a10
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 these steps for users `kpartner` and `ldoran`.

The password for `kpartner` is `eager2please`. She 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.
```

The password for `ldoran` is `too_much2do`. She can access the rows labeled PUB.

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

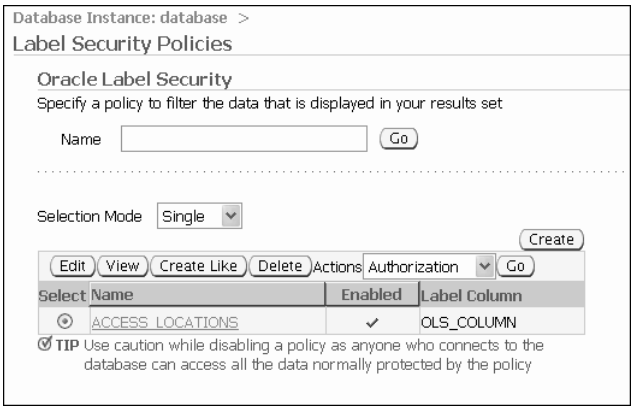
```
South San Francisco       US          PUB
South Brunswick           US          PUB
Seattle                   US          PUB
Toronto                   CA          PUB
Whitehorse                CA          PUB
Bombay                    IN          PUB
Sydney                    AU          PUB
London                    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.
```

Step 10: Optionally, Remove the Components for This Example

Remove the components that you created for this example.

To remove the components for this example:

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 **Server** to display the Server subpage.
8. Click the **Datebase Instance** link to return to the Database Home page.
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`, whose password is `steplively2day`.
13. Click **Server** to display the Server subpage.
14. In the Security section, click **`Oracle Label Security`**.
15. In the Label Security Policies page, in the **Name** field, enter `ACCESS%` and then click **Go**.



16. Ensure that `ACCESS_LOCATIONS` is selected, and then click **Delete**.

    Deleting the `ACCESS_LOCATIONS` policy also drops the `OLS_COLUMN` column from the `HR.LOCATIONS` table.

17. In the Confirmation page, click **Yes**.

7

Auditing Database Activity

You can audit user and database activity by using standard auditing. Standard auditing enables you to audit an entire component, such as a particular SQL statement.

This chapter contains the following topics:

- About Auditing
- Why Is Auditing Used?
- Where Are Standard Audited Activities Recorded?
- Auditing General Activities Using Standard Auditing
- Example: Creating a Standard Audit Trail
- Guidelines for Auditing
- Initialization Parameters Used for Auditing

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

About Auditing

Auditing is the monitoring and recording of selected user database actions. You can use standard auditing to audit SQL statements, privileges, schemas, objects, and network and multitier activity. In standard auditing, you use initialization parameters and the `AUDIT` and `NOAUDIT` SQL statements to audit SQL statements, privileges, and schema objects, as well as network and multitier activities.

There are also activities that Oracle Database always audits, regardless of whether or not 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 future accountability for current actions.**

  These include actions taken in a particular schema, table, or row, or affecting specific content.

- **Deter users (or others, such as intruders) from inappropriate actions based on 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

- **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.

- **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

Where Are Standard Audited 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.

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 explores the following topics:

- 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
- Example: 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 direct the audit for a specific schema table if you want. To perform this type of audit, you use Database Control.

Standard audit records can be written 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.

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` 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 all 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.)
   - `OS`: Enables database auditing and directs all audit records to an operating system file. If you are using a highly secure database configuration, Oracle recommends that you use this setting because it reduces the likelihood of a Denial of Service (DoS) attack. This setting also makes it easier to secure the audit trail. 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 default directory is `$ORACLE_HOME/rdbms/audit`.
   - `NONE`: Disables standard auditing.
   - `DB, EXTENDED`: Performs all actions of the `AUDIT_TRAIL=DB` setting and also populates the SQL bind and SQL text CLOB-type columns of the `SYS.AUD$` 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 except `Sql_Text` and `Sql_Bind` to the operating system XML audit file.
   - `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.)

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.
- 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

This section explains how you can enable the Oracle-recommended audit parameters. It covers the following topics:

- About Default Auditing
- Enabling Default Auditing

About Default Auditing

When you create a new database or modify an existing database, you use the Security Settings window in Database Configuration Assistant (DBCA) to enable or disable its default security settings. This section explains how to start DBCA and enable the default security settings. Oracle recommends that you enable these settings. When these settings are enabled, Oracle Database audits some of the security-relevant SQL statements and privileges. It also sets the AUDIT_TRAIL initialization parameter to DB.

Oracle Database audits the AUDIT ROLE SQL statement by default. The privileges that are audited by default are as follows:

```
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 DB 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 also audits all privileges and statements that have the BY ACCESS clause.

If you are concerned that auditing these statements and privileges will adversely affect your applications, you can disable auditing by using Database Configuration Assistant (DBCA). When you modify your applications to use auditing, you can reenable the default auditing of these statements and privileges.

Oracle strongly recommends that you enable auditing by default. 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, ensuring that patches are applied on schedule, and preventing ad hoc changes. By enabling auditing by default, you can generate an audit record for audit and compliance personnel. However, be aware that auditing may affect database performance.

> **See Also:** *Oracle Database SQL Language Reference* for detailed information about the SQL statements described in this section and the `AUDIT_TRAIL` initialization parameter

Enabling Default Auditing

This section explains how to use Database Configuration Assistant to enable default auditing.

To enable the default profile security settings using Database Configuration Assistant:

1. 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.
2. In the Welcome window, click **Next**.

   The Operations window appears.
3. From the list, select the current database instance, and then click **Next**.

   The Management Options page appears.
4. Select **Keep the database configured with Database Control**.

   The Security Settings page appears.
5. Select the security option you prefer, and then click **Next**.

   Oracle recommends that you take advantage of the enhanced security settings for this release.

   The Database Components page appears.
6. Click **Next**.

   The Connection Mode page appears.
7. 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.

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, such as the `SELECT ANY TABLE` statement. 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;
```

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. In a multitier environment, Oracle Database preserves the identity of the client through all tiers. Thus, you can audit actions performed on behalf of the client by a middle-tier application.

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.

You can use the SQL `AUDIT` statement to audit the activities of a client in a multitier environment. To do so, use the `BY PROXY` clause in the `AUDIT` statement.

For example, to audit SELECT TABLE statements issued on behalf of client jackson by the proxy application server appserve:

```
AUDIT SELECT TABLE BY appserve ON BEHALF OF jackson;
```

> **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 schema 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

Example: Creating a Standard Audit Trail

Suppose you wanted to audit SELECT statements on the OE.CUSTOMERS table. In this example, 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.

Follow these steps to complete this example:

- 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 Example
- 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` 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` 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 the `DB` (Database) option.

   The DB option 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.
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**.

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.
2. Grant `sec_admin` `SELECT` privileges on the `OE.CUSTOMERS` table.
3. Log in to Database Control as user `sec_admin`, whose password is `fussy2all`.
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. Shut down the database instance and then restart it.
    a. In the upper, right corner of the Database Control page, select **Logout**.
    b. Click **Login**.
    c. In the Login page, enter the following login information:

       User Name: `SYS`

       Password: The password of the system administrator

       Connect As: `SYSDBA`

       Use the `SYSDBA` system privilege to shut down and restart the database.
    d. Under General, click **Shutdown**.
    e. In the Startup/Shutdown Credentials page, enter your credentials.

       See *Oracle Database 2 Day DBA* for more information.
    f. After the shutdown completes, click **Startup**.
    g. Exit 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 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, whose password is fussy2all.

   ```
   SQLPLUS sec_admin
   Enter password: fussy2all
   ```

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, TIMESTAMP FROM DBA_AUDIT_TRAIL;
   ```

   Oracle Database displays information similar to the following:

   ```
   USERNAME          TIMESTAMP
   --------------------------
   SEC_ADMIN         07-MAY-07
   ```

4. Exit SQL*Plus:

   ```
   EXIT
   ```

Step 4: Optionally, Remove the Components for This Example

Optionally, remove the audit settings you created earlier.

To remove the audit settings in Database Control:

1. Log in to Database Control using administrative privileges.
2. Go to the Database Control home page.
3. Click **Server** to display the Server subpage.
4. In the Security section, click **Audit Settings**.

   The Audit Settings page appears.
5. Select the **Audited Objects** subpage.
6. Under Schema, enter OE.
7. Under Object Name, enter CUSTOMERS.
8. Click **Search**.
9. Select the check box next to the OE.CUSTOMERS audited schema, and then click **Remove**.

   A Confirmation dialog box appears.
10. Select **Yes**.
11. Exit Database Control.

To set AUDIT_TRAIL to its original value:

- Follow the procedure in "Step 1: Log In and Enable Standard Auditing" on page 7-9 to log in to SQL*Plus and set the `AUDIT_TRAIL` parameter back to its original value. Afterward, shut down and then restart the database.

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, you should remove it.

To remove the sec_admin security administrator account:

1. Log in to Database Control using administrative privileges.
2. Go to the Database Control home page.
3. Click **Server** to display the Server subpage.
4. In the Security section, click **Users**.

   The Users page appears.
5. In the **Name** field, enter `sec_admin`.
6. Click **Search**.
7. Select the check box next to the `sec_admin` user account, and then click **Remove**.

   A Confirmation dialog box appears.
8. Select **Yes**.
9. Exit Database Control.

Guidelines for Auditing

This section contains the following topics:

- Enabling Default Auditing of SQL Statements and Privileges
- Keeping Audited Information Manageable
- Auditing Typical Database Activity
- Auditing Suspicious Database Activity

Enabling 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, as defined in the Sarbanes-Oxley Act. See "Using Default Auditing for Security-Relevant SQL Statements and Privileges" on page 7-5 for more information about default auditing.

Keeping Audited Information Manageable

Although auditing does not severely affect 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 and using valuable space in the `SYSTEM` tablespace. Balance your need to gather sufficient security information with your ability to store and process it.

   For example, if you are auditing to gather information about database activity, then determine exactly what types of activities you want to track, audit only the activities of interest, and audit only for the amount of time necessary to gather the information that you want. As another example, do not audit *objects* if you are only interested in logical I/O information for each session.

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. **Remember 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.

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.
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, writes audit records for user SYS.
AUDIT_SYS_OPERATIONS	FALSE	Enables or disables the auditing of operations issued by user SYS, and users connecting with SYSDBA or SYSOPER privileges. Oracle Database writes the audit records to the audit trail of the operating system. Furthermore, it writes the audit records in XML format if the AUDIT_TRAIL initialization parameter is set to XML or XML, EXTENDED. 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

B

C

D

E

F

G

P

R

S

T

U

V

W

X