Oracle® Label Security

Administrator's Guide

11*g* Release 1 (11.1)

B28529-01

July 2007

ORACLE®

Oracle Label Security Administrator's Guide, 11*g* Release 1 (11.1)

B28529-01

Primary Author: Sumit Jeloka

Contributor: Peter Wahl, Paul Needham, Vikram Pesati, Srividya Tata, Chi Ching Chui, Digvijay Sirmukaddam, Hozefa Palitanawala, Pat Huey, Manoj Kamani

Contents

3 Understanding Access Controls and Privileges

Part II Using Oracle Label Security Functionality

4 Getting Started with Oracle Label Security

5 Working with Labeled Data

6 Oracle Label Security Using Oracle Internet Directory

Part III Administering an Oracle Label Security Application

7 Creating an Oracle Label Security Policy

8 Administering User Labels and Privileges

9 Implementing Policy Enforcement Options and Labeling Functions

10 Applying Policies to Tables and Schemas

11 Administering and Using Trusted Stored Program Units

12 Auditing Under Oracle Label Security

13 Using Oracle Label Security with a Distributed Database

14 Performing DBA Functions Under Oracle Label Security

15 Releasability Using Inverse Groups

Index

List of Figures

List of Tables

Preface

Oracle Label Security enables access control to reach specific (labeled) rows of a database. With Oracle Label Security in place, users with varying privilege levels automatically have (or are excluded from) the right to see or alter labeled rows of data.

This *Oracle Label Security Administrator's Guide* describes how to use Oracle Label Security to protect sensitive data. It explains the basic concepts behind label-based security and provides examples to show how it is used.

This preface contains these topics:

Audience

The Oracle Label Security Administrator's Guide is intended for database administrators (DBAs), application programmers, security administrators, system operators, and other Oracle users who perform the following tasks:

- Analyze application security requirements
- Create label-based security policies
- Administer label-based security policies
- Use label-based security policies

To use this document, you need a working knowledge of SQL and Oracle fundamentals. You should also be familiar with Oracle security features described in "Related Documentation" on page -xx. To use SQL*Loader, you must know how to use the file management facilities of your operating system.

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 Documentation

For more information, see these Oracle resources:

- *Oracle Database Concepts*
- *Oracle Database Security Guide*
- *Oracle Database Enterprise User Security Administrator's Guide*
- *Oracle Database Advanced Application Developer's Guide*
- *Oracle Database Administrator's Guide*
- *Oracle Database SQL Language Reference*
- *Oracle Database Reference*
- *Oracle Database Advanced Replication*
- *Oracle Database Utilities*
- *Oracle Database Performance Tuning Guide*

Many of the examples in this book use the sample schemas, which are installed by default when you select the Basic Installation option with an Oracle Database installation. Refer to *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

http://oraclestore.oracle.com/

To download free release notes, installation documentation, white papers, or other collateral, please visit the Oracle Technology Network (OTN). You must register online before using OTN; registration is free and can be done at

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

If you already have a user name and password for OTN, then you can go directly to the documentation section of the OTN Web site at

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

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples
- Conventions for Windows Operating Systems

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width) font`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, Recovery Manager keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.
`lowercase monospace (fixed-width) font`	Lowercase monospace typeface indicates executable programs, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names and connect identifiers, user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. *Note:* Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to start SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true`. Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase italic monospace (fixed-width) font`	Lowercase italic monospace font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run *`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from usual text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
`[ ]`	Anything enclosed in brackets is optional.	`DECIMAL (digits [ , precision ])`
`{ }`	Braces are used for grouping items.	`{ENABLE | DISABLE}`
`|`	A vertical bar represents a choice of two options.	`{ENABLE | DISABLE}` `[COMPRESS | NOCOMPRESS]`
`...`	Ellipsis points mean repetition in syntax descriptions.	`CREATE TABLE ... AS subquery;`
	In addition, ellipsis points can mean an omission in code examples or text.	`SELECT col1, col2, ... , coln FROM employees;`
Other symbols	You must use symbols other than brackets ([ ]), braces ({ }), vertical bars (|), and ellipsis points (...) exactly as shown.	`acctbal NUMBER(11,2);` `acct CONSTANT NUMBER(4) := 3;`
Italics	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/system_password` `DB_NAME = database_name`

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. Because these terms are not case-sensitive, you can use them in either UPPERCASE or lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates user-defined programmatic elements, such as names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Conventions for Windows Operating Systems

The following table describes conventions for Windows operating systems and provides examples of their use.

Convention	Meaning	Example
Choose **Start**, ***menu item***	How to start a program.	To start the Database Configuration Assistant, choose **Start**, **Programs**, **Oracle** - ***HOME_NAME***, **Configuration and Migration Tools**, **Database Configuration Assistant**.

Convention	Meaning	Example
File and directory names	File and directory names are not case-sensitive. The following special characters are not allowed: left angle bracket (<), right angle bracket (>), colon (:), double quotation marks ("), slash (/), pipe (|), and dash (-). The special character backslash (\) is treated as an element separator, even when it appears in quotation marks. If the filename begins with \\, then Windows assumes it uses the Universal Naming Convention.	c:\winnt"\"system32 is the same as C:\WINNT\SYSTEM32
`C:\>`	Represents the Windows command prompt of the current hard disk drive. The escape character in a command prompt is the caret (^). Your prompt reflects the subdirectory in which you are working. Referred to as the *command prompt* in this manual.	`C:\oracle\oradata>`
Special characters	The backslash (\) special character is sometimes required as an escape character for the double quotation mark (") special character at the Windows command prompt. Parentheses and the single quotation mark (') do not require an escape character. Refer to your Windows operating system documentation for more information about escape and special characters.	`C:\> exp HR/HR TABLES=emp QUERY=\"WHERE job='REP'\"`
HOME_NAME	Represents the Oracle home name. The home name can be up to 16 alphanumeric characters. The only special character allowed in the home name is the underscore.	`C:\> net start Oracle`*HOME_NAME*`TNSListener`

Convention	Meaning	Example
ORACLE_HOME and *ORACLE_BASE*	In releases prior to Oracle8*i* release 8.1.3, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory. The default for Windows NT was `C:\orant`. This release complies with Optimal Flexible Architecture (OFA) guidelines. All subdirectories are not under a top level *ORACLE_HOME* directory. There is a top level directory called *ORACLE_BASE* that by default is `C:\oracle\product\10.1.0`. If you install the latest Oracle release on a computer with no other Oracle software installed, then the default setting for the first Oracle home directory is `C:\oracle\product\10.1.0\db_`*n*, where *n* is the latest Oracle home number. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. Refer to *Oracle Database Installation Guide for Microsoft Windows* for additional information about OFA compliances and for information about installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE*`\`*ORACLE_HOME*`\rdbms\admin` directory.

Part I

This part introduces the terms, concepts, and relationships that constitute the basic elements of Oracle Label Security. It contains the following chapters:

- Chapter 1, "Introduction to Oracle Label Security"
- Chapter 2, "Understanding Data Labels and User Labels"
- Chapter 3, "Understanding Access Controls and Privileges"

1

Introduction to Oracle Label Security

Control of access to sensitive information is of concern to managers, information officers, DBAs, application developers, and many others. Selective access control based on a user's level of security clearance can ensure confidentiality without overbroad limitations. This level of access control ensures that sensitive information will be unavailable to unauthorized persons even while authorized users have access to needed information, sometimes in the same tables.

Data can be viewed as sensitive for different reasons. Examples include personal and private matters or communications, professional trade secrets, company plans for marketing or finance, military information, or government plans for research, purchases, or other actions.

Allowing information to be seen or used by inappropriate persons can be embarrassing, damaging, or dangerous to individuals, careers, organizations, agencies, governments, or countries.

However, such data is often intermingled with other, less sensitive information that is legitimately needed by diverse users. Restricting access to entire tables or segregating sensitive data into separate databases can create an awkward working environment that is costly in hardware, software, user time, and administration.

Oracle Label Security obviates the need for such measures by enabling row-level access control, based on the virtual private database technology of *Oracle Database* 11*g* Release 1 (11.1) Enterprise Edition. It controls access to the contents of a row by comparing that row's label with a user's label and privileges. Administrators can easily add selective row-restrictive policies to existing databases by means of the user-friendly graphical interface provided by Enterprise Manager Database Control. Developers can readily add label-based access control to their *Oracle Database* applications.

This chapter introduces Oracle Label Security in the larger context of data security. It contains the following sections:

Note: This book assumes that you understand the basic concepts and terminology of Oracle Database administration and application development. It supplements core *Oracle Database* 11*g* Release 1 (11.1) documentation by focusing on the additional considerations involved in using, administering, and developing applications for Oracle Label Security.

See Also: For a complete introduction to *Oracle Database* 11*g* Release 1 (11.1) features and terminology, refer to *Oracle Database Concepts*

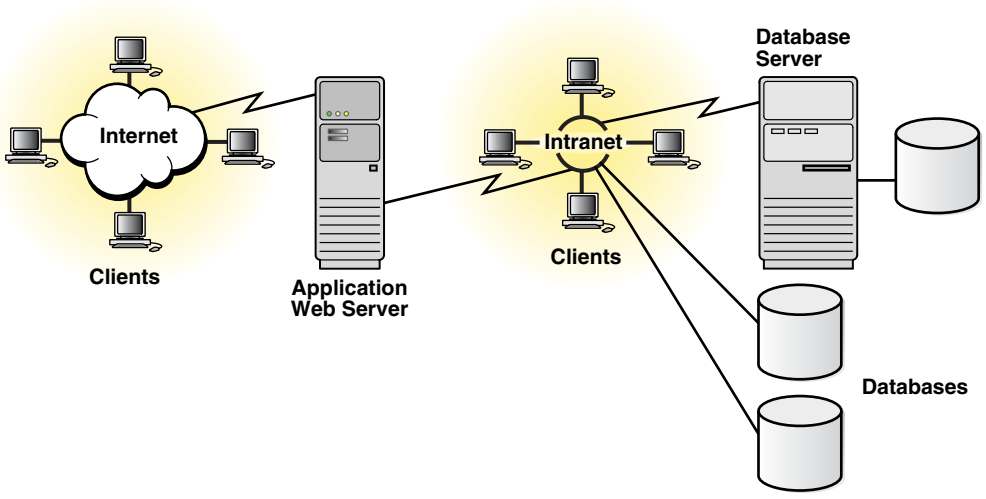
Computer Security and Data Access Controls

Computer security involves the protection of computerized data and processes from unauthorized modification, destruction, disclosure, or delay. In the Internet age, the risks to valuable and sensitive data are greater than ever before. Figure 1–1, "Scope of Data Security Needs" shows the complex computing environment that your data security plan must encompass.

This section introduces basic terms and concepts of computer security as they relate to Oracle Label Security, in the following topics:

- Oracle Label Security and Security Standards
- Security Policies
- Access Control

Figure 1–1 Scope of Data Security Needs



Security officers, administrators, and application programmers must protect databases and the servers on which those databases reside. Also they must administer and protect the rights of internal database users, and they must guarantee electronic commerce confidentiality as customers access those databases. Oracle provides products to address this full spectrum of computer security issues.

Oracle Label Security and Security Standards

Oracle is a leader in information assurance. Security evaluation is a formal assessment process performed by independent bodies against national and international criteria. It provides external and objective assurance that a system meets the security criteria for which it was designed. On successful completion of evaluation, a security rating is assigned to the system or product. This certification provides confidence in the security of products and systems to commercial and government users.

Oracle RDBMS has met the Database Management System Protection Profile (DBMS PP). Oracle Label Security has been evaluated under the Common Criteria (ISO 15408) at Evaluation Assurance Level (EAL) 4, the highest level generally achieved by commercial software vendors.

Security Policies

A database security policy implements an overall system security policy within a broad, organizational security policy. The overall security policy can enforce the following types of rules:

Type of Rules	Purpose
Data Integrity Rules	To ensure that information in the system is consistent
Availability Rules	To ensure that information in the system is available
Access Control Rules	To prevent unauthorized disclosure of information Oracle Label Security provides a default policy for information access control and also enables you to define other, more customized policies for use at any given site.

Access Control

Access control defines a user's ability to read, write, update, insert, or delete information. The following approaches are available to meet access control needs:

- Discretionary Access Control
- Oracle Label Security
- How Oracle Label Security Works with Discretionary Access Control

Discretionary Access Control

Oracle Database 11*g* Release 1 (11.1) provides **discretionary access control** (DAC) on each table, controlling access to information through privileges (SELECT, INSERT, UPDATE, and DELETE) that authorize corresponding SQL operations on the table.

DAC controls access to data in a one-dimensional, binary way, meaning that access is granted or denied to the entire object. The administrator grants users privileges that determine the operations they can perform upon data. To access an *object*, such as a table or view, a user or process must have the proper privilege, such as the SELECT privilege. To access the data in an object, a user or process must first have the necessary DAC privileges.

Oracle Label Security

Labels enable sophisticated access control rules beyond those of DAC by using data in the row. When a policy is applied, a new column is added to each data row. This column will store the label reflecting each row's sensitivity within that policy. Level

access is then determined by comparing the user's identity and label with that of the row.

Oracle Label Security access control depends first on the basic DAC policy. Together, DAC and Oracle Label Security dictate the criteria controlling whether access to a row is permitted or denied.

In most applications, only a relatively small number of tables need the extra security of label-based access controls. The protection provided by standard DAC is sufficient for the majority of application tables.

How Oracle Label Security Works with Discretionary Access Control

To be allowed access to a row, a user must first satisfy *Oracle Database* DAC requirements and then satisfy Oracle Label Security requirements.

Oracle Database enforces DAC based on the user's system and object privileges: The user must be authenticated to the *Oracle Database* and must also have the object and system privileges DAC requires for the requested operation.

If DAC permits access, the user's requested operation must then meet the criteria added by Oracle Label Security, using all of the following guidelines:

- Oracle Label Security label definitions and label hierarchies
- the labels of the user and row
- Oracle Label Security enforcement options
- the user's Oracle Label Security policy privileges

The flexibility and functionality of Oracle Label Security supports applications in a wide variety of production environments. It maintains standard *Oracle Database* 11*g* Release 1 (11.1) data integrity, availability, and recovery capabilities, including user accountability and auditing, while enforcing a site's security policies.

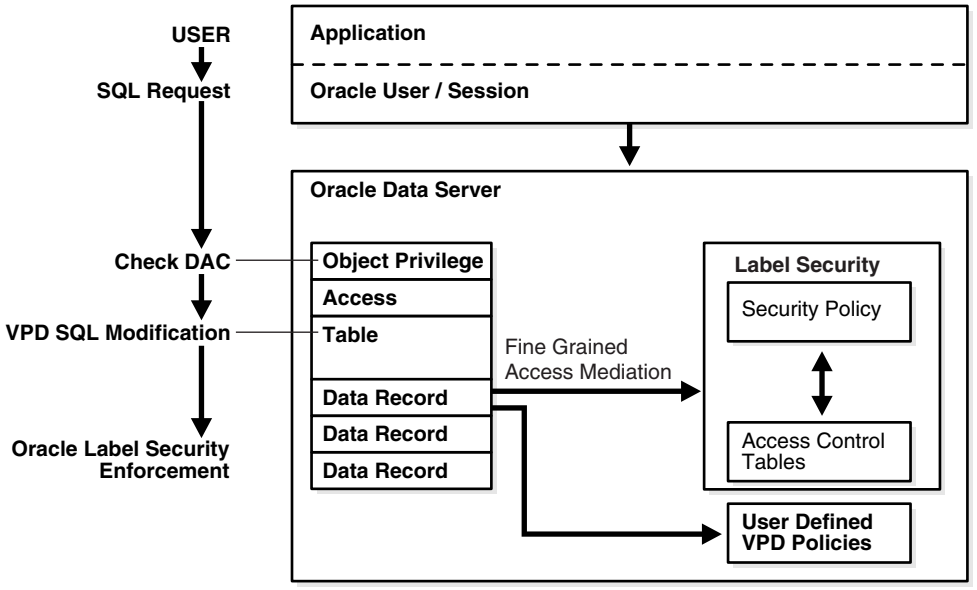
Figure 1–2, "Oracle Label Security Architecture" illustrates how data is accessed under Oracle Label Security, showing the sequence of DAC and label security checks. An application user in an *Oracle Database* 11*g* Release 1 (11.1) session sends out a SQL request. *Oracle Database* checks the DAC privileges, making sure that the user has SELECT privileges on the table. Then it checks whether a Virtual private Database (VPD) policy has been attached to the table, finding that the table is protected by Oracle Label Security. The SQL statement is modified.

Oracle Label Security is started for each row. Access is granted or denied based on result of comparing the data label and the session label of the user, which is again based on the Oracle Label Security privileges of the user.

Oracle Label Security Architecture

Oracle Label Security is built on the VPD technology delivered in the *Oracle Database* 11*g* Release 1 (11.1) Enterprise Edition and leverages that product's Application Context functionality.

Figure 1–2 Oracle Label Security Architecture



Features of Oracle Label Security

Oracle Label Security provides row-level security access controls that operate in addition to the underlying access controls of the *Oracle Database*. This section presents Oracle Label Security features in the following topics:

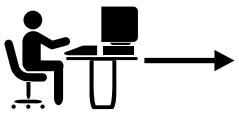
- Overview of Oracle Label Security Policy Functionality
- Oracle Enterprise Edition: VPD Technology
- Oracle Label Security: An Out-of-the-Box VPD
- Label Policy Features

Overview of Oracle Label Security Policy Functionality

A Label Security administrator defines a set of labels for data and users, along with authorizations for users and program units, that govern access to specified protected objects. A policy is nothing more than a name associated with these labels, rules, and authorizations.

For example, assume that a user has the SELECT privilege on an application table. As illustrated in Figure 1–3, "Oracle Label Security Label-Based Security", when the user runs a SELECT statement, Oracle Label Security evaluates each row selected to determine whether the user can access it. The decision is based on the privileges and access labels assigned to the user by the security administrator. Oracle Label Security can be configured to perform security checks on UPDATE, DELETE, and INSERT statements as well.

Figure 1–3 Oracle Label Security Label-Based Security



GRADE	RATE	ROW LABEL
Manager	600	UNCLASSIFIED
Senior	400	UNCLASSIFIED
Director	750	HIGHLY_SENSITIVE
Principal	600	SENSITIVE
Senior	450	SENSITIVE

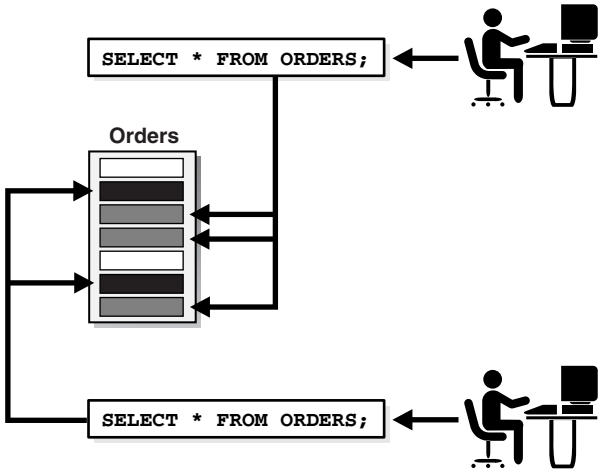
- Oracle Label Security enables a comprehensive set of access authorizations, explained in Chapter 3, "Understanding Access Controls and Privileges", to ensure that the sensitivity label itself can be protected, separately from the other data contained in the row.
- Oracle Label Security provides for flexible policy enforcement to handle special processing requirements. Examples include limiting enforcement to only one type of Data Manipulation Language (DML) statement, limiting label creation by users, or enabling default labels.
- Policies can protect individual application tables. Usually not all tables in an application need to be protected. For example, lookup tables such as zip codes do not need such protection.
- Oracle Label Security allows the security administrator to add special labeling functions and SQL predicates to a policy, possibly simplifying user operations.
- Administrators or application developers can create multiple Oracle Label Security policies. For example, a human resources policy can coexist with a defense policy in the same database. Each policy can be independently configured, with its own unique label definitions and its own column for data labels.
- A single policy can be defined and applied to multiple application tables.

Oracle Enterprise Edition: VPD Technology

VPD supports policy-driven access control. VPD policies enforce object-level access control or row-level security. It provides an application programming interface (API) that allows security policies to be assigned to database tables and views. For example, one can allow access to salary data only for managers in the same facility. Using PL/SQL, developers and security administrators can create security policies with stored procedures. These procedures can be bound to a table or view by means of a call to an RDBMS package. Such policies restrict access by using the content of application data stored in *Oracle Database* or context variables provided by Oracle, such as user name or IP address. Using VPD policies permits developers to remove access security mechanisms from applications and centralize them within *Oracle Database*.

As illustrated in Figure 1–4, "Oracle Database 11g Release 1 (11.1) Enterprise Edition Virtual Private Database Technology", VPD lets you associate security conditions with tables, views, or synonyms. In this example, when each user selects from the ORDERS table, the required security condition is automatically enforced. No matter how the data is accessed, the server automatically enforces security policies, eliminating the need to use many views to implement security.

Figure 1–4 Oracle Database 11g Release 1 (11.1) Enterprise Edition Virtual Private Database Technology



Oracle Label Security: An Out-of-the-Box VPD

Oracle Label Security provides a built-in security policy and infrastructure that easily enforces row-level security. This out-of-the-box solution requires no programming, thereby reducing both total cost of ownership and the time to market for new products and applications.

Oracle Label Security administrators can create policies for row-level security by providing a descriptive name, without writing PL/SQL. There is no need to write additional code. In a single step you can apply a security policy to a given table. This straightforward, efficient way to implement fine-grained security policies allows a granularity and flexibility not easily achieved with VPD alone. This way Oracle Label Security is a generic solution that can be used in different circumstances.

Label Policy Features

Oracle Label Security adds label-based access controls to the Oracle object-relational database management system. Access to data is mediated based on these factors:

Table 1–1 Access Mediation Factors in Oracle Label Security

Label or Policy Factor	Chapter Reference
The label of the data row to which access is requested	Chapter 3, "Understanding Access Controls and Privileges"
The label of the user session requesting access	Chapter 3, "Understanding Access Controls and Privileges"
The policy privileges for that user session	Chapter 3, "Understanding Access Controls and Privileges"
The policy enforcement options established for that table	Chapter 3, "Understanding Access Controls and Privileges"

Consider, for example, a standard DML operation (such as SELECT) performed on a row of data. When evaluating this access request by a user with the CONFIDENTIAL label, to a data row labeled CONFIDENTIAL, Oracle Label Security determines that

this access can, in fact, be achieved. If the row label were higher, say TOP SECRET, access would be denied.

In this way, data of different sensitivities, or belonging to different companies, can be stored and managed on a single system, while preserving data security through standard Oracle access controls. Likewise, applications from a broad range of industries can use row labels with policies providing additional highly targeted access control wherever necessary, without disturbing other existing uses for the same tables.

Labels and policy enforcement depend on the factors explained in the following sections:

- Data Labels
- Label Authorizations
- Policy Privileges
- Policy Enforcement Options
- Summary: Four Aspects of Label-Based Row Access

Data Labels

In Oracle Label Security, each row of a table can be labeled based on its level of confidentiality. Every label contains three components:

- a single level (sensitivity) ranking
- zero or more horizontal compartments or categories
- zero or more hierarchical groups

Levels represent a hierarchy of data sensitivity to exposure or corruption, where the concern is maintaining privacy or security. Levels constitute the primary mechanism to exclude users who are not authorized to see or alter certain data. A user with a lower authorization level, represented by a numerically lower number, is automatically restricted from accessing data labeled with a higher level number. A typical government organization might define levels CONFIDENTIAL, SENSITIVE, and HIGHLY_SENSITIVE. A commercial organization might define a single level for COMPANY_CONFIDENTIAL data.

The compartment component is not hierarchical, but it designates some useful categories typically defined to segregate data, such as data related to separate ongoing strategic initiatives. Some organizations omit using compartments initially.

The group component is hierarchical and is used to reflect ownership. For example, FINANCE and ENGINEERING groups can be defined as children of the CEO group, creating an ownership relation. This hierarchy determines that a user labeled with only ENGINEERING could not view data labeled with FINANCE, but a user labeled CEO could see data labeled as either subgroup. The full rules for how groups determine access are described in Chapter 3, "Understanding Access Controls and Privileges".

A label can be any one of the following four combinations of components:

- a single level component, with no groups or compartments, such as U::
- a level and a set of compartments with no groups, such as U:Alpha, Beta:
- a level and a set of groups with no compartments, such as U::FIN, ASIA
- a level with both compartments and groups, such as U:Beta, Psi:ASIA, FIN

Label Authorizations

Users can be granted label authorizations that determine the kind of access (read or write) they have to the rows that are labeled. When a label has been applied to a row, only users authorized for access to that label can see it or possibly change it. No user can access or affect rows for which that user lacks necessary authorization. If a row has multiple labels, then a user must have the required authorizations for each such label to see or alter that row.

Policy Privileges

Policy privileges enable a user or stored program unit to bypass some aspects of the label-based access control policy. In addition, the administrator can authorize the user or program unit to perform specific actions, such as the ability of one user to assume the authorizations of a different user. Chapter 3, "Understanding Access Controls and Privileges" explains privileges.

Privileges can be granted to program units, authorizing the procedure, rather than the user, to perform privileged operations. System security is at its highest when only stored program units, and not individual users, have Oracle Label Security privileges. Further, such program units encapsulate the policy, minimizing the amount of application code that must be reviewed for security.

Policy Enforcement Options

In Oracle Label Security, administrators or application developers can apply different policy enforcement options for maximum flexibility in controlling the DML operations users can perform. Chapter 8, "Administering User Labels and Privileges" explains policy enforcement options.

Summary: Four Aspects of Label-Based Row Access

When label-based access is enforced within a protected table, access to a row requires a user's label to meet certain criteria determined by policy definitions. These access controls act as a secondary access mediation check, after the discretionary access controls implemented by the application developers.

In summary, Oracle Label Security provides four aspects of label-based access control:

- A user's label indicates the information that a user is permitted to access, and determines the type of access (read or write) the user is allowed to perform.
- A row's label indicates the sensitivity of the information that the row contains, and can also indicate its ownership and its affiliation with similar data.
- A user's policy privileges can enable bypassing some aspects of a label-based access control policy.
- A table's policy enforcement options determine various aspects of how access controls are enforced for read and write operations.

Oracle Label Security Integration with Oracle Internet Directory

Sites that integrate their use of Oracle Label Security with Oracle Internet Directory gain significant efficiencies of label security operation and administration. Policies and user authorization profiles are created and managed directly in the directory by means of the commands described in Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory". Changes are automatically propagated to the associated directories.

A complete introduction to this integration is presented in Chapter 6, "Oracle Label Security Using Oracle Internet Directory".

2

Understanding Data Labels and User Labels

This chapter discusses the fundamental concepts of data labels and user labels, and introduces the terminology that will help you understand Oracle Label Security.

The chapter includes:

Introduction to Label-Based Security

Label-based security provides a flexible way of controlling access to sensitive data. Oracle Label Security controls data access based on the identity and label of the user, and the sensitivity and label of the data. Label security adds protections beyond the discretionary access controls that determine the operations users can perform upon data in an *object*, such as a table or view.

An Oracle Label Security policy controls access to data in three dimensions:

Data Dimension	Explanation
Data Labels	A data row label indicates the level and nature of the row's sensitivity and specifies the additional criteria that a user must meet to gain access to that row.
User Labels	A user label specifies that user's sensitivity level plus any compartments and groups that constrain the user's access to labeled data. Each user is assigned a range of levels, compartments, and groups, and each session can operate within that authorized range to access labeled data within that range.
Policy Privileges	Users can be given specific rights (privileges) to perform special operations or to access data beyond their label authorizations.

Note that the discussion here concerns *access* to data. The particular *type* of access, such as reading or writing the data, is covered in Chapter 3, "Understanding Access Controls and Privileges". Policy privileges are covered in Chapter 8, "Administering User Labels and Privileges"

When an Oracle Label Security policy is applied to a database table, a column is added to the table to contain each row's label. The administrator can choose to display or hide this column.

Label Components

This section describes the elements defined for use in labels.

Label Component Definitions and Valid Characters

A sensitivity label is a single attribute with multiple components. All data labels must contain a level component, but the compartment and group components are optional. An administrator must define the label components before creating labels.

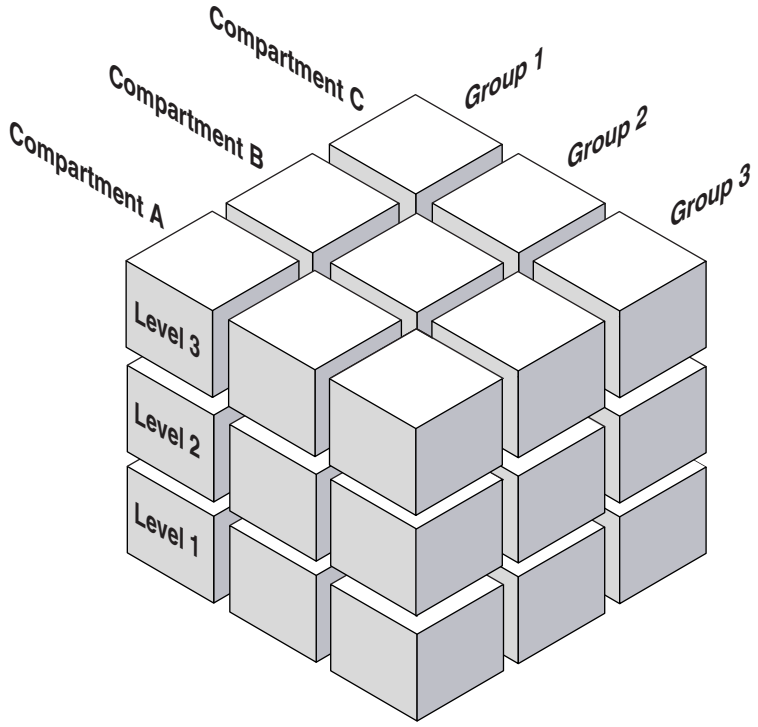
Table 2–1 Sensitivity Label Components

Component	Description	Examples
Level	A single specification of the labeled data's sensitivity within the ordered ranks established	CONFIDENTIAL (1), SENSITIVE (2), HIGHLY_ SENSITIVE (3)
Compartments	Zero or more categories associated with the labeled data	FINANCIAL, STRATEGIC, NUCLEAR
Groups	Zero or more identifiers for organizations owning or accessing the data	EASTERN_REGION, WESTERN_REGION

Valid characters for specifying all label components include alphanumeric characters, underscores, and spaces. (Leading and trailing spaces are ignored.)

The following figure illustrates the three dimensions in which data can be logically classified, using levels, compartments, and groups.

Figure 2–1 Data Categorization with Levels, Compartments and Groups



Levels

A *level* is a ranking that denotes the sensitivity of the information it labels. The more sensitive the information, the higher its level. The less sensitive the information, the lower its level.

Every label must include one level. Oracle Label Security permits defining up to 10,000 levels in a policy. For each level, the Oracle Label Security administrator defines a numeric form and character forms.

For example, you can define a set of levels such as the following:

Table 2–2 Level Example

Numeric Form	Long Form	Short Form
40	*HIGHLY_SENSITIVE*	HS
30	*SENSITIVE*	S
20	*CONFIDENTIAL*	C
10	*PUBLIC*	P

Table 2–3 Forms of Specifying Levels

Form	Explanation
Numeric form, also called "tag"	The numeric form of the level can range from 0 to 9999. Sensitivity is ranked by this numeric value, so you must assign higher numbers to levels that are more sensitive, and lower numbers to levels that are less sensitive. In Table 2–2, 40 (HIGHLY_SENSITIVE) is a higher level than 30, 20, and 10. Administrators should avoid using sequential numbers for the numeric form of levels. A good strategy is to use even increments (such as 50 or 100) between levels. You can then insert additional levels between two preexisting levels, at a later date.
Long form	The long form of the level name can contain up to 80 characters.
Short form	The short form can contain up to 30 characters.

Although the administrator defines both long and short names for the level (and for each of the other label components), only the short form of the name is displayed upon retrieval. When users manipulate the labels, they use only the short form of the component names.

Other sets of levels that users commonly define include TOP_SECRET, SECRET, CONFIDENTIAL, and UNCLASSIFIED or TRADE_SECRET, PROPRIETARY, COMPANY_CONFIDENTIAL, PUBLIC_DOMAIN.

If only levels are used, a level 40 user (in this example) can access or alter any data row whose level is 40 or less.

Note: All levels and labels (including "TOP_SECRET," "SECRET," "CONFIDENTIAL," and so on) in this guide, are used as illustrations only.

Compartments

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

Compartments associate the data with one or more security areas. All data related to a particular project can be labeled with the same compartment. For example, you can define a set of compartments like the following:

Table 2–4 Compartment Example

Numeric Form	Long Form	Short Form
85	*FINANCIAL*	FINCL
65	*CHEMICAL*	CHEM
45	*OPERATIONAL*	OP

Table 2–5 Forms of Specifying Compartments

Form	Explanation

Numeric form	The numeric form can range from 0 to 9999. It is unrelated to the numbers used for the levels. The numeric form of the compartment does not indicate greater or less sensitivity. Instead, it controls the display order of the short form compartment name in the label character string. For example, assume a label is created that has all three compartments listed in Table 2–4, and a level of SENSITIVE. When this label is displayed in string format, it looks like this:
	`S:OP,CHEM,FINCL`
	The display order follows the order of the numbers assigned to the compartments: 45 is lower than 65, and 65 is lower than 85. By contrast, if the number assigned to the FINCL compartment were 5, the character string format of the label would look like this:
	`S:FINCL,OP,CHEM`

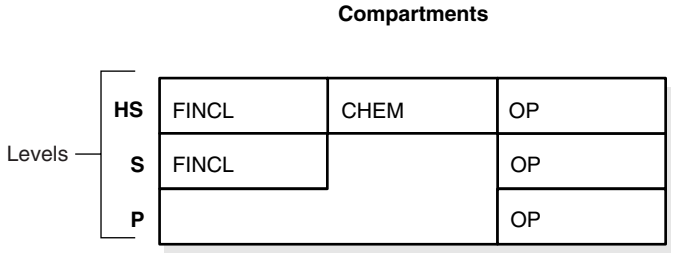
Long form	The long form of the compartment name scan have up to 80 characters.
Short form	The short form can contain up to 30 characters.

Compartments are optional. A label can contain zero or more compartments. Oracle Label Security permits defining up to 10,000 compartments.

Not all labels need to have compartments. For example, you can specify HIGHLY_SENSITIVE and CONFIDENTIAL levels with no compartments, and a SENSITIVE level that does contain compartments.

When you analyze the sensitivity of data, you may find that some compartments are only useful at specific levels. Figure 2–2, "Label Matrix" shows how compartments can be used to categorize data.

Figure 2–2 Label Matrix



Here, compartments FINCL, CHEM, and OP are used with the level HIGHLY_SENSITIVE (40). The label HIGHLY_SENSITIVE:FINCL, CHEM indicates a level of 40 with the two named compartments. Compartment FINCL is not more sensitive than CHEM, nor is CHEM more sensitive than FINCL. Note also that some data in the protected table may not belong to any compartment.

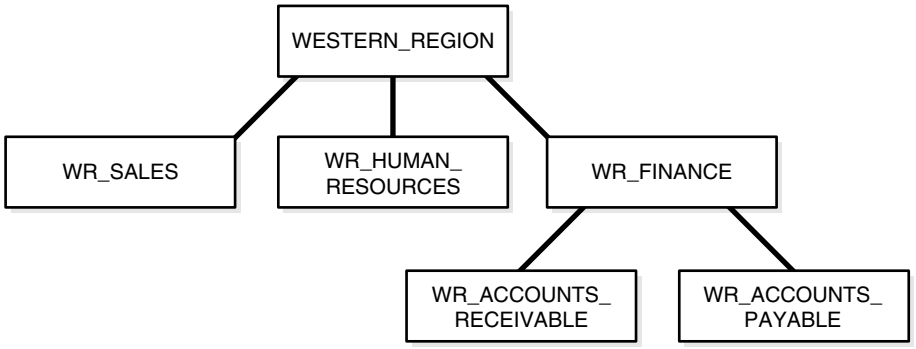
If compartments are specified, then a user whose level would normally permit access to a row's data will nevertheless be prevented from such access unless the user's label also contains all the compartments appearing in that row's label.

Groups

Groups identify organizations owning or accessing the data, such as EASTERN_REGION, WESTERN_REGION, WR_SALES. All data pertaining to a certain department can have that department's group in the label. Groups are useful for the controlled dissemination of data and for timely reaction to organizational change. When a company reorganizes, data access can change right along with the reorganization.

Groups are hierarchical. You can label data based upon your organizational infrastructure. A group can thus be associated with a parent group. For example, you can define a set of groups corresponding to the following organizational hierarchy:

Figure 2–3 Group Example



The WESTERN_REGION group includes three subgroups: WR_SALES, WR_HUMAN_RESOURCES, and WR_FINANCE. The WR_FINANCE subgroup is subdivided into WR_ACCOUNTS_RECEIVABLE and WR_ACCOUNTS_PAYABLE.

Table 2–6 shows how the organizational structure in this example can be expressed in the form of Oracle Label Security groups. Notice that the numeric form assigned to the groups affects display order only. The administrator specifies the hierarchy (that is, the parent/child relationships) separately.

Table 2–6 Group Example

Numeric Form	Long Form	Short Form	Parent Group
1000	WESTERN_REGION	WR	
1100	WR_SALES	WR_SAL	WR
1200	WR_HUMAN_RESOURCES	WR_HR	WR
1300	WR_FINANCE	WR_FIN	WR
1310	WR_ACCOUNTS_PAYABLE	WR_AP	WR_FIN
1320	WR_ACCOUNTS_RECEIVABLE	WR_AR	WR_FIN

Table 2–7 Forms of Specifying Groups

Form	Explanation
Numeric form	The numeric form of the group can range from 0 to 9999, and it must be unique for each policy. The numeric form does not indicate any kind of ranking. It does not indicate a parent-child relationship, or greater or less sensitivity. It only controls the display order of the short form group name in the label character string. For example, assume that a label is created that has the level SENSITIVE, the compartment CHEMICAL, and the groups WESTERN_REGION and WR_HUMAN_RESOURCES as listed in Table . When displayed in string format, the label looks like this: `S:CHEM:WR,WR_HR` WR is displayed before WR_HR because 1000 comes before 1200.
Long form	The long form of the group name can contain up to 80 characters.
Short form	The short form can contain up to 30 characters.

Groups are optional; a label can contain zero or more groups. Oracle Label Security permits defining up to 10,000 groups.

All labels need not have groups. When you analyze the sensitivity of data, you may find that some groups are only used at specific levels. For example, you can specify HIGHLY_SENSITIVE and CONFIDENTIAL labels with no groups, and a SENSITIVE label that does contain groups.

See Also: Chapter 15, "Releasability Using Inverse Groups"

Industry Examples of Levels, Compartments, and Groups

Table 2–8 illustrates the flexibility of Oracle Label Security levels, compartments, and groups, by listing typical ways in which they can be implemented in various industries.

Table 2–8 Typical Levels, Compartments, and Groups, by Industry

Industry	Levels	Compartments	Groups
Defense	TOP_SECRET SECRET CONFIDENTIAL UNCLASSIFIED	ALPHA DELTA SIGMA	UK NATO SPAIN
Financial Services	ACQUISITIONS CORPORATE CLIENT OPERATIONS	INSURANCE EQUITIES TRUSTS COMMERCIAL_LOANS CONSUMER_LOANS	CLIENT TRUSTEE BENEFICIARY MANAGEMENT STAFF
Judicial	NATIONAL_SECURITY SENSITIVE PUBLIC	CIVIL CRIMINAL	ADMINISTRATION DEFENSE PROSECUTION COURT
Health Care	PRIMARY_PHYSICIAN PATIENT_CONFIDENTIAL PATIENT_RELEASE	PHARMACEUTICAL INFECTIOUS_DISEASES	CDC RESEARCH NURSING_STAFF HOSPITAL_STAFF
Business to Business	TRADE_SECRET PROPRIETARY COMPANY_ CONFIDENTIAL PUBLIC	MARKETING FINANCIAL SALES PERSONNEL	AJAX_CORP BILTWELL_CO ACME_INC ERSATZ_LTD

Label Syntax and Type

After defining the label components, the administrator creates data labels by combining particular sets of level, compartments, and groups. Out of all the possible permutations of label components, the administrator specifies those combinations that will actually be used as valid data labels in the database.

This can be done by using the Oracle Enterprise Manager graphical user interface or by using a command line procedure. Character string representations of labels use the following syntax:

LEVEL:*COMPARTMENT1*,...,*COMPARTMENTn*:*GROUP1*,...,*GROUPn*

The text string specifying the label can have a maximum of 4,000 characters, including alphanumeric characters, spaces, and underscores. The labels are case-insensitive. You can enter them in uppercase, lowercase, or mixed case, but the string is stored in the data dictionary and displayed in uppercase. A colon is used as the delimiter between components. It is not necessary to enter trailing delimiters in this syntax.

For example, the administrator might create valid labels such as these:

```
SENSITIVE:FINANCIAL,CHEMICAL:EASTERN_REGION,WESTERN_REGION
CONFIDENTIAL:FINANCIAL:VP_GRP
SENSITIVE
HIGHLY_SENSITIVE:FINANCIAL
SENSITIVE::WESTERN_REGION
```

When a valid data label is created, two additional things occur:

- The label is automatically designated as a valid data label. This functionality limits the labels that can be assigned to data. Oracle Label Security can also create valid data labels dynamically at run time, from those that are predefined in Oracle Internet Directory. Most users, however, prefer to create the labels manually in order to limit data label proliferation.
- A numeric label tag is associated with the text string representing the label. It is this label tag, rather than the text string, that is stored in the policy label column of the protected table.

Note: For Oracle Label Security installations that are not using Oracle Internet Directory, dynamic creation of valid data labels uses the TO_DATA_LABEL function. Its usage should be tightly controlled. Refer to "Inserting Labels Using TO_DATA_LABEL" on page 5-13.

See Also:

- Chapter 7, "Creating an Oracle Label Security Policy" for instructions on creating label components and labels
- "The Policy Label Column and Label Tags" on page 5-1
- "Label Tags" on page 5-3

How Data Labels and User Labels Work Together

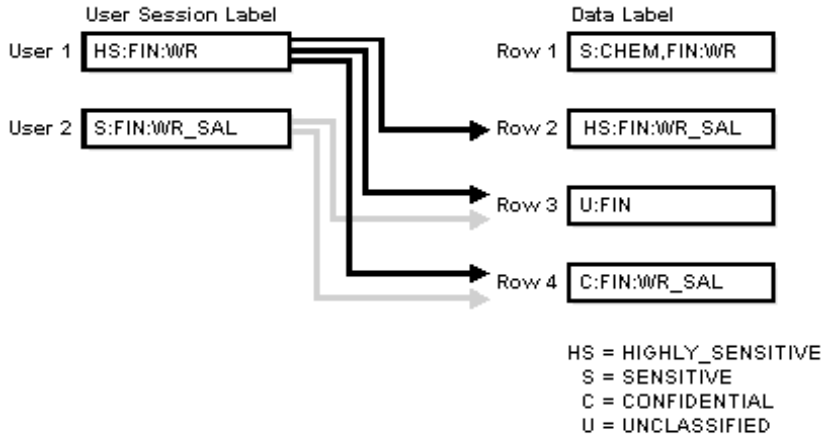
A user can access data only within the range of his or her own label authorizations. A user has:

- Maximum and minimum levels
- A set of authorized compartments
- A set of authorized groups (and, implicitly, authorization for any subgroups)

For example, if a user is assigned a maximum level of SENSITIVE, then the user potentially has access to SENSITIVE, CONFIDENTIAL, and UNCLASSIFIED data. The user has no access to HIGHLY_SENSITIVE data.

Figure 2–4, "Example: Data Labels and User Labels" shows how data labels and user labels work together to provide access control in Oracle Label Security. While data labels are discrete, user labels are inclusive. Depending upon authorized compartments and groups, a user can potentially access data corresponding to all levels within his or her range.

Figure 2–4 Example: Data Labels and User Labels

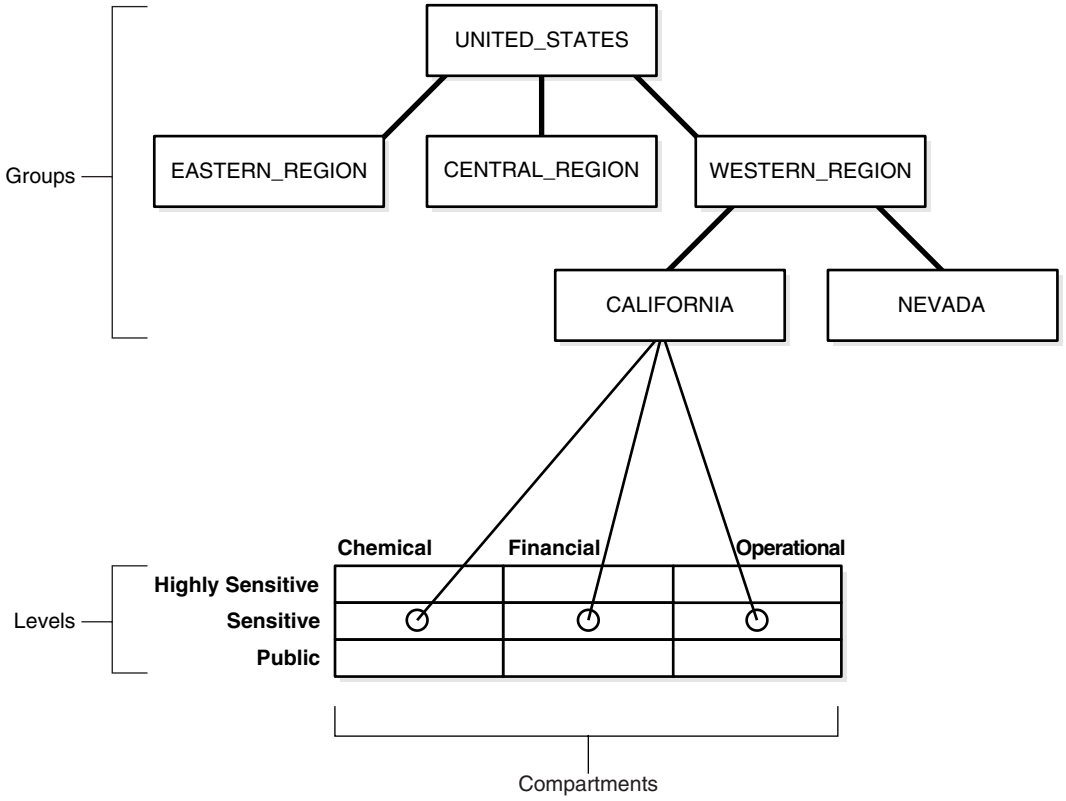


As shown in the figure, User 1 can access the rows 2, 3, and 4 because her maximum level is HS. She has access to the FIN compartment, and her access to group WR hierarchically includes group WR_SAL. She cannot access row 1 because she does not have the CHEM compartment. (A user must have authorization for *all* compartments in a row's data label to be able to access that row.)

User 2 can access rows 3 and 4. His maximum level is S, which is less than HS in row 2. Although he has access to the FIN compartment, he only has authorization for group WR_SAL. So, he cannot access row 1.

Figure 2–5, "How Label Components Interrelate" shows how data pertaining to an organizational hierarchy fits into data levels and compartments.

Figure 2–5 How Label Components Interrelate



For example, the UNITED_STATES group includes three subgroups: EASTERN_REGION, CENTRAL_REGION, and WESTERN_REGION. The WESTERN_REGION subgroup is further subdivided into CALIFORNIA and NEVADA. For each group and subgroup, there may be data belonging to some of the valid compartments and levels within the database. So, there may be SENSITIVE data that is FINANCIAL, within the CALIFORNIA subgroup.

Note that data is generally labeled with a single group whereas users' labels form a hierarchy. If users have a particular group, then that group may implicitly include child groups. This way a user associated with the UNITED_STATES group has access to all data, but a user associated with CALIFORNIA would have access to data pertaining to only that subgroup.

Administering Labels

Oracle Label Security provides administrative interfaces to define and manage the labels used in a database. You define labels in Oracle Database using Oracle Label Security packages or by using Oracle Enterprise Manager. Initially, an administrator must define the levels, compartments, and groups that compose the labels, and then, the user can define the set of valid data labels for the contents of the database.

The administrator can apply a policy to individual tables in the database or to entire application schemas. Finally, the administrator assigns to each database user the label components (and privileges, if needed) required for the user's job function.

See Also: Chapter 10, "Applying Policies to Tables and Schemas" for information about the Oracle Label Security interfaces used to manage label components

3

Understanding Access Controls and Privileges

Chapter 2, "Understanding Data Labels and User Labels" introduced the concept of labels (with their levels, compartments, and groups) and the basic notion of access control based on the row's data label and the user's label. This chapter examines the access controls and privileges that determine the *type* of access users can have to labeled rows.

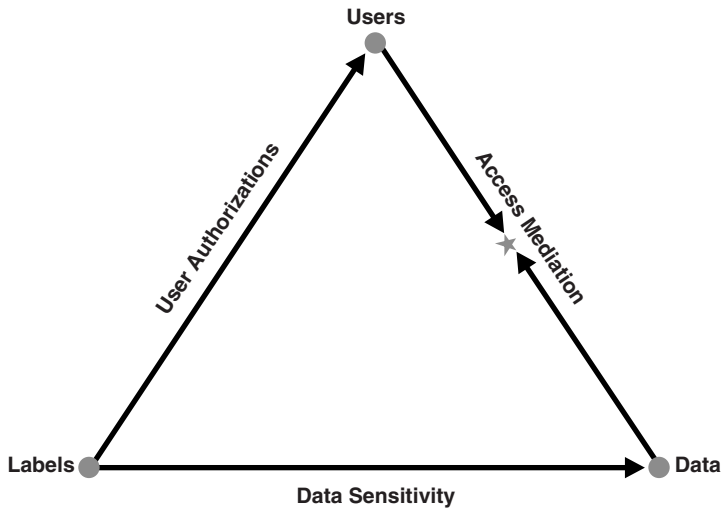
This chapter contains these sections:

Introducing Access Mediation

To access data protected by an Oracle Label Security policy, a user must have authorizations based on the labels defined for the policy. Figure 3–1, "Relationships Between Users, Data, and Labels" illustrates the relationships between users, data, and labels.

- Data labels specify the sensitivity of data rows.
- User labels provide the appropriate authorizations to users.
- Access mediation between users and rows of data depends on users' labels.

Figure 3–1 Relationships Between Users, Data, and Labels



Note: Oracle Label Security enforcement options affect how access controls apply to tables and schemas. This chapter assumes that all policy enforcement options are in effect.

See Also: For more information, Refer to "Choosing Policy Options" on page 9-1

Understanding Session Label and Row Label

This section introduces the basic user labels.

The Session Label

Each Oracle Label Security user has a set of authorizations that include:

- A maximum and minimum level
- A set of authorized compartments
- A set of authorized groups
- For each compartment and group, a specification of read-only access, or read/write access

The administrator also specifies the user's initial session label when setting up these authorizations for the user.

The *session label* is the particular combination of level, compartments, and groups at which a user works at any given time. The user can change the session label to any combination of components for which the user is authorized.

See Also: "Changing Your Session and Row Labels with SA_SESSION" on page 5-14

The Row Label

When a user writes data without specifying its label, a *row label* is assigned automatically, using the user's session label. However, the user can set the label for the written row, within certain restrictions on the components of the label he specifies.

The level of this label can be set to any level within the range specified by the administrator. For example, it can be set to the level of the user's current session label down to the user's minimum level. However, the compartments and groups for this row's new label are more restricted. The new label can include only those compartments and groups contained in the current session label and, among those, only the ones for which the user has write access.

When the administrator sets up the user authorizations, he or she also specifies an initial default row label.

See Also:

- "Managing User Labels by Component, with SA_USER_ADMIN" on page 8-1
- "Changing Your Session and Row Labels with SA_SESSION" on page 5-14

Session Label Example

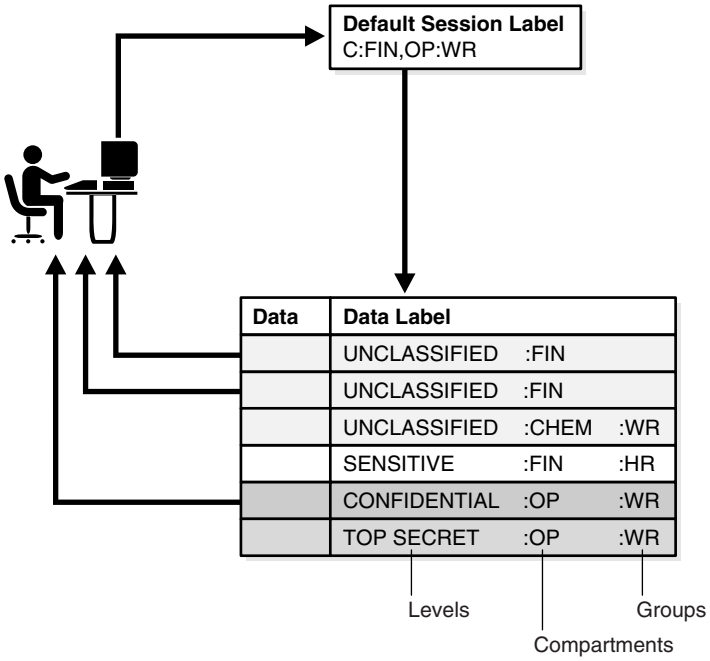
The session label and the row label can fall anywhere within the range of the user's level, compartment, and group authorizations. In Figure 3–2, "User Session Label", the user's maximum level is SENSITIVE and the minimum level is UNCLASSIFIED. However, his default session label is C:FIN,OP:WR. In this example, the administrator has set the user's session label so that the user connects to the database at the CONFIDENTIAL level.

Similarly, although the user is authorized for compartments FIN and OP, and group WR, the administrator could set the session label so that the user connects with only compartment FIN and group WR.

See Also:

- "SA_USER_ADMIN.SET_COMPARTMENTS" on page 8-2 or
- "SA_USER_ADMIN.ALTER_COMPARTMENTS" on page 8-4

Figure 3–2 User Session Label



Understanding User Authorizations

There are two types of user authorizations:

- Authorizations Set by the Administrator
- Computed Session Labels

Authorizations Set by the Administrator

The administrator explicitly sets a number of user authorizations:

- Authorized Levels
- Authorized Compartments
- Authorized Groups

Authorized Levels

The administrator explicitly sets the following level authorizations:

Table 3–1 Authorized Levels Set by the Administrator

Authorization	Meaning
User Max Level	The maximum ranking of sensitivity that a user can access during read and write operations
User Min Level	The minimum ranking of sensitivity that a user can access during write operations. The User Max Level must be equal to or greater than the User Min Level.
User Default Level	The level that is assumed by default when connecting to *Oracle Database*

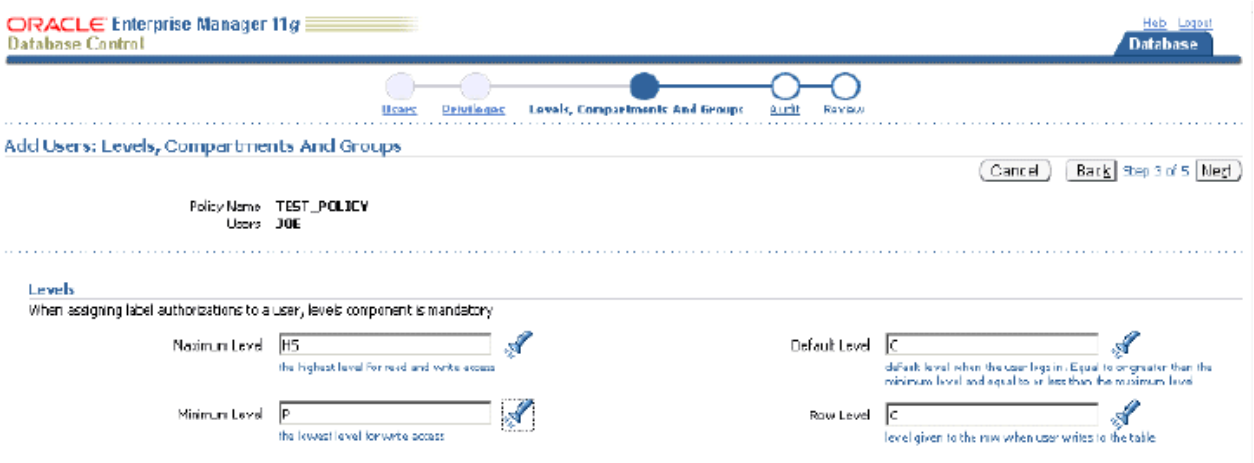
Table 3–1 (Cont.) Authorized Levels Set by the Administrator

Authorization	Meaning
User Default Row Level	The level that is used by default when inserting data into *Oracle Database*

For example, in Oracle Enterprise Manager, the administrator might set the following level authorizations for user Joe:

Type	Short Name	Long Name	Description
Maximum	HS	HIGHLY_SENSITIVE	User's highest level
Minimum	P	PUBLIC	User's lowest level
Default	C	CONFIDENTIAL	User's default level
Row	C	CONFIDENTIAL	Row level on `INSERT`

Figure 3–3 Setting Up Authorized Levels In Enterprise Manager



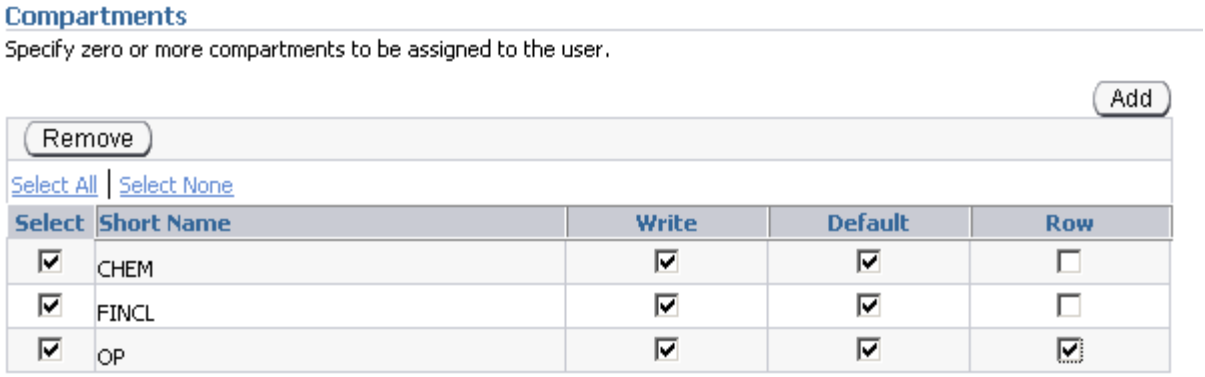
Authorized Compartments

The administrator specifies the list of compartments that a user can place in their session label. Write access must be explicitly given for each compartment. A user cannot directly insert, update, or delete a row that contains a compartment that she does not have authorization to write.

For example, in Oracle Enterprise Manager, the administrator might set the following compartment authorizations for user Joe:

Short Name	Long Name	WRITE	DEFAULT	ROW
CHEM	CHEMICAL	YES	YES	NO
FINCL	FINANCIAL	YES	YES	NO
OP	OPERATIONAL	YES	YES	YES

Figure 3–4 Setting Up Authorized Compartments In Enterprise Manager



In Figure 3–4, "Setting Up Authorized Compartments In Enterprise Manager", the row designation indicates whether the compartment should be used as part of the default row label for newly inserted data. Note also that the LABEL_DEFAULT policy option must be in effect for this setting to be valid.

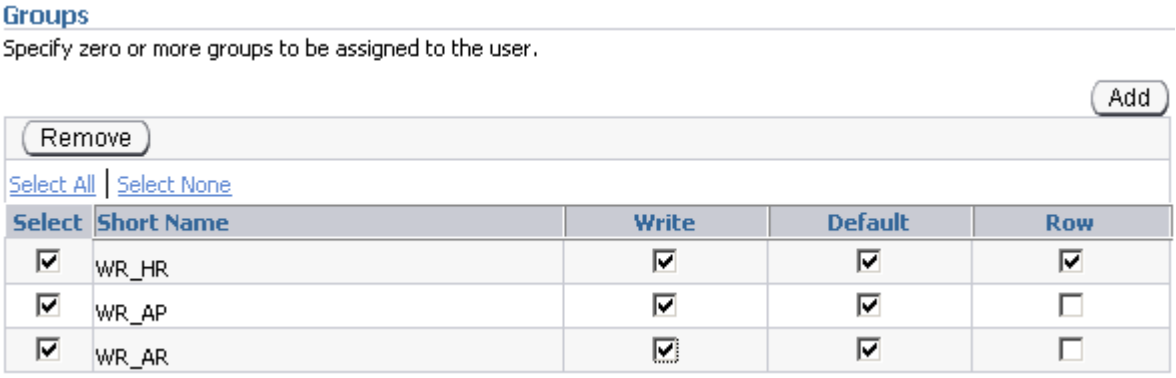
Authorized Groups

The administrator specifies the list of groups that a user can place in session label. Write access must be explicitly given for each group listed.

For example, in Oracle Enterprise Manager, the administrator might set the following group authorizations:

Short Name	Long Name	WRITE	DEFAULT	ROW	Parent
WR_HR	WR_HUMAN_RESOURCES	YES	YES	YES	WR
WR_AP	WR_ACCOUNTS_PAYABLE	YES	YES	NO	WR_FIN
WR_AR	WR_ACCOUNTS_RECEIVABLE	YES	YES	NO	WR_FIN

Figure 3–5 Setting Up Authorized Groups in Enterprise Manager



In Figure 3–5, "Setting Up Authorized Groups in Enterprise Manager", the row designation indicates whether the group should be used as part of the default row label for newly inserted data. Note also that the LABEL_DEFAULT policy option must be in effect for this setting to be valid.

See Also:

- Chapter 8, "Administering User Labels and Privileges" for instructions on setting the authorizations
- "LABEL_DEFAULT: Using the Session's Default Row Label" on page 9-6

Computed Session Labels

Oracle Label Security automatically computes a number of labels based on the value of the session label. These include:

Table 3–2 Computed Session Labels

Computed Label	Definition
Maximum Read Label	The user's maximum level combined with any combination of compartments and groups for which the user is authorized.
Maximum Write Label	The user's maximum level combined with the compartments and groups for which the user has been granted write access.
Minimum Write Label	The user's minimum level.
Default Read Label	The single default level combined with compartments and groups that have been designated as default for the user.
Default Write Label	A subset of the default read label, containing the compartments and groups to which the user has been granted write access. The level component is equal to the level default in the read label. This label is automatically derived from the read label based on the user's write authorizations.
Default Row Label	The combination of components between the user's minimum write label and the maximum write label, which has been designated as the default value for the data label for inserted data.

See Also: "Computed Labels with Inverse Groups" on page 15-4

Evaluating Labels for Access Mediation

When a table is protected by an Oracle Label Security policy, the user's label components are compared to the row's label components to determine whether the user can access the data. In this way, Oracle Label Security evaluates whether the user is authorized to perform the requested operation on the data in the row. This section explains the rules and options by which user access is mediated. It contains these topics:

- Introducing Read/Write Access
- The Oracle Label Security Algorithm for Read Access
- The Oracle Label Security Algorithm for Write Access

Introducing Read/Write Access

Although data labels are stored in a column within data records, information about user authorizations is stored in relational tables. When a user logs on, the tables are used to dynamically generate user labels for use during the session.

Difference Between Read and Write Operations

Two fundamental types of access mediation on Data Manipulation language (DML) operations exist, within protected tables:

- Read access
- Write access

The user has a maximum authorization for the data he or she can read; the user's write authorization is a subset of that. The minimum write level controls the user's ability to disseminate data by lowering its sensitivity. The user cannot write data with a level lower than the minimum level the administrator assigned to this user.

In addition, there are separate lists of compartments and groups for which the user is authorized; that is, for which the user has at least read access. An access flag indicates whether the user can also write to individual compartments or groups.

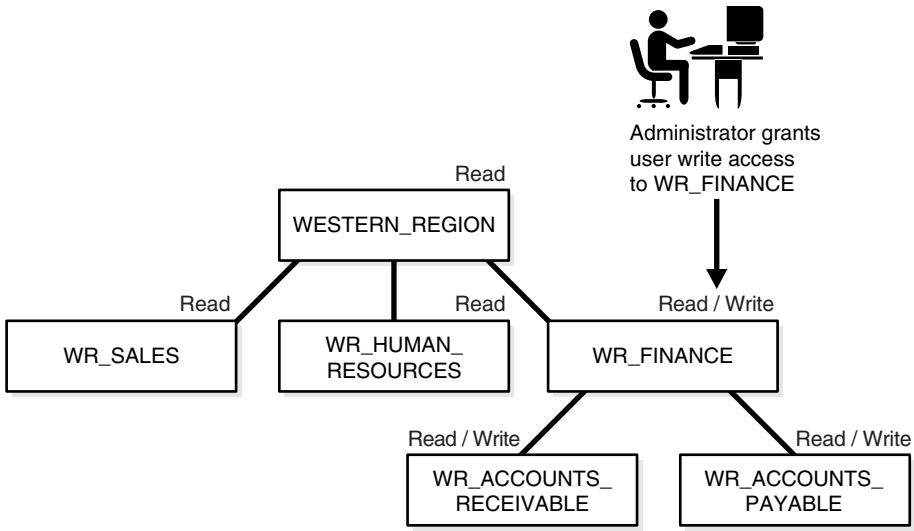
Propagation of Read/Write Authorizations on Groups

When groups are organized hierarchically, a user's assigned groups include all subgroups that are subordinate to the group to which she belongs. In this case, the user's read/write authorizations on a parent group flow down to all the subgroups.

Consider the parent group WESTERN_REGION, with three subgroups as illustrated in Figure 3–6, "Subgroup Inheritance of Read/Write Access". If the user has read access to WESTERN_REGION, then the read access is also granted to the three subgroups. The administrator can give the user write access to subgroup WR_FINANCE, without granting write access to the WESTERN_REGION parent group (or to the other subgroups). On the other hand, if the user has read/write access on WESTERN_REGION, then read/write access is also granted on all of the subgroups subordinate to it in the tree.

Write authorization on a group does not give a user write authorization on the parent group. If a user has read-only access to WESTERN_REGION and WR_FINANCE, then the administrator can grant write access to WR_ACCOUNTS_RECEIVABLE, without affecting the read-only access to the higher-level groups.

Figure 3–6 Subgroup Inheritance of Read/Write Access



See Also:

- "Introduction to User Label and Privilege Management" on page 8-1
- "How Inverse Groups Work" on page 15-2

The Oracle Label Security Algorithm for Read Access

The READ_CONTROL enforcement determines the ability to read data in a row. The following rules are used, in the sequence listed, to determine a user's read access to a row of data:

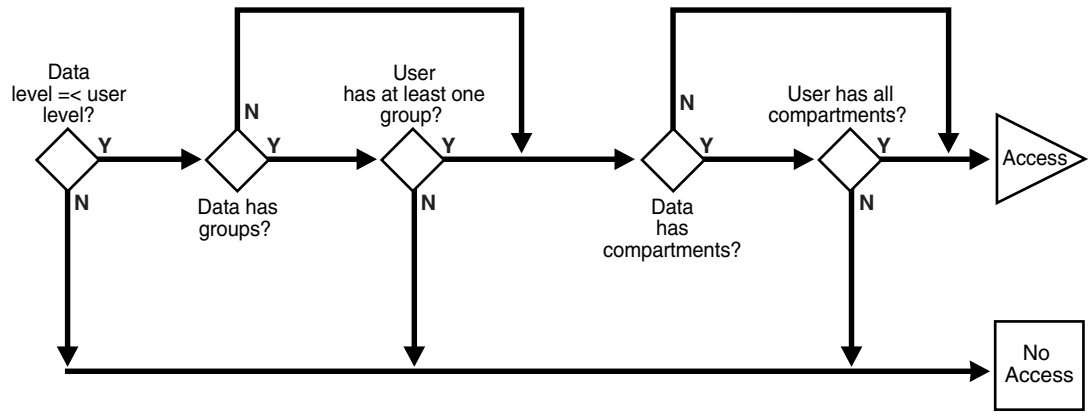
1. The user's level must be *greater than or equal to* the level of the data.
2. The user's label must include *at least one of the groups* that belong to the data (or the parent group of one such subgroup).
3. The user's label must include *all the compartments* that belong to the data.

If the user's label passes these tests, then it is said to dominate the row's label.

Note that there is no notion of read or write access connected with levels. This is because the administrator specifies a range of levels (minimum to maximum) within which a user can potentially read and write. At any time, the user can read all data equal to or less than the current session level. No privileges (other than FULL) allow the user to write below the minimum authorized level.

The label evaluation process proceeds from levels to groups to compartments, as illustrated in Figure 3–7, "Label Evaluation Process for Read Access". Note that if the data label is null or invalid, then the user is denied access.

Figure 3–7 Label Evaluation Process for Read Access



As a read access request comes in, Oracle Label Security evaluates each row to determine the following:

1. Is the user's level equal to, or greater than, the level of the data?
2. If so, does the user have access to at least one of the groups present in the data label?
3. If so, does the user have access to all the compartments present in the data label? (That is, are the data's compartments a subset of the user's compartments?)

If the answer is no at any stage in this evaluation process, then Oracle Label Security denies access to the row and moves on to evaluate the next row of data.

Oracle Label Security policies allow user sessions to read rows at their label and below, which is called *reading down*. Sessions cannot read rows at labels that they do not dominate.

For example, if you are logged in at SENSITIVE:ALPHA,BETA, you can read a row labeled SENSITIVE:ALPHA because your label dominates that of the row. However, you cannot read a row labeled SENSITIVE:ALPHA,GAMMA because your label does not dominate that of the row.

Note that the user can gain access to the rows otherwise denied, if she or he has special Oracle Label Security privileges.

See Also:

- "Privileges Defined by Oracle Label Security Policies" on page 3-12
- "The Access Control Enforcement Options" on page 9-6
- "Algorithm for Read Access with Inverse Groups" on page 15-6
- "Analyzing the Relationships Between Labels" on page A-1

The Oracle Label Security Algorithm for Write Access

In the context of Oracle Label Security, WRITE_CONTROL enforcement determines the ability to insert, update, or delete data in a row.

WRITE_CONTROL enables you to control data access with ever finer granularity. Granularity increases when compartments are added to levels. It increases again when groups are added to compartments. Access control becomes even more fine grained when you can manage the user's ability to write the data that he can read.

To determine whether a user can write a particular row of data, Oracle Label Security evaluates the following rules, in the order given:

1. The level in the data label must be greater than or equal to the user's minimum level and less than or equal to the user's session level.
2. When groups are present, the user's label must include *at least one of the groups with write access* that appear in the data label (or the parent of one such subgroup). In addition, the user's label must include *all the compartments* in the data label.
3. When no groups are present, the user's label must have write access on *all of the compartments* in the data label.

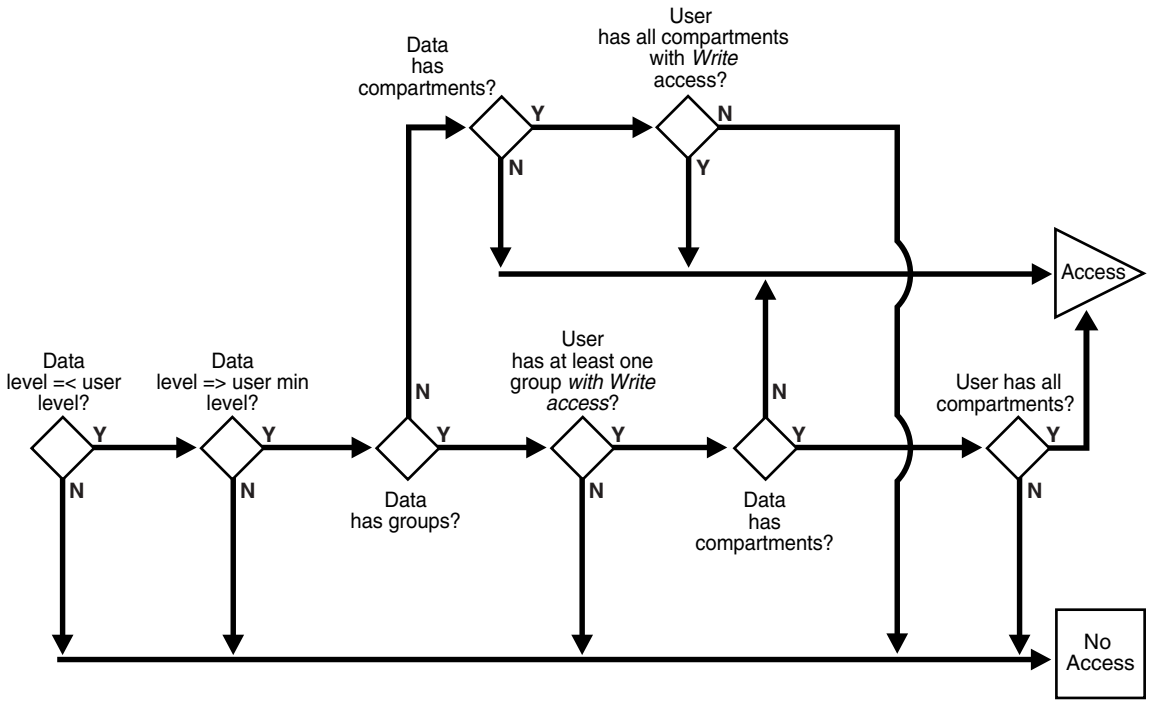
To state tests 2 and 3 another way:

- If the label has *no* groups, then the user must have write access on all the compartments in the label in order to write the data.
- If the label *does* have groups and the user has write access to one of the groups, she only needs read access to the compartments in order to write the data.

Just as with read operations, the label evaluation process proceeds from levels to groups to compartments. Note that the user cannot write any data below the authorized minimum level, nor above the current session level. The user can always read below the minimum level.

The following figure illustrates how the process works with INSERT, UPDATE, and DELETE operations. Note that if the data label is null or invalid, then the user is denied access.

Figure 3–8 Label Evaluation Process for Write Access



As an access request comes in, Oracle Label Security evaluates each row to determine the following:

1. Is the data's level equal to, or less than the level of the user?
2. Is the data's level equal to, or greater than the user's minimum level?
3. If the data's level falls within the foregoing bounds, then does the user have write access to at least one of the groups present in the data label?
4. If so, does the user have access to all the compartments with at least read access that are present in the data label?
5. If there are no groups but there are compartments, then does the user have write access to all of the compartments?

If the answer is no at any stage in this evaluation process, then Oracle Label Security denies access to the row, and moves on to evaluate the next row of data.

Consider a situation in which your session label is S:ALPHA,BETA but you have write access to only compartment ALPHA. In this case, you can read a row with the label S:ALPHA,BETA but you cannot update it.

In summary, write access is enforced on INSERT, UPDATE and DELETE operations upon the data in the row.

In addition, each user may have an associated minimum level below which the user cannot write. The user cannot update or delete any rows labeled with levels below the minimum, and cannot insert a row with a row label containing a level less than the minimum.

See Also:

- "The Access Control Enforcement Options" on page 9-6
- "Algorithm for Write Access with Inverse Groups" on page 15-7

Using Oracle Label Security Privileges

This section introduces the Oracle Label Security database and row label privileges:

- Privileges Defined by Oracle Label Security Policies
- Special Access Privileges
- Special Row Label Privileges
- System Privileges, Object Privileges, and Policy Privileges

Privileges Defined by Oracle Label Security Policies

Oracle Label Security supports special privileges that allow authorized users to bypass certain parts of the policy. Table 3–3 summarizes the full set of privileges that can be granted to users or trusted stored program units. Each privilege is more fully discussed after the table.

Table 3–3 Oracle Label Security Privileges

Security Privilege	Explanation
READ	Allows read access to all data protected by the policy
FULL	Allows full read and write access to all data protected by the policy
COMPACCESS	Allows a session access to data authorized by the row's compartments, independent of the row's groups
PROFILE_ACCESS	Allows a session to change its labels and privileges to those of a different user
WRITEUP	Allows users to set or raise only the level, within a row label, up to the maximum level authorized for the user. (Active only if LABEL_UPDATE is active.)
WRITEDOWN	Allows users to set or lower the level, within a row label, to any level equal to or greater than the minimum level authorized for the user. (Active only if LABEL_UPDATE is active.)
WRITEACROSS	Allows a user to set or change groups and compartments of a row label, but does not allow changes to the level. (Active only if LABEL_UPDATE is active.)

Special Access Privileges

A user's authorizations can be modified with any of four privileges:

- READ
- FULL
- COMPACCESS
- PROFILE_ACCESS

READ

A user with the READ privilege can read all data protected by the policy, regardless of the authorizations or session label. The user does not even need to have label authorizations. Note, in addition, that a user with READ privilege can *write* to any data rows for which he or she has write access, based on any label authorizations.

Note: However, access mediation is still enforced on UPDATE, INSERT, and DELETE operations.

Refer to Chapter 9, "Implementing Policy Enforcement Options and Labeling Functions", particularly

- "Overview of Policy Enforcement Options" on page 9-1,
- Table 9–2, " Policy Enforcement Options" on page 9-3, and
- "The Access Control Enforcement Options" on page 9-6.

This privilege is useful for system administrators who need to export data but who should not be allowed to change data. It is also useful for people who must run reports and compile information but not change data. The READ privilege enables optimal performance on SELECT statements, because the system behaves as though the Oracle Label Security policy were not even present.

FULL

The FULL privilege has the same effect and benefits as the READ privilege, with one difference. A user with the FULL privilege can also *write* to all the data. For a user with the FULL privilege, the READ and WRITE algorithms are not enforced.

Note that Oracle system and object authorizations are still enforced. For example, a user must still have SELECT on the application table. The FULL authorization turns off the access mediation check at the individual row level.

COMPACCESS

The COMPACCESS privilege allows a user to access data based on the row label's compartments, independent of the row label's groups. If a row label has no compartments, then access is determined by the group authorizations. However, when compartments do exist and access to them is authorized, then the group authorization is bypassed. This allows a privileged user whose label matches all the compartments of the data to access any data in any particular compartment, independent of what groups may own or otherwise be allowed access to the data.

Figure 3–9, "Label Evaluation Process for Read Access with COMPACCESS Privilege" shows the label evaluation process for read access with the COMPACCESS privilege. Note that if the data label is null or invalid, then the user is denied access.

Figure 3–9 Label Evaluation Process for Read Access with COMPACCESS Privilege

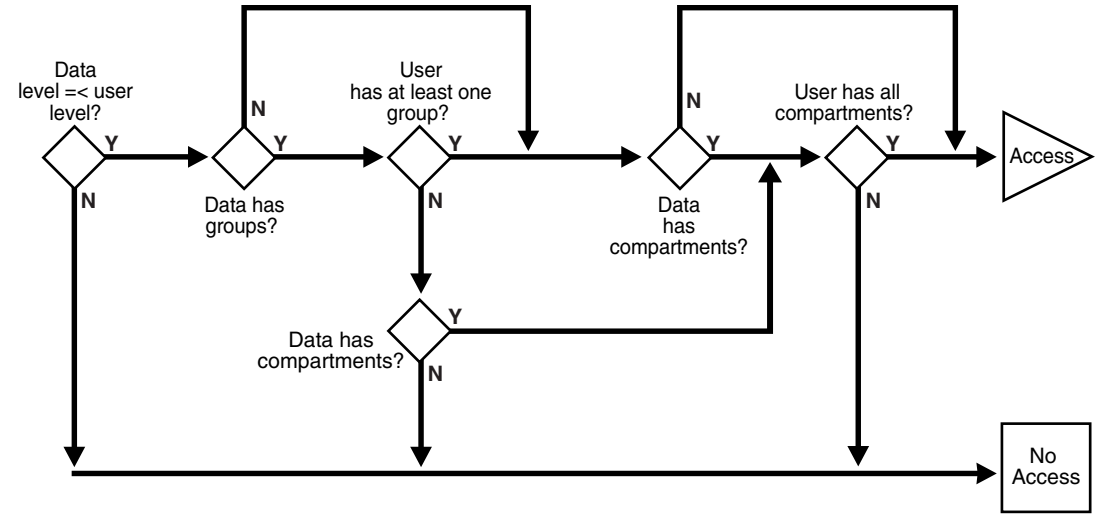
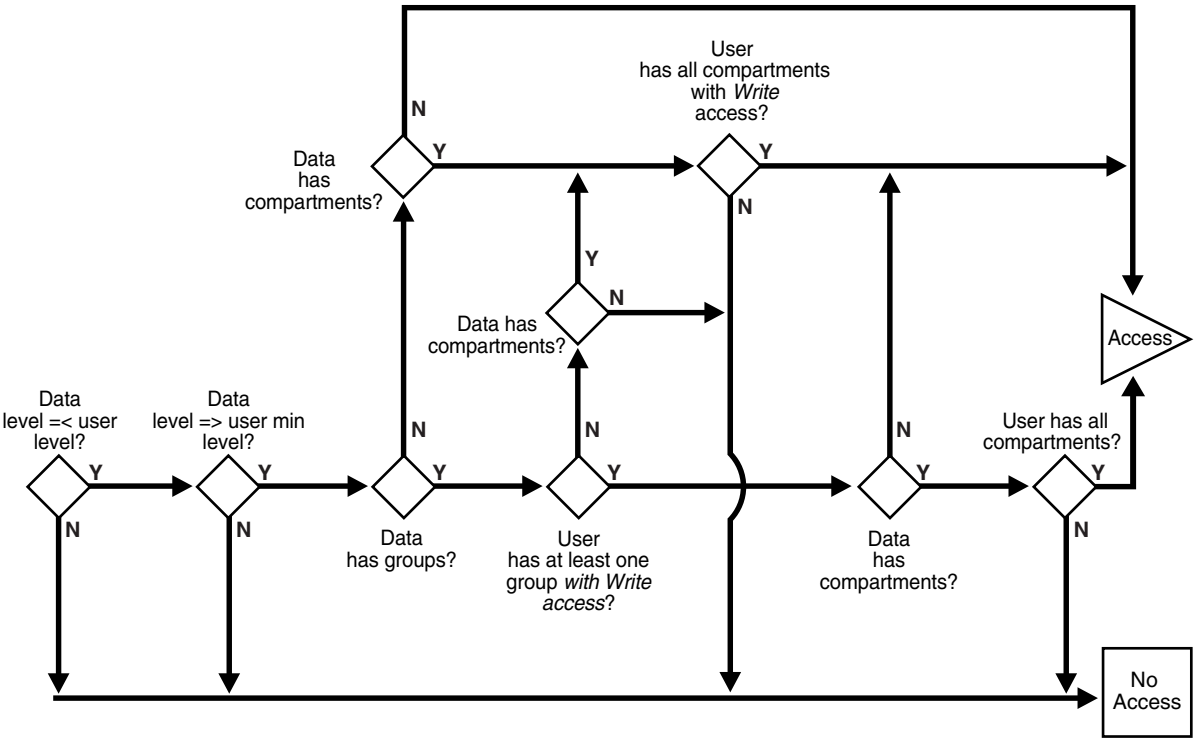


Figure 3–10, "Label Evaluation Process for Write Access with COMPACCESS Privilege" shows the label evaluation process for write access with COMPACCESS privilege. Note that if the data label is null or invalid, then the user is denied access.

Figure 3–10 Label Evaluation Process for Write Access with COMPACCESS Privilege



PROFILE_ACCESS

The PROFILE_ACCESS privilege allows a session to change its session labels and session privileges to those of a different user. This is a very powerful privilege,

because the user can potentially become a user with FULL privileges. This privilege cannot be granted to a trusted stored program unit.

Special Row Label Privileges

Once the label on a row has been set, Oracle Label Security privileges are required to modify the label. These privileges include WRITEUP, WRITEDOWN, and WRITEACROSS.

Note that the LABEL_UPDATE enforcement option must be on for these label modification privileges to be enforced. When a user updates a row label, the new label and old label are compared, and the required privileges are determined.

WRITEUP

The WRITEUP privilege enables the user to raise the level of data within a row, without compromising the compartments or groups. The user can raise the level up to his or her maximum authorized level.

For example, an authorized user can raise the level of a data row that has a level lower than his own minimum level. If a row is UNCLASSIFIED and the user's maximum level is SENSITIVE, then the row's level can be raised to SENSITIVE. It can be raised above the current session level, but it cannot change the compartments.

WRITEDOWN

The WRITEDOWN privilege enables the user to lower the level of data within a row, without compromising the compartments or groups. The user can lower the level to any level equal to or greater than his or her minimum authorized level.

WRITEACROSS

The WRITEACROSS privilege allows the user to change the compartments and groups of data, without altering its sensitivity level. This guarantees, for example, that SENSITIVE data remains at the SENSITIVE level, but at the same time enables the data's dissemination to be managed.

It lets the user change compartments and groups to anything that is currently defined as a valid compartment or group within the policy, while maintaining the level. With the WRITEACROSS privilege, a user with read access to one group (or more) can write to a different group without explicitly being given access to it.

System Privileges, Object Privileges, and Policy Privileges

Remember that Oracle Label Security privileges are different from the standard *Oracle Database* system and object privileges.

Table 3–4 Types of Privilege

Source	Privileges	Definition
Oracle Database	System Privileges	The right to run a particular type of SQL statement
	Object Privileges	The right to access another user's object
Oracle Label Security	Policy Privileges	The ability to bypass certain parts of the label security policy

Oracle Database enforces the discretionary access control privileges that a user has been granted. By default, a user has no privileges except those granted to the PUBLIC user group. A user must explicitly be granted the appropriate privilege to perform an operation.

For example, to read an object in *Oracle Database*, you must either be the object's owner, or be granted the SELECT privilege on the object, or be granted the SELECT ANY TABLE system privilege. Similarly, to update an object, you must either be the object's owner, or be granted the UPDATE privilege on the object, or be granted the UPDATE ANY TABLE privilege.

> **See Also:** For more information about which *Oracle Database* privileges are required to perform a certain operation and how to grant and revoke these discretionary access control privileges, see *Oracle Database Administrator's Guide*

Access Mediation and Views

Prior to accessing data through a view, the users must have the appropriate system and object privileges on the view. If the underlying table (on which the view is based) is protected by Oracle Label Security, then the user of the view must have authorization from Oracle Label Security to access specific rows of labeled data.

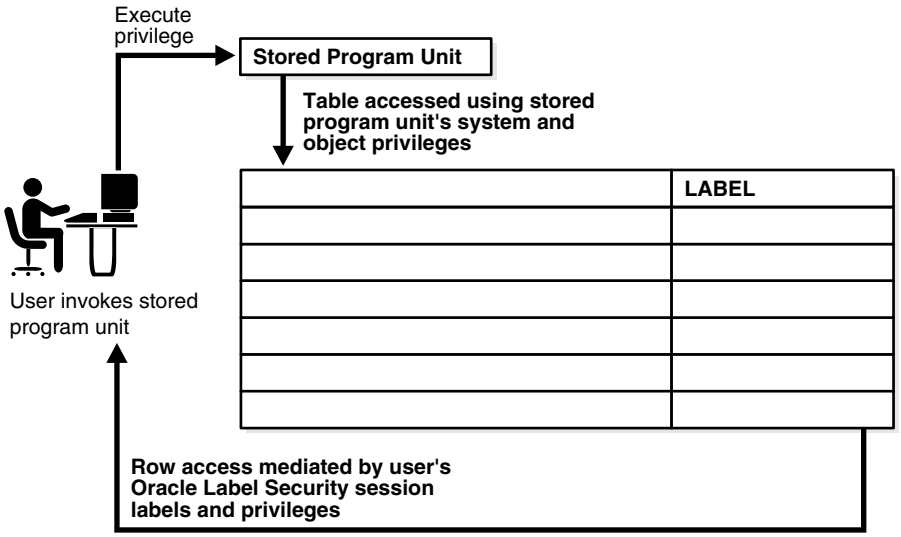
Access Mediation and Program Unit Execution

In *Oracle Database*, if User1 executes a procedure that belongs to User2, the procedure runs with User2's system and object privileges. However, any procedure executed by User1 runs with User1's own Oracle Label Security labels and privileges. This is true even when User1 executes stored program units owned by other users.

Figure 3–11, "Stored Program Unit Execution" illustrates this process:

- Stored program units run with the DAC privileges of the procedure's owner (User2).
- In addition, stored program units accessing tables protected by Oracle Label Security mediate access to data rows based on the label attached to the row, and the Oracle Label Security labels and privileges of the invoker of the procedure (User1).

Figure 3–11 Stored Program Unit Execution



Stored program units can become *trusted* when an administrator assigns them Oracle Label Security privileges. A stored program unit can be run with its own autonomous Oracle Label Security privileges rather than those of the user who calls it. For example, if you possess no Oracle Label Security privileges in your own right but run a stored program unit that has the WRITEDOWN privilege, then you can update labels. In this case, the privileges used are those of the stored program unit, and not your own.

Trusted program units can encapsulate privileged operations in a controlled manner. By using procedures, packages, and functions with assigned privileges, you may be able to access data that your own labels and privileges would not authorize. For example, to perform aggregate functions over all data in a table, not just the data visible to you, you might use a trusted program set up by an administrator. This way program units can thus perform operations on behalf of users, without the need to grant privileges directly to users.

> **See Also:** Chapter 11, "Administering and Using Trusted Stored Program Units"

Access Mediation and Policy Enforcement Options

An administrator can choose from among a set of policy enforcement options when applying an Oracle Label Security policy to individual tables. These options enable enforcement to be tailored differently for each database table. In addition to the access controls based on the labels, a SQL predicate can also be associated with each table. The predicate can further define which rows in the table are accessible to the user. Policy enforcement options and predicates are discussed in Chapter 9, "Implementing Policy Enforcement Options and Labeling Functions".

In cases where the label to be associated with a new or updated row should be automatically computed, an administrator can specify a labeling function when applying the policy. That function will thereafter always be invoked to provide the data labels written under that policy, because active labeling functions take precedence over any alternative means of supplying a label.

Except where noted, this guide assumes that all enforcement options are in effect.

See Also:

- "Using a Labeling Function" on page 9-9
- "Applying a Policy with SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY" on page 10-5

Working with Multiple Oracle Label Security Policies

This section describes aspects of using multiple policies.

Multiple Oracle Label Security Policies in a Single Database

Several Oracle Label Security policies may be protecting data in a single database. Each defined policy is associated with a set of labels used only by that policy. Data labels are constrained by the set of defined labels for each policy.

Each policy may protect a different table, but multiple policies can also apply to a single table. To access data, you must have label authorizations for all policies protecting that data. To access any particular row, you must be authorized by *all* policies protecting the table containing your desired rows. If you require privileges, then you may need privileges for all of the policies affecting your work.

Multiple Oracle Label Security Policies in a Distributed Environment

If you work in a distributed environment, where multiple databases may be protected by the same or different Oracle Label Security policies, your remote connections will also be controlled by Oracle Label Security.

See Also: Chapter 13, "Using Oracle Label Security with a Distributed Database"

Part II

Using Oracle Label Security Functionality

This part presents the following chapters, each discussing the indicated contents:

- Chapter 4, "Getting Started with Oracle Label Security"
- Chapter 5, "Working with Labeled Data"
- Chapter 6, "Oracle Label Security Using Oracle Internet Directory"

4

Getting Started with Oracle Label Security

Oracle Label Security (OLS) provides row-level security for your database tables. It protects data rows by labeling individual rows. If a user tries to access a data row protected by a policy, then he must have proper authorization as determined by the OLS label for the row.

This chapter helps you get started with OLS. It discusses the tasks involved in creating a simple OLS policy. It also uses a scenario to help you create and test a sample OLS policy. This chapter includes the following topics:

Installing OLS and Enabling the LBACSYS User

A default Oracle Database installation does not include Oracle Label Security (OLS). Use Oracle Universal Installer to install OLS in an existing database. OLS provides its own user account, `LBACSYS`, which you need to enable after the installation.

This section covers the following topics:

Installing Oracle Label Security

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

> **Note:** Before you run Oracle Universal Installer (OUI) to install Oracle Label Security, you should shut down the database instance. You should also shut down the corresponding database service if you are using Windows.
>
> In case you haven't shut down the database service, you would be prompted to do so during installation.

To install Oracle Label Security:

1. 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.

  The Welcome screen appears.

2. Click **Next**.

   The **Select Installation Method** screen appears.

3. Select **Advanced Installation**. Click **Next**.

   The **Select Installation Type** screen appears.

4. Select **Custom**. Click **Next**.

   The Specify Home Details screen appears.

5. Ensure that the correct Oracle base and Oracle home directories are selected. Click **Next**.

   At this point the installer verifies that your system meets the minimum requirements. Next, the Available Product Components screen is displayed.

6. Select the check box corresponding to **Oracle Label Security**. This option can be found under **Oracle Database 11g**, E**nterprise Edition Options**. Click **Next**.

   The **Summary** screen is displayed.

7. Review your choices and click **Install**.

   The progress screen is displayed.

8. The End of Installation screen is displayed. Click **Exit**.

Registering Oracle Label Security with the Database

After you complete the installation, you need to register Oracle Label Security with the database.

To register Oracle Label Security with the Database:

1. Start Database Configuration Assistant (DBCA).

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

     ```
     $ORACLE_HOME/bin/dbca
     ```

   - **Windows**: From the **Start** menu, click **All Programs**. Then click **Oracle - *ORACLE_HOME***, then **Configuration and Migration Tools**, and then **Database Configuration Assistant**.

   The Welcome screen appears.

2. Click **Next**.

   The Operations screen appears.

3. Select **Configure Database Options**. Click **Next**.

   The Database screen appears.

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

   The Database Content screen appears.

5. Select **Oracle Label Security**. Click **Next**.

The Connection Mode screen appears.

6. Select either **Dedicated Server Mode** or **Shared Server Mode**. Click **Finish**.

   A dialog box is displayed informing you that the operation will require the database to be restarted.

7. Click **OK**.

   A confirmation dialog box is displayed.

8. Click **OK**.

   The DBCA progress screen is displayed.

9. After the operation is complete, you are prompted to perform another operation. Click **No** to exit DBCA.

Enabling the LBACSYS User Account

The OLS installation process creates a default user account, `LBACSYS`, which has the privileges to manage OLS administration. By default, `LBACSYS` is created as a locked account with it's password expired. Your next step is to unlock `LBACSYS` and create a new password. You also need to grant `LBACSYS` the `SELECT ANY DICTIONARY` system privilege. This privilege allows `LBACSYS` to log in to Enterprise Manager.

To unlock LBACSYS and create a new password:

1. Log in to Database Control as the `SYSTEM` user.
2. Click the **Schema** tab.
3. Click **Users** under Users and privileges.

   The Users page appears

4. Select **LBACSYS**. Click **Edit**.

   The Edit User page appears.

5. Change the Status to **Unlocked**.
6. Enter a password in the **Enter Password** field. Reenter the password in the **Confirm Password** field.
7. Click the **System Privileges** tab.
8. Select the `SELECT ANY DICTIONARY` system privilege.
9. Click **Apply**.

Creating an OLS Policy

This section explores the following topics:

- Step 1: Creating the Policy
- Step 2: Creating Label Components for the Policy
- Step 3: Creating Data Labels for the Policy
- Step 4: Authorizing Users for the Policy
- Step 5: Applying the Policy to a Database Table
- Step 6: Adding Policy Labels to Table Rows

Step 1: Creating the Policy

You begin by defining a policy name, label column, and enforcement options for the policy.

To create a policy with default policy enforcement options:

1. Log in to Oracle Enterprise Manager Database Control using the `LBACSYS` account.
2. Click the **Server** tab.
3. Click **Oracle Label Security** under Security. The Label Security Policies page appears.
4. Click **Create** to start creating a new label security policy.

   The Create Label Security Policy page appears.
5. Define the policy's name, label column, and the default policy enforcement options.
   - **Name**: Enter a name for the policy, for example, `ACCESS_LOCATIONS`.
   - **Label Column**: Enter a name for the label column, for example, `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)`. You can also use an existing table column of the `NUMBER(10)` data type as the label column.
   - **Hide Label Column**: Select to hide the column. When you first create the policy, you may want to disable **Hide Label Column** during the development phase of the policy. When the policy is satisfactory and ready for use by users, hide the column so that it is transparent to applications.
   - **Enabled**: Toggle to enable or disable the policy.
   - **Enforcement Options**: The default policy enforcement options are used when the policy is applied. Ensure that these meet the needs of the application to which you are applying the policy.

     Select from the following options:
     - **Apply No Policy Enforcements (NO_CONTROL)**
     - **Apply Policy Enforcements**

       For all queries (READ_CONTROL)

       For Insert operations (INSERT_CONTROL)

       For Update Operations (UPDATE_CONTROL)

       Use session's default label for label column update (LABEL_DEFAULT)

       Operations that update the label column (LABEL_UPDATE)

       Update and Insert operations so that they are read accessible (CHECK_CONTROL)
6. Click **OK**.

   The new policy appears in the Oracle Label Security Policies page.

Step 2: Creating Label Components for the Policy

At this stage, you have created a container for the policy and have set enforcement options for it. Next, you need to create label components for the policy.

To create the label components:

1. In the Oracle Label Security Policies page, select the policy you just created. Click **Edit**.
2. In the Edit Label Security Policy page, select the **Label Components** tab.
3. Click **Add 5 Rows** under Levels to add levels for the policy. Enter a Long Name, Short Name, and Numeric Tag for each level that you create. The numeric tag corresponds to the sensitivity of the level. To create more levels, you can click **Add 5 Rows** again. Use the same steps to create compartments and rows. For compartments and groups, the numeric tags do not correspond to sensitivity.

   At a minimum, you must create one level, such as SECRET. Creating compartments and groups is optional.

   The level numbers indicate the level of sensitivity for their corresponding labels. A greater number implies greater sensitivity. Select a numeric range that can be expanded later on, in case your security policy needs more levels. For example, if you have created levels PUBLIC (7000) and SENSITIVE (8000), and you now want to create an intermediate level called CONFIDENTIAL, then you can assign the numeric value 7500 to this level.

   Compartments identify categories associated with data, providing a finer level of granularity within a level. For example, a single table might have data corresponding to different departments that you might like to separate using compartments. 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.
4. Click **Apply**.

Step 3: Creating Data Labels for the Policy

You are now ready to create data labels for the policy. Each data label must have exactly one level associated with it. You can optionally add one or more compartments and groups to the label.

To create a data label, you need to assign a numeric tag to the label. Later on, the tag number will be stored in the security column when you apply the policy to a table. The label tag is not linked to the sensitivity (level) of the label. It is only used to identify the label.

To create data labels for each level:

1. In the Label Security Policies page, select the policy that needs to have labels linked to levels.
2. In the **Actions** box, select Data Labels. Click **Go**.

   The Data Labels page appears.
3. Click **Add**.

   The Create Data Label page appears.
4. Enter the following information:

- **Numeric Tag**: Enter a number that uniquely identifies the label. This number should be unique across all policies.
- **Level**: Select a level from the list.

5. You can optionally select Compartments to add to the label. To add compartments, click **Add** under Compartments. Select the compartments to be added to the label. Click **Select** to add the compartments.
6. You can optionally select Groups to add to the label. To add groups, click **Add** under Groups. Select the groups to be added to the label. Click **Select** to add the groups.
7. Click **OK** in the Create Data Label page.

   The data label appears in the Data Labels page.
8. Repeat steps 3 to 7 to create more data labels.

Step 4: Authorizing Users for the Policy

You are now ready to authorize users for the Oracle Label Security policy.

To authorize users for the OLS policy:

1. In the Label Security Policies page, select the policy that needs authorization.
2. In the **Actions** box, select Authorization. Click **Go**.

   The Create User page appears.
3. Add users as follows:
   - Under Database Users, click **Add**. In the Search and Select window, select users that you want and then click **Select**.
   - Under Non Database Users, click **Add 5 Rows**, and then add the user names of the non-database users that you want to add. Most application users are considered non-database users. A non-database 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 non-database 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 real database user.
4. In the Create User page, select the user that you want to authorize. Click **Next**. If you have multiple users that need the same authorizations, then select all users who need the same authorizations. Click **Next**.

   The Privileges step appears.
5. Next, you can assign privileges to the user you selected in the preceding step. Privileges allow a database user to bypass certain controls enforced by the policy. Select the privileges you want to grant. Click **Next**.

   If you do not wish to assign any privilege to the user, click **Next** without selecting any privileges.

   The Labels, Compartments, and Groups step appears.
6. Next, you need to create the user label for the user. Under Levels, use the flashlight icon to select data to enter for the following fields:

- **Maximum Level**: Enter the highest level for read and write access for this user.
- **Minimum Level**: Enter the lowest level for write access.
- **Default Level**: Enter the default level when the user logs in.

  This value is equal to or greater than the minimum level and equal to or less than the maximum level.
- **Row Level**: Enter the level given to the row when user writes to the table.

7. Click **Add** under Compartments, to add compartments to the user label. Select the compartments to add. Click **Select**.
8. For each compartment that you add, you can select the following properties:
   - **Write**: Allows the user to write to data that has the compartment as part of it's label
   - **Default**: Adds the compartment to the user's default session label
   - **Row**: Adds the compartment to the data label when the user writes to the table
9. Click **Add** under Groups, to add groups to the user label. Select the groups and click **Select**.
10. For each group that you add, you can select the following properties:
    - **Write**: Allows the user to write to data that has the group as part of it's label
    - **Default**: Adds the group to the user's default session label
    - **Row**: Adds the group to the data label when the user writes to the table
11. Click **Next**.

    The Audit step appears.
12. Next, you can choose to set the policy audit options for the selected user. You can set audit options for the following operations:
    - Policy Applied:

      Audit On Success By audits successful application of the policy to a table or schema. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed application of the policy to a table or schema. Select ACCESS to audit by access or SESSION to audit by session.
    - Policy Removed:

      Audit On Success By audits successful removal of the policy from a table or schema. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed removal of the policy from a table or schema. Select ACCESS to audit by access or SESSION to audit by session.
    - Labels And Privileges Set:

      Audit On Success By audits successful setting of user authorizations and privileges. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed setting of user authorizations and privileges. Select ACCESS to audit by access or SESSION to audit by session.
    - All Policy Specific Privileges:

Audit On Success By audits successful use of policy privileges. Select ACCESS to audit by access or SESSION to audit by session.

Audit On Failure By audits failed use of policy privileges. Select ACCESS to audit by access or SESSION to audit by session.

13. Click **Next**.
14. You can review the policy authorization settings. Click **Finish** to create the policy authorization. Alternatively, you can click **Back** to modify the authorization settings.

Step 5: Applying the Policy to a Database Table

Next, apply the OLS policy to a database table.

To apply the policy to a database table:

1. In the Label Security Policies page, select the policy that needs to be applied to a table.
2. Select Apply from the **Actions** box. Click **Go**.

   The Apply page appears.
3. Select the **Tables** tab to apply the policy to a table.

   Note: Select the **Schemas** tab if you are applying the policy to a schema.The process is same as applying the policy to a table.

4. Click **Create**.

   The Add Table page appears.
5. Next to the **Table** box, click the flashlight icon.
6. In the Search and Select window, enter the following information under Search:
   - **Schema**: Enter the name of the schema in which the table appears. Leaving this field empty displays tables in all schemas.
   - **Name**: Optionally, enter the name of the table. Leaving this box empty displays all the tables within the schema.

   To narrow the search by using wildcards, use the percent (%) sign. For example, enter `O%` to search for all tables beginning with the letter O.
7. Select the table and click **Select**.

   The Add Table page appears.
8. Enter the following information:
   - **Policy Enforcement Options**: Select enforcement options as needed. These options will apply to the table on top of the enforcement options that you selected when you created the policy in Step 1: Creating the Policy.

     To make no change from those enforcement options, that is, to use the same enforcement options created earlier, select **Use Default Policy Enforcement**. To add more enforcement options, select from the other options listed.
   - **Labeling Function**: Optionally, specify a labeling function to automatically compute the label to be associated with a new or updated row. That function is always invoked thereafter to provide the data labels written under that policy,

because active labeling functions take precedence over any alternative means of supplying a label.

- **Predicate**: Optionally, specify an additional predicate to combine (using `AND` or `OR`) with the label-based predicate for `READ_CONTROL`.

9. Click **OK**.

Step 6: Adding Policy Labels to Table Rows

After you have applied a policy to a table, you need to add data labels to the rows in the table. These labels are stored in the policy label column created in the table. The user updating the table needs to have the FULL security privilege for the policy. This user is normally the owner of the table.

To grant the table owner FULL privilege for the OLS 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** box. Click **Go**.

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

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

   The Search and Select window appears.
6. Select the check box corresponding to the user that owns the table. Click **Select**.

   The Create User page lists the user that was added.
7. Click **Next**.

   The Privileges step appears.
8. Select the following privilege:

   Bypass all Label Security checks (FULL)

   Click **Next**.

   The Labels, Compartments, and Groups step appears.
9. Click **Next**.

   The Audit step appears.
10. Click **Next**.

    The Review step appears.
11. Click **Finish**.

To add data labels to the table:

- In SQL*Plus, enter an `UPDATE` statement using the following syntax:

```
UPDATE LOCATIONS
SET OLS_COLUMN = CHAR_TO_LABEL('OLS_POLICY','DATA_LABEL')
WHERE UPPER(TABLE_COLUMN) IN (COLUMN_DATA);
```

For example, suppose LABCSYS has created a policy called ACCESS_LOCATIONS and wants to add the label SENS to the cities Beijing, Tokyo, and Singapore in the HR.LOCATIONS table. The policy label column is called ROW_LABEL. The UPDATE statement is as follows:

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

If you want to check that your labels really made it into the table, run the the following SELECT statement:

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

Creating a Sample OLS Policy

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, KPARTNERS, and LDORAN will have access to specific rows within this table, based on the cities listed in the LOCATIONS table.

The HR.LOCATIONS is described as follows:

```
SQL> DESCRIBE locations

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

You will follow these steps to complete this example:

- Step 1: Creating Users for the Oracle Label Security Example
- Step 2: Creating the ACCESS_LOCATIONS Policy
- Step 3: Defining the ACCESS_LOCATIONS Policy-Level Components
- Step 4: Creating the ACCESS_LOCATIONS Policy Data Labels
- Step 5: Creating the ACCESS_LOCATIONS Policy User Authorizations
- Step 6: Applying the ACCESS_LOCATIONS Policy to the HR.LOCATIONS Table
- Step 7: Adding Policy Labels to Table Data
- Step 8: Testing the ACCESS_LOCATIONS Policy
- Step 9: Removing the Components for This Example (Optional)

Step 1: Creating 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 the EMP_ROLE Role

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

To create the role EMP_ROLE:

1. Log in to Database Control as the `SYSTEM` user.
2. Click the **Schema** tab.
3. Under Users and Privileges, click **Roles**.

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

   The Create Role page appears.
5. Enter EMP_ROLE in the **Name** field. Leave **Authentication** set to None.
6. Click the **Object Privileges** tab.
7. Select Table from the **Select Object Type** box. 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. Under **Available Privileges**, move SELECT to the **Selected Privileges** list.
9. Click **OK**.
10. Click **OK** in the Create Role page.

Creating the Users SKING, KPARTNERS, and LDORAN

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 Sales president, so he has full read access to the `HR.LOCATIONS` table. Karen Partners (KPARTNERS) 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 SKING, KPARTNERS, and LDORAN:

1. Log in to Database Control as `SYSTEM`.
2. Click the **Schema** tab.
3. Under Users and Privileges, 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**: there4all
- **Default Tablespace**: USERS
- **Temporary Tablespace**: TEMP
- **Roles**: Select the **Roles** tab and grant the EMP_ROLE role to SKING. Select the **Default** check box.
- **System Privileges**: Select the **System Privileges** tab and grant the `CREATE SESSION` system privilege.

6. Click **OK**.
7. In the Users page, select SKING, select Create Like in the **Actions** box, and then click **Go**.

   The Create User page appears.
8. Create accounts for KPARTNERS and LDORAN, with eager2please as the password for KPARTNERS and ready2go as the password for LDORAN.

   You only need to create their names and passwords. You do not need to grant roles or system privileges to them. Their roles and system privileges, defined in `SKING`'s account, are automatically created.

At this stage, you have created three users who have identical privileges.

Step 2: Creating the ACCESS_LOCATIONS Policy

The policy is the container for the label components that you will create later.

To create the ACCESS_LOCATIONS policy:

1. Log in to Database Control as the `LBACSYS` user.
2. Click the **Server** tab.
3. Click **Oracle Label Security** under 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
   - **Hide Label Column**: Deselect this check box so that the label column will not be hidden.

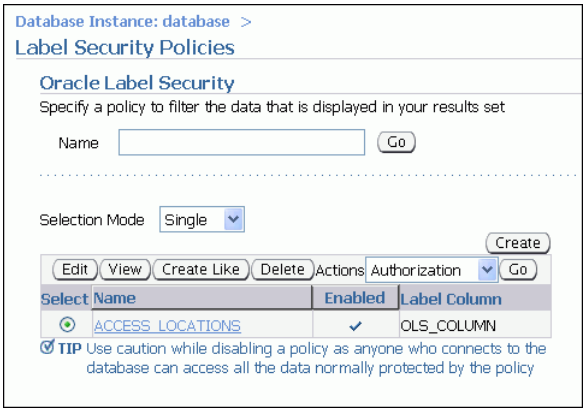
     Usually, it should be hidden, but during the development phase, you may want to have it visible so that you can check it. After the policy is created and working, you should hide this column for greater security.
   - **Enabled**: Toggle to enable this policy. (It should be enabled by default.)
   - **Enforcement Options**: Select **Apply Policy Enforcements**, and then select the following options:

     For all queries (READ_CONTROL)

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 3: Defining the ACCESS_LOCATIONS Policy-Level Components

You can now define the label components 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. Click **Edit**.

   The Edit Label Security Policy page appears.

2. Select the **Label Components** tab.

3. Under Levels, click **Add 5 Rows**, and add the following levels:

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

4. Click **Apply**.

Step 4: Creating the ACCESS_LOCATIONS Policy Data Labels

In this step, you create data labels corresponding to the levels that you created in the last step.

To create the data labels:

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

2. Select the **ACCESS_LOCATIONS** policy.

3. Select Data Labels from the **Actions** list. 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.

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**.

Step 5: Creating 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. Select Authorization from the **Actions** box. Click **Go**.

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

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

   The Search and Select window appears.
6. Select the check box for user **SKING** and then click **Select**.

   The Create User page lists user SKING.

7. Click **Next**.

The Privileges step appears.

8. Click **Next**.

   The Labels, Compartments and Groups step appears.

9. Set the following levels for the user SKING:
   - **Maximum Level**: SENS (for SENSITIVE)
   - **Minimum Level**: CONF (for CONFIDENTIAL)
   - **Default Level**: SENS
   - **Row Level**: SENS

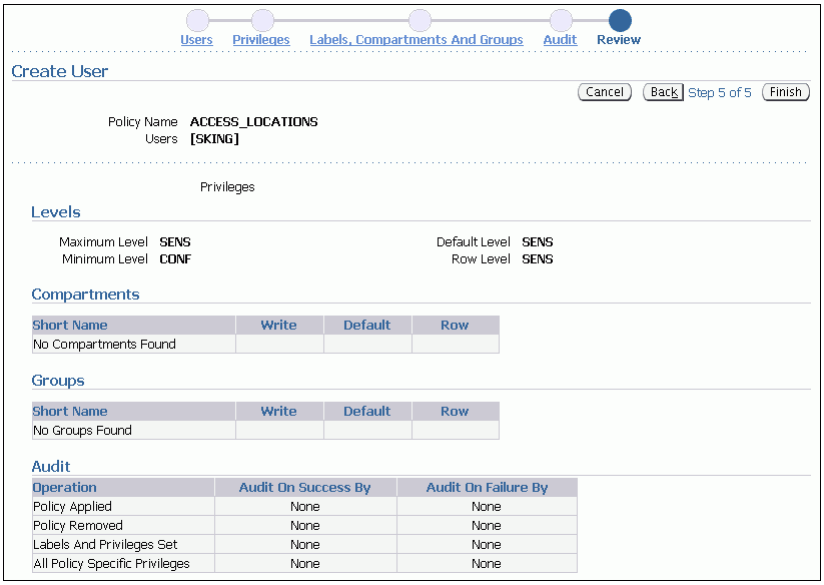
   This allows SKING to read CONFIDENTIAL and SENSITIVE data.

10. Click **Next**.

    The Audit step appears.

11. Ensure that all of the audit operations are set to None, and then click **Next**.

    The Review step appears.



12. The Review step lists all the authorization settings you have selected. Ensure that the settings are correct, and then click **Finish**.

13. Repeat these steps to create the following authorizations for user KPARTNERS, so that she can read confidential and public data in `HR.LOCATIONS`.
    - **Labels, Compartments And Groups**: Set all four levels to the following:
      - **Maximum Level**: CONF (for CONFIDENTIAL)
      - **Minimum Level**: PUB (for PUBLIC)
      - **Default Level**: CONF
      - **Row Level**: CONF
    - **Audit**: Set all to None.

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

- **Labels, Compartments And Groups**: Set all four levels to PUB.
- **Audit**: Set all to None.

Step 6: Applying 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:

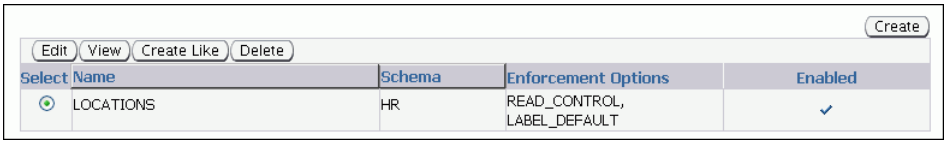
1. Return to the Label Security Policies page by selecting the **Label Security Policies** link.
2. Select the **ACCESS_LOCATIONS** policy.
3. Select Apply from the **Actions** box. Click **Go**.

   The Apply page appears.
4. Ensure that the **Tables** tab is selected. 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 7: Adding Policy Labels to Table Data

After you have applied the ACCESS_LOCATIONS policy to the `HR.LOCATIONS` table, you need to label the rows in the table. `HR` is the owner of the LOCATIONS table. `HR` needs to have the `FULL` security privilege for the ACCESS_LOCATIONS policy, in order to successfully update the LOCATIONS table with the policy labels.

- Granting FULL Privilege to the Owner of the Application Table
- Updating the OLS_COLUMN Table in HR.LOCATIONS

Granting FULL Privilege to the Owner of the Application Table

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

To grant HR FULL Privilege for the OLS 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** box. Click **Go**.

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

   The Create 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 following privilege:

   Bypass all Label Security checks (FULL)

   Click **Next**. The Labels, Compartments, and Groups step appears.
9. Click **Next**. The Audit step appears.
10. Click **Next**. The Review step appears.
11. Click **Finish**.

Updating the OLS_COLUMN Table in HR.LOCATIONS

The user HR can now update the HR.LOCATIONS table with the appropriate labels.

To update the OLS_COLUMN table in HR.LOCATIONS:

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

```
CONNECT HR
Enter password:
```

   If you receive an error message saying that HR is locked, you can unlock the account and reset its password by entering the following statements:

```
CONNECT system
Enter password: sys_password
ALTER USER HR ACCOUNT UNLOCK IDENTIFIED BY password
CONNECT hr
Enter 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 query:

```
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 will not show the label column if it is hidden. This feature allows the label column to remain transparent to applications. An application that was designed before the label column was added will not know about the label column and will never see it.

Step 8: Testing 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 there4all.

```
CONNNECT SKING
Enter password: there4all
```

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 rows that are labeled PUB, CONF, and SENS.

3. Repeat these steps for users KPARTNERS and LDORAN.

   The password for KPARTNERS is eager2please and the password for LDORAN is ready2go.

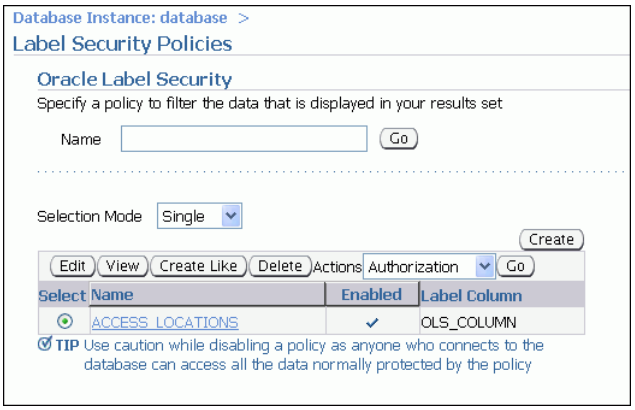
   KPARTNERS can access rows labeled CONF and PUB, and LDORAN is able to access the rows labeled PUB.

Step 9: Removing the Components for This Example (Optional)

If you want, 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 the **Schema** tab.
3. Click **Users** under Users and Privileges.
4. Select user **KPARTNERS** and then click **Delete**.
5. In the Confirmation page, click **Yes**.
6. Repeat Step 4 and Step 5 for users LDORAN and SKING.
7. Log out of Database Control, and then log back in as the `LABCSYS` user.
8. Click the **Server** tab.
9. Click **Oracle Label Security** under Security.
10. In the Label Security Policies page, in the **Name** field, enter `ACCESS%` and then click **Go**.



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

    The Confirmation page appears.

12. If you select Drop Column, the OLS_COLUMN policy column is also dropped from the `HR.LOCATIONS` table. Click **Yes** to delete the policy.

5

Working with Labeled Data

This chapter explains how to

- Use Oracle Label Security features to manage labeled data
- View the value of security attributes for a session
- Change the value of those session attributes

The chapter contains these sections:

> **Note:** Many of the examples in this book use the *HUMAN_RESOURCES* sample policy. Its policy name is *HR* and its policy label column is *HR_LABEL*. Unless otherwise noted, the examples assume that the SQL statements are performed on rows within the user's authorization and with full Oracle Label Security policy enforcement in effect.

The Policy Label Column and Label Tags

This section explains how policy label columns in a table or schema are created and filled, using these topics:

The Policy Label Column

Each policy that is applied to a table creates a column in the database. By default, the data type of the policy label column is NUMBER.

Note: The act of creating a policy does not in itself have any effect on tables or schemas. It only applyies the policy to a table or schema. Refer to these sections:

- "Creating a Policy with SA_SYSDBA.CREATE_POLICY" on page 7-11
- Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY on page 10-3
- Applying a Policy with SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY on page 10-5

Each row's label for that policy is represented by a tag in that column, using the numeric equivalent of the character-string label value. The label tag is automatically generated when the label is created, unless the administrator specifies the tag manually at that time.

The automatic label generation follows the rules established by the administrator while defining the label components, as described in Chapter 2, "Understanding Data Labels and User Labels".

Hiding the Policy Label Column

The administrator can decide not to display the column representing a policy by applying the HIDE option to the table. After a policy using HIDE is applied to a table, a user executing a SELECT * or performing a DESCRIBE operation will not see the policy label column. If the policy label column is not hidden, then the label tag is displayed as data type NUMBER. Refer to "The HIDE Policy Column Option" on page 9-5.

Example 1: Numeric Column Data Type (NUMBER)

```
SQL> describe emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- --------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              CHAR(10)
 JOB                                                CHAR(9)
 MGR                                                NUMBER(4)
 SAL                                                NUMBER(7,2)
 DEPTNO                                    NOT NULL NUMBER(2)
 HR_LABEL                                           NUMBER(10)
```

Example 2: Numeric Column Data Type with Hidden Column

Notice that in this example, the HR_LABEL column is *not* displayed.

```
SQL> describe emp;
 Name                                      Null?    Type
 ----------------------------------------- -------- --------
 EMPNO                                     NOT NULL NUMBER(4)
 ENAME                                              CHAR(10)
 JOB                                                CHAR(9)
 MGR                                                NUMBER(4)
 SAL                                                NUMBER(7,2)
 DEPTNO                                    NOT NULL NUMBER(2)
```

Label Tags

As noted in Chapter 2, "Understanding Data Labels and User Labels", the administrator first defines a set of label components to be used in a policy. When creating labels, the administrator specifies the set of valid combinations of components that can make up a label, that is, a level optionally combined with one or more groups or compartments. Each such valid label within a policy is uniquely identified by an associated numeric tag assigned by the administrator or generated automatically upon its first use. Manual definition has the advantage of allowing the administrator to control the ordering of label values when they are sorted or logically compared.

However, label tags must be unique across all policies in the database. When you use multiple policies in a database, you cannot use the same numeric label tag in different policies. Remember that each label tag uniquely identifies one label, and that numeric tag is what is stored in the data rows, not the label's character-string representation.

This section contains these topics:

- Manually Defining Label Tags to Order Labels
- Manually Defining Label Tags to Manipulate Data
- Automatically Generated Label Tags

Manually Defining Label Tags to Order Labels

By manually defining label tags, the administrator can implement a data manipulation strategy that permits labels to be meaningfully sorted and compared. To do this, the administrator predefines all of the labels to be associated with protected data, and assigns to each label a meaningful label tag value. Manually assigned label tags can have up to eight digits. The value of a label tag must be greater than zero.

It may be advantageous to implement a strategy in which label tag values are related to the numeric values of label components. In this way, you can use the tags to group data rows in a meaningful way. This approach, however, is not mandatory. It is good practice to set tags for labels of higher sensitivity to a higher numeric value than tags for labels of lower sensitivity.

Table 5–1 illustrates a set of label tags that have been assigned by an administrator. Notice that, in this example, the administrator has based the label tag value on the numeric form of the levels, compartments, and rows that were discussed in Chapter 2, "Understanding Data Labels and User Labels".

Table 5–1 Administratively Defined Label Tags (Example)

Label Tag	Label String
10000	P
20000	C
21000	C:FNCL
21100	C:FNCL,OP
30000	S
31110	S:OP:WR
40000	HS
42000	HS:OP

In this example, labels with a level of PUBLIC begin with "1", labels with a level of CONFIDENTIAL begin with "2", labels with a level of SENSITIVE begin with "3", and labels with a level of HIGHLY_SENSITIVE begin with "4".

Labels with the FINANCIAL compartment then come in the 1000 range, labels with the compartment OP are in the 1100 range, and so on. The tens place is used to indicate the group WR, for example.

Another strategy might be completely based on groups, where the tags might be 3110, 3120, 3130, and so on.

Note, however, that label tags identify the *whole* label, independent of the numeric values assigned for the individual label components. The label tag is used as a whole integer, not as a set of individually evaluated numbers.

Manually Defining Label Tags to Manipulate Data

An administratively defined label tag can serve as a convenient way to reference a complete label string (that is, a particular combination of label components). As illustrated in Table 5–1, for example, the tag "31110" could stand for the complete label string "S:OP:WR".

Label tags can be used as a convenient way to partition data. For example, all data with labels in the range 1000 - 1999 could be placed in tablespace A, all data with labels in the range 2000 - 2999 could be placed in tablespace B, and so on.

This simplified notation also comes in handy when there is a finite number of labels and you need to perform various operations upon them. Consider a situation in which one company hosts a human resources system for many other companies. Assume that all users from Company Y have the label "C:ALPHA:CY", for which the tag "210" has been set. To determine the total number of application users from Company Y, the host administrator can enter:

```
SELECT * FROM tab1
  WHERE hr_label = 210;
```

Automatically Generated Label Tags

Dynamically generated label tags, illustrated in Table 5–2, have 10 digits, with no relationship to numbers assigned to any label component. There is no way to group the data by label.

Table 5–2 Generated Label Tags (Example)

Label Tag	Label String
100000020	P
100000052	C
100000503	C:FNCL
100000132	C:FNCL,OP
100000003	S
100000780	S:OP:WR
100000035	HS
100000036	HS:OP

See Also:

- "Creating a Valid Data Label with SA_LABEL_ADMIN.CREATE_LABEL" on page 7-19
- "Planning a Label Tag Strategy to Enhance Performance" on page 14-6

Assigning Labels to Data Rows

For rows that are being inserted, refer to Inserting Labeled Data on page 5-11.

For existing data rows, labels can be assigned by a labeling function that you create. In such a function, you specify the exact table and row conditions defining what label to insert. The function can be named in the call to apply a policy to a table or schema, or in an update by the administrator.

See Also:

- "Using a Labeling Function" on page 9-9
- Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY on page 10-3.
- "Applying a Policy with SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY" on page 10-5

Presenting the Label

When you retrieve labels, you do not automatically obtain the character string value. By default, the label tag value is returned. Two label manipulation functions enable you to convert the label tag value to and from its character string representation:

- Converting a Character String to a Label Tag, with CHAR_TO_LABEL
- Converting a Label Tag to a Character String, with LABEL_TO_CHAR

Converting a Character String to a Label Tag, with CHAR_TO_LABEL

Use the CHAR_TO_LABEL function to convert a character string to a label tag. This function returns the label tag for the specified character string.

Syntax:

```
FUNCTION CHAR_TO_LABEL (
     policy_name     IN VARCHAR2,
     label_string    IN VARCHAR2)
RETURN NUMBER;
```

Example:

```
INSERT INTO emp (empno,hr_label)
VALUES (999, CHAR_TO_LABEL('HR','S:A,B:G5');
```

Here, `HR` is the label policy name, `S` is a sensitivity level, `A`, `B` compartments, and `G5` a group.

Converting a Label Tag to a Character String, with LABEL_TO_CHAR

When you query a table or view, you automatically retrieve all of the rows in the table or view that satisfy the qualifications of the query and are dominated by your label. If

the policy label column is not hidden, then the label tag value for each row is displayed. You must use the LABEL_TO_CHAR function to display the character string value of each label.

Note that all conversions must be explicit. There is no automatic casting to and from tag and character string representations.

Syntax:

```
FUNCTION LABEL_TO_CHAR (
     label                 IN NUMBER)
RETURN VARCHAR2;
```

LABEL_TO_CHAR Examples

The examples that follow illustrate the use of LABEL_TO_CHAR.

Example 1: To retrieve the label of a row from a table or view, specify the policy label column in the SELECT statement as follows:

```
SELECT label_to_char (hr_label) AS label, ename FROM tab1;
  WHERE ename = 'RWRIGHT';
```

This statement returns the following:

```
LABEL          ENAME
------------   ----------
S:A,B:G1       RWRIGHT
```

Example 2: You can also specify the policy label column in the WHERE clause of a SELECT statement. The following statement displays all rows that have the policy label `S:A,B:G1`

```
SELECT label_to_char (hr_label) AS label,ename FROM emp
  WHERE hr_label = char_to_label ('HR', 'S:A,B:G1');
```

This statement returns the following:

```
LABEL           ENAME
-------------   ---------
S:A,B:G1        RWRIGHT
S:A,B:G1        ESTANTON
```

Alternatively, you could use a more flexible statement to look up data that contains the string "S:A,B:G1" anywhere in the text of the HR_LABEL column:

```
SELECT label_to_char (hr_label) AS label,ename FROM emp
  WHERE label_to_char (hr_label) like '%S:A,B:G1%';
```

If you do not use the LABEL_TO_CHAR function, then you will see the label tag.

Example 3: The following example is with the numeric column data type (NUMBER) and dynamically generated label tags, but without using the LABEL_TO_CHAR function. If you do not use the LABEL_TO_CHAR function, then you will see the label tag.

```
SQL> select empno, hr_label from emp
     where ename='RWRIGHT';

EMPNO      HR_LABEL
---------- ----------
7839       1000000562
```

Retrieving All Columns from a Table When the Policy Label Column Is Hidden

If the policy label column is hidden, then it is not automatically returned when you select all columns from a table using the SELECT * command. You must explicitly specify that you want to retrieve the label. For example, to retrieve all columns from the DEPT table (including the policy label column in its character representation), enter the following:

```
SQL> column label format a10
SQL> select label_to_char (hr_label) as label, dept.*
  2  from dept;
```

Running these SQL statements returns the following data:

Table 5–3 Data Returned from Sample SQL Statements re Hidden Column

LABEL	DEPTNO	DNAME	LOC
L1	10	ACCOUNTING	NEW YORK
L1	20	RESEARCH	DALLAS
L1	30	SALES	CHICAGO
L1	40	OPERATIONS	BOSTON

By contrast, if you do not explicitly specify the HR_LABEL column, the label is not displayed at all. Note that while the policy column name is on a policy basis, the HIDE option is on a table-by-table basis.

See Also: "The HIDE Policy Column Option" on page 9-5

Filtering Data Using Labels

During the processing of SQL statements, Oracle Label Security makes calls to the security policies defined in the database by the create and apply procedures discussed in Chapter 7, "Creating an Oracle Label Security Policy" and Chapter 10, "Applying Policies to Tables and Schemas". For SELECT statements, the policy filters the data rows that the user is authorized to see. For INSERT, UPDATE, and DELETE statements, Oracle Label Security permits or denies the requested operation, based on the user's authorizations.

This section contains these topics:

- Using Numeric Label Tags in WHERE Clauses
- Ordering Labeled Data Rows
- Ordering by Character Representation of Label
- Determining Upper and Lower Bounds of Labels
- Merging Labels with the MERGE_LABEL Function

See Also: "Partitioning Data Based on Numeric Label Tags" on page 14-7

Using Numeric Label Tags in WHERE Clauses

This section describes techniques of using numeric label tags in WHERE clauses of SELECT statements.

When using labels in the NUMBER format, the administrator can set up labels so that a list of their label tags distinguishes the different levels. Comparisons of these numeric label tags can be used for ORDER BY processing, and with the logical operators.

For example, if the administrator has assigned all UNCLASSIFIED labels to the 1000 range, all SENSITIVE labels to the 2000 range, and all HIGHLY_SENSITIVE labels to the 3000 range, then you can list all SENSITIVE records by entering:

```
SELECT * FROM emp
WHERE hr_label BETWEEN 2000 AND 2999;
```

To list all SENSITIVE and UNCLASSIFIED records, you can enter:

```
SELECT * FROM emp
WHERE hr_label <3000;
```

To list all HIGHLY_SENSITIVE records, you can enter:

```
SELECT * FROM emp
WHERE hr_label=3000;
```

Note: Remember that such queries have meaning only if the administrator has applied a numeric ordering strategy to the label tags that he or she originally assigned to the labels. In this way, the administrator can provide for convenient dissemination of data. If, however, the label tag values are generated automatically, then there is no intrinsic relationship between the value of the tag and the order of the labels.

Alternatively, you can use dominance relationships to set up an ordering strategy.

See Also: "Using Dominance Functions" on page A-2

Ordering Labeled Data Rows

You can perform an ORDER BY referencing the policy label column to order rows by the numeric label tag value that the administrator has set. For example:

```
SELECT * from emp
ORDER BY hr_label;
```

Notice that no functions were necessary in this statement. The statement made use of label tags set up by the administrator.

Note: Again, such queries have meaning only if the administrator has applied a numeric ordering strategy to the label tags originally assigned to the labels.

Ordering by Character Representation of Label

Using the LABEL_TO_CHAR function, you can order data rows by the character representation of the label. For example, the following statement returns all rows sorted by the text order of the label:

```
SELECT * FROM emp
ORDER BY label_to_char (hr_label);
```

Determining Upper and Lower Bounds of Labels

This section describes the Oracle Label Security functions that determine the least upper bound or the greatest lower bound of two or more labels. Two single-row functions operate on each row returned by a query. They return one result for each row.

- Finding Least Upper Bound with LEAST_UBOUND
- Finding Greatest Lower Bound with GREATEST_LBOUND

> **Note:** In all functions that take multiple labels, the labels must all belong to the same policy.

Finding Least Upper Bound with LEAST_UBOUND

The LEAST_UBOUND (LUBD) function returns a character string label that is the least upper bound of *label1* and *label2:* that is, the one label that dominates both. The least upper bound is the highest level, the union of the compartments in the labels, and the union of the groups in the labels. For example, the least upper bound of HIGHLY_SENSITIVE:ALPHA and SENSITIVE:BETA is HIGHLY_SENSITIVE:ALPHA,BETA.

Syntax:

```
FUNCTION LEAST_UBOUND (
     label1                    IN NUMBER,
     label2                    IN NUMBER)
RETURN VARCHAR2;
```

The LEAST_UBOUND function is useful when joining rows with different labels, because it provides a high water mark label for joined rows.

The following query compares each employee's label with the label of his or her department, and returns the higher label, whether it be in the EMP table or the DEPT table.

```
SELECT ename,dept.deptno,
  LEAST_UBOUND(emp.hr_label,dept.hr_label) as label
  FROM emp, dept
  WHERE emp.deptno=dept.deptno;
```

This query returns the following data:

Table 5–4 Data Returned from Sample SQL Statements re Least_UBound

ENAME	DEPTNO	LABEL
KING	10	L3:M:D10
BLAKE	30	L3:M:D30
CLARK	10	L3:M:D10
JONES	20	L3:M:D20
MARTIN	30	L2:E:D30

Finding Greatest Lower Bound with GREATEST_LBOUND

The GREATEST_LBOUND (GLBD) function can be used to determine the lowest label of the data that can be involved in an operation, given two different labels. It returns a character string label that is the greatest lower bound of *label1* and *label2*. The greatest lower bound is the lowest level, the intersection of the compartments in the labels and

the groups in the labels. For example, the greatest lower bound of HIGHLY_ SENSITIVE:ALPHA and SENSITIVE is SENSITIVE.

Syntax:

```
FUNCTION GREATEST_LBOUND (
     label1                  IN NUMBER,
     label2                  IN NUMBER)
RETURN VARCHAR2;
```

Merging Labels with the MERGE_LABEL Function

The MERGE_LABEL function is a utility for merging two labels together. It accepts the character string form of two labels and the three-character specification of a merge format. Its syntax is as follows:

Syntax:

```
FUNCTION merge_label (label1 IN number,
                      label2 IN number,
                      merge_format IN VARCHAR2)
RETURN number;
```

The valid merge format is specified with a three-character string:

<highest level or lowest level><union or intersection of compartments><union or intersection of groups>

- The first character indicates whether to merge using the highest level or the lowest level of the two labels.
- The second character indicates whether to merge using the union or the intersection of the compartments in the two labels.
- The third character indicates whether to merge using the union or the intersection of the groups in the two labels.

The following table defines the MERGE_LABEL format constants.

Table 5–5 MERGE_LABEL Format Constants

Format Specification	Data Type	Constant	Meaning	Positions in Which Format Is Used
max_lvl_fmt	CONSTANT varchar2(1)	H	Maximum level	First (level)
min_lvl_fmt	CONSTANT varchar2(1)	L	Minimum level	First (Level)
union_fmt	CONSTANT varchar2(1)	U	Union of the two labels	Second (compartments) and Third (groups)
inter_fmt	CONSTANT varchar2(1)	I	Intersection of the two labels	Second (compartments) and Third (groups)
minus_fmt	CONSTANT varchar2(1)	M	Remove second label from first label	Second (compartments) and Third (groups)
null_fmt	CONSTANT varchar2(1)	N	If specified in compartments column, returns no compartments. If specified in groups column, returns no groups.	Second (compartments) and Third (groups)

For example, HUI specifies the highest level of the two labels, union of the compartments, intersection of the groups.

The MERGE_LABEL function is particularly useful to developers if the LEAST_UBOUND function does not provide the intended result. The LEAST_UBOUND function, when used with two labels containing groups, may result in a less sensitive data label than expected. The MERGE_LABEL function enables you to compute an intersection on the groups, instead of the union of groups that is provided by the LEAST_UBOUND function.

For example, if the label of one data record contains the group UNITED_STATES, and the label of another data record contains the group UNITED_KINGDOM, and the LEAST_UBOUND function is used to compute the least upper bound of these two labels, then the resulting label would be accessible to users authorized for either the UNITED_STATES or the UNITED_KINGDOM.

If, by contrast, the MERGE_LABEL function is used with a format clause of HUI, then the resulting label would contain the highest level, the union of the compartments, and no groups. This is because UNITED_STATES and UNITED_KINGDOM do not intersect.

Inserting Labeled Data

When you insert data into a table protected by a policy under Oracle Label Security, a numeric label value tag must be supplied, usually in the INSERT statement itself.

To do this, you must explicitly specify the tag for the desired label or explicitly convert the character string representation of the label into the correct tag. Note that this does not mean generating new label tags, but referencing the correct tag. When Oracle Label Security is using Oracle Internet Directory, the only permissible labels (and corresponding tags) are those pre-defined by the administrator and already in Oracle Internet Directory.

The only times an INSERT statement may omit a label value are:

1. if the LABEL_DEFAULT enforcement option was specified when the policy was applied, or
2. if no enforcement options were specified when the policy was applied and LABEL_DEFAULT was specified when the policy was created, or
3. if the statement applying the policy named a labeling function.

In cases 1 and 2, the user's session default row label is used as the inserted row's label. In case 3, the inserted row's label is created by that labeling function.

See Also:

- "Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY" on page 10-3, or to schemas on page 10-5
- "Creating a Policy with SA_SYSDBA.CREATE_POLICY" on page 7-11
- "Using a Labeling Function" on page 9-9
- All of Chapter 9, "Implementing Policy Enforcement Options and Labeling Functions" regarding reading and writing labeled data (and labels) and according to policy enforcement options

This section explains the different ways to specify a label in an INSERT statement:

Inserting Labels Using CHAR_TO_LABEL

To insert a row label, you can specify the label character string and then transform it into a label using the CHAR_TO_LABEL function. Using the definition for table emp on page 5-2, the following example shows how to insert data with explicit labels:

```
INSERT INTO emp (ename,empno,hr_label)
VALUES ('ESTANTON',10,char_to_label ('HR', 'SENSITIVE'));
```

Inserting Labels Using Numeric Label Tag Values

You can insert data using the numeric label tag value of a label, rather than using the CHAR_TO_LABEL function. For example, if the numeric label tag for SENSITIVE is 3000, it would look like this:

```
INSERT INTO emp (ename, empno, hr_label)
VALUES ('ESTANTON', 10, 3000);
```

Inserting Data Without Specifying a Label

If LABEL_DEFAULT is set, or if there is a labeling function applied to the table, then you do not need to specify a label in your INSERT statements. The label will be provided automatically. You can enter the following command:

```
INSERT INTO emp (ename, empno)
VALUES ('ESTANTON', 10);
```

The resulting row label is set according to the default value (or by a labeling function).

See Also:

- "Overview of Policy Enforcement Options" on page 9-1
- "The Label Management Enforcement Options" on page 9-5
- "Using a Labeling Function" on page 9-9
- "Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY" on page 10-3
- "Creating a Policy with SA_SYSDBA.CREATE_POLICY" on page 7-11

Inserting Data When the Policy Label Column Is Hidden

If the label column is hidden, then the existence of the column is transparent to the insertion of data. INSERT statements can be written that do not explicitly list the table columns and do not include a value for the label column. The session's row label is used to label the data, or a labeling function is used if one was specified when the policy was applied to the table or schema.

You can insert into a table without explicitly naming the columns, as long as you specify a value for each non-hidden column in the table. The following example shows

how to insert a row into the table described in "Example 2: Numeric Column Data Type with Hidden Column" on page 5-2:

```
INSERT INTO emp
VALUES ('196','ESTANTON',Technician,RSTOUT,50000,10);
```

Its label will be one of the following three possibilities:

- The label you specify
- The label established by the LABEL_DEFAULT option of the policy being applied
- The label created by a labeling function named by the policy being applied

---

Note: If the policy label column is *not* hidden, then you must explicitly include a label value (possibly null, indicated by a comma) in the INSERT statement.

Inserting Labels Using TO_DATA_LABEL

Note: When Oracle Label Security is installed to work with Oracle Internet Directory, dynamic label generation is not allowed, because labels are managed centrally in Oracle Internet Directory, using `olsadmintool` commands. Refer to Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory"

So, when Oracle Label Security is directory-enabled, this function, TO_DATA_LABEL, is not available and will generate an error message if used.

If you are generating new labels dynamically as you insert data, then you can use the TO_DATA_LABEL function to guarantee that this produces valid data labels. To do this, you must the have EXECUTE authority on the TO_DATA_LABEL function.

Whereas the CHAR_TO_LABEL function requires that the label already be an existing *data* label for the transaction to succeed, the TO_DATA_LABEL does not have this requirement. It will automatically create a valid data label.

For example:

```
INSERT INTO emp (ename, empno, hr_label)
VALUES ('ESTANTON', 10, to_data_label ('HR', 'SENSITIVE'));
```

Note: The TO_DATA_LABEL function must be explicitly granted to individuals, in order to be used. Its usage should be tightly controlled.

See Also: Chapter 10, "Applying Policies to Tables and Schemas" for more information about inserting, updating, and deleting labeled data

Changing Your Session and Row Labels with SA_SESSION

During a given session, a user can change his or her labels, within the authorizations set by the administrator.

This section contains these topics:

SA_SESSION Functions to Change Session and Row Labels

The following functions enable the user to change the session and row labels:

Table 5–6 Functions to Change Session Labels

Function	Purpose
SA_SESSION.SET_LABEL	Lets the user set a new level and new compartments and groups to which he or she has read access
SA_SESSION.SET_ROW_LABEL	Lets the user set the default row label that will be applied to new rows
SA_SESSION.RESTORE_DEFAULT_LABELS	Lets the user reset the current session label and row label to the stored default settings
SA_SESSION.SAVE_DEFAULT_LABELS	Lets the user store the current session label and row label as the default for future sessions

Changing the Session Label with SA_SESSION.SET_LABEL

Use the SET_LABEL procedure to set the label of the current database session.

Syntax:

```
PROCEDURE SET_LABEL (policy_name IN VARCHAR2,
                     label IN VARCHAR2);
```

Parameter	Specifies
policy_name	The name of an existing policy.
label	The value to set as the label

A user can set the session label to:

- Any level equal to or less than the maximum, and equal to or greater than the minimum level
- Include any compartments in the authorized compartment list
- Include any groups in the authorized group list. (Subgroups of authorized groups are implicitly included in the authorized list.)

Note that if you change the session label, this change may affect the value of the session's row label. The session's row label contains the subset of compartments and

groups for which the user has write access. This may or may not be equivalent to the session label. For example, if you use the SA_SESSION.SET_LABEL command to set your current session label to C:A,B:US and you have write access only on the A compartment, then your row label would be set to C:A.

See Also: "SA_USER_ADMIN.SET_DEFAULT_LABEL" on page 8-9

Changing the Row Label with SA_SESSION.SET_ROW_LABEL

Use the SET_ROW_LABEL procedure to set the default row label value for the current database session. The compartments and groups in the label must be a subset of the compartments and groups in the session label to which the user has write access. When the LABEL_DEFAULT option is set, this row label value is used on insert if the user does not explicitly specify the label.

Syntax:

```
PROCEDURE SET_ROW_LABEL (policy_name IN VARCHAR2,
                         row_label IN VARCHAR2);
```

Parameter	Specifies
policy_name	The name of an existing policy
label	The value to set as the default row label

If the SA_SESSION.SET_ROW_LABEL procedure is not used to set the default row label value, then this value is automatically derived from the session label. It contains the level of the session label and the subset of the compartments and groups in the session label for which the user has write authorization.

The row label is automatically reset if the session label changes. For example, if you change your session level from HIGHLY_SENSITIVE to SENSITIVE, then the level component of the row label automatically changes to SENSITIVE.

The user can set the row label independently, but only to include:

- A level that is less than or equal to the level of the session label, and greater than or equal to the user's minimum level
- A subset of the compartments and groups from the session label, for which the user is authorized to have write access

If the user tries to set the row label to an invalid value, then the operation is not permitted and the row label value is unchanged.

See Also: "SA_USER_ADMIN.SET_ROW_LABEL" on page 8-9

Restoring Label Defaults with SA_SESSION.RESTORE_DEFAULT_LABELS

The RESTORE_DEFAULT_LABELS procedure restores the session label and row label to those stored in the data dictionary. This command is useful to reset values after a SA_SESSION.SET_LABEL command has been processed.

Syntax:

```
PROCEDURE RESTORE_DEFAULT_LABELS (policy_name in VARCHAR2);
```

where *policy_name* provides the name of an existing policy.

Saving Label Defaults with SA_SESSION.SAVE_DEFAULT_LABELS

The SAVE_DEFAULT_LABELS procedure stores the current session label and row label as your initial session label and default row label. It permits you to change your defaults to reflect your current session label and row label. The saved labels will be used as the initial default settings for future sessions.

Syntax:

```
PROCEDURE SAVE_DEFAULT_LABELS (policy_name in VARCHAR2);
```

where *policy_name* provides the name of an existing policy.

When you log in to a database, your default session label and row label are used to initialize the session label and row label. When the administrator originally authorized your Oracle Label Security labels, he or she also defined your default level, default compartments, and default groups. If you change your session label and row label, and want to save these values as the default labels, you can use the SA_SESSION.SAVE_DEFAULT_LABELS procedure.

This procedure is useful if you have multiple sessions and want to be sure that all additional sessions have the same labels. You can save the current labels as the default, and all future sessions will have these as the initial labels.

Consider a situation in which you connect to the database through Oracle Forms and want to run a report. By saving the current session labels as the default before you call Oracle Reports, you ensure that Oracle Reports will initialize at the same labels as are being used by Oracle Forms.

Note: The SA_SESSION.SAVE_DEFAULT_LABELS procedure overrides the settings established by the administrator.

Viewing Session Attributes with SA_SESSION Functions

You can use SA_SESSION functions to view the policy attributes for a session.

- USER_SA_SESSION View to Return All Security Attributes
- Functions to Return Individual Security Attributes

USER_SA_SESSION View to Return All Security Attributes

You can display security attribute values by using the USER_SA_SESSION view. Access to this view is PUBLIC. It lets you see the security attributes for your current session. For example:

Table 5–7 Security Attribute Names and Types

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
SA_USER_NAME		VARCHAR2(4000)
PRIVS		VARCHAR2(4000)
MAX_READ_LABEL		VARCHAR2(4000)
MAX_WRITE_LABEL		VARCHAR2(4000)
MIN_LEVEL		VARCHAR2(4000)
LABEL		VARCHAR2(4000)

Table 5–7 (Cont.) Security Attribute Names and Types

Name	Null?	Type
COMP_WRITE		VARCHAR2(4000)
GROUP_WRITE		VARCHAR2(4000)
ROW_LABEL		VARCHAR2(4000)

Functions to Return Individual Security Attributes

The SA_SESSION functions take a *policy_name* as the only input parameter. They return VARCHAR2 character string values for use in SQL statements.

Table 5–8 SA_SESSION Functions to View Security Attributes

Function	Purpose
SA_SESSION.PRIVS	Returns the set of current session privileges, in a comma-delimited list
SA_SESSION.MIN_LEVEL	Returns the minimum level authorized for the session
SA_SESSION.MAX_LEVEL	Returns the maximum level authorized for the session
SA_SESSION.COMP_READ	Returns a comma-delimited list of compartments that the user is authorized to read
SA_SESSION.COMP_WRITE	Returns a comma-delimited list of compartments that the user is authorized to write. This is a subset of SA_SESSION.COMP_READ.
SA_SESSION.GROUP_READ	Returns a comma-delimited list of groups that the user is authorized to read
SA_SESSION.GROUP_WRITE	Returns a comma-delimited list of groups that the user is authorized to write. This is a subset of SA_SESSION.GROUP_READ.
SA_SESSION.LABEL	Returns the session label (the level, compartments, and groups) with which the user is currently working. The user can change this value with SA_SESSION.SET_LABEL. Refer to Changing the Session Label with SA_SESSION.SET_LABEL.
SA_SESSION.ROW_LABEL	Returns the session's default row label value. The user can change this value with SA_SESSION.SET_ROW_LABEL. Refer to Changing the Row Label with SA_SESSION.SET_ROW_LABEL.
SA_SESSION.SA_USER_NAME	Returns the username associated with the current Oracle Label Security session

For example, the following statement shows the current session label for the Human Resources policy:

```
SQL> select sa_session.label ('human_resources')
  2  from dual;

SA_SESSION.LABEL('HUMAN_RESOURCES')
-------------------------------------------
L3:M,E
```

See Also: "Using SA_UTL Functions to Set and Return Label Information" on page 11-5 for additional functions that return numeric label tags and BOOLEAN values

6

Oracle Label Security Using Oracle Internet Directory

Managing Oracle Label Security metadata in a centralized LDAP repository provides many benefits. Policies and user label authorizations can be easily provisioned and distributed throughout the enterprise. In addition, when employees are terminated, their label authorizations can be revoked in one place and the change automatically propagated throughout the enterprise. This chapter describes the integration between Oracle Label Security and Oracle Internet Directory, in the following sections:

Introducing Label Management on Oracle Internet Directory

Previous releases of Oracle Label Security have relied on the Oracle Database as the central repository for policy and user label authorizations. This architecture leveraged the scalability and high availability of the Oracle Database, but did not leverage the identity management infrastructure, which includes the Oracle Internet Directory. This directory is part of Oracle Identity Management Platform. Integrating your installation of Oracle Label Security with Oracle Internet Directory allows label authorizations to be part of your standard provisioning process.

These advantages accrue also to directory-stored information about policies, user labels, and privileges that Oracle Label Security assigns to users. These labels and privileges are specific to the installation's policies defining access control on tables and schemas.If a site is not using Oracle Internet Directory, then such information is stored locally in the database.

The following Oracle Label Security information is stored in the directory:

- Policy information, namely policy name, column name, policy enforcement options, and audit options
- User profiles identifying their labels and privileges
- Policy label components: levels, compartments, groups
- Policy data labels

Database-specific metadata is not stored in the directory. Examples include

- Lists of schemas or tables, with associated policy information, and
- Program units, with associated policy privileges

The following three notes identify important aspects of integrating your installation of Oracle Label Security with Oracle Internet Directory:

Note: Oracle will continue to support both the database and directory-based architectures for Oracle Label Security. However, a single database environment cannot host both architectures. Administrators must decide whether to use the centralized LDAP administration model or the database-centric model.

Note: Managing Oracle Label Security policies directly in the directory is done using a new command-line tool, the Oracle Label Security administration tool (`olsadmintool`), described in Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory".

Starting this release, you can also use the graphical user interface provided by Oracle Enterprise Manager Database Control or Grid Control to manage Oracle Label Security. Detailed documentation can be found in Oracle Enterprise Manager help.

For sites that use Oracle Internet Directory, databases retrieve Oracle Label Security policy information from the directory. Administrators use the `olsadmintool` policy administration tool or the Enterprise Manager graphical user interface to operate directly on the directory to insert, alter, or remove metadata as needed. Because enterprise users can log in to multiple databases using the credentials stored in Oracle Internet Directory, it is logical to store their Oracle Label Security policy authorizations and privileges there as well. An administrator can then modify these authorizations and privileges by updating such metadata in the directory.

For distributed databases, centralized policy management removes the need for replicating policies, because the appropriate policy information is available in the directory. Changes are effective without further effort, synchronized with policy information in the databases by means of the Directory Integration Platform.

> **See Also:** Synchronization using the Directory Integration Platform is described in the *Oracle Internet Directory Administrator's Guide*.

Figure 6–1, "Diagram of Oracle Label Security Metadata Storage in Oracle Internet Directory" illustrates the structure of metadata storage in Oracle Internet Directory.

Figure 6–2, "Oracle Label Security Policies Applied through Oracle Internet Directory" illustrates applying different policies stored in Oracle Internet Directory to the databases accessed by different enterprise users. Determining the policy to be applied is controlled by the directory entries corresponding to the user and the accessed database.

Figure 6–1 Diagram of Oracle Label Security Metadata Storage in Oracle Internet Directory

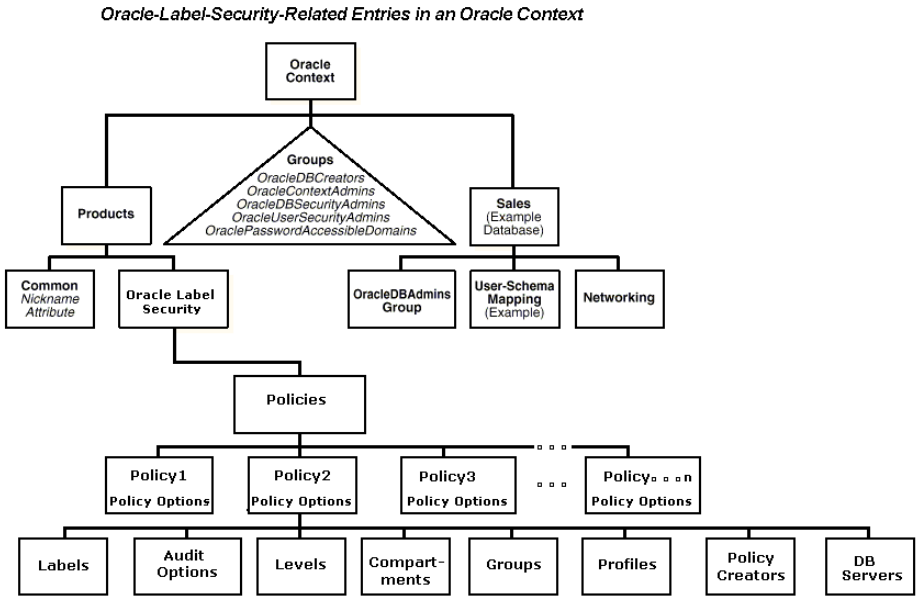
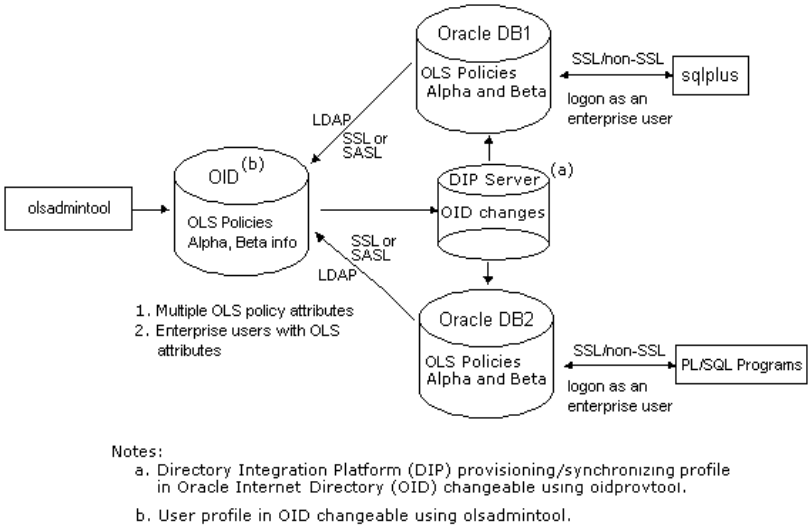


Figure 6–2 Oracle Label Security Policies Applied through Oracle Internet Directory



In this example, the directory has information about two Oracle Label Security policies, Alpha, applying to database DB1, and Beta, applying to database DB2 Although both policies are known to each database, only the appropriate one is applied in each case. In addition, enterprise users who are to access rows protected by Oracle Label Security are listed in profiles within the Oracle Label Security attributes in Oracle Internet Directory.

As Figure 6–2, "Oracle Label Security Policies Applied through Oracle Internet Directory" shows, the connections between different databases and the directory are established over either SSL or SASL. The database always binds to the directory as a known identity using password-based authentication. Links between databases and their clients (such as a sqlplus session, any PL/SQL programs , and so on) can use either SSL or non-SSL connections. The example of Figure 6–2, "Oracle Label Security Policies Applied through Oracle Internet Directory" assumes that users are logged on through password authentication. The choice of connection type depends on the enterprise user model.

The Oracle Label Security policy administration tool operates directly on metadata in Oracle Internet Directory. Changes in the directory are then propagated to the Oracle Directory Integration and Provisioning server, which is configured to send changes to the databases at specific time intervals.

The databases update the policy information in Oracle Internet Directory only when policies are being applied to tables or schemas. These updates ensure that policies that are in use will not be dropped from the directory.

See Also:

- *Oracle Database Enterprise User Security Administrator's Guide* for more information on enterprise domains, user models and authentication activities
- *Oracle Internet Directory Administrator's Guide* for detailed information on Oracle Internet Directory

Configuring Oracle Internet Directory-Enabled Label Security

You can configure a database for Oracle Internet Directory-enabled Label Security at any time after database creation or during custom database creation. Oracle Internet Directory-enabled label security relies on the Entrerprise User security feature.

See Also:

- "Enterprise User Security Configuration Tasks and Troubleshooting" in the *Oracle Database Enterprise User Security Administrator's Guide*, for prerequisites and steps to configure a database for directory usage, and
- "Database Configuration Assistant" in the *Oracle Database Enterprise User Security Administrator's Guide*, for information about DBCA, the Database Configuration Assistant.

Granting Permissions for Configuring Oracle Internet Directory enabled Oracle Label Security

Users who perform Oracle Internet Directory enabled Oracle Label Security using the Database Configuration Assistant (DBCA) need additional privileges. The following steps describe what permissions are needed, and how to grant them:

- Use Enterprise Manager to add the user to the OracleDBCreators group.

  See Also: "Managing Identity Management Realm Administrators" in the *Oracle Database Enterprise User Security Administrator's Guide* for more information on adding a user to an administrative group

- Add the user to the Provisioning Admins group. This is necessary because DBCA creates a DIP provisioning profile for Oracle Label Security. Use ldapmodify command with the following .ldif file to add a user to the Provisioning Admins group:

```
dn: cn=Provisioning Admins,cn=changelog subscriber, cn=oracle internet
directory
changetype: modify
add: uniquemember
uniquemember: DN of the user who is to be added
```

- Add the user to the policyCreators group using the `olsadmintool` command line tool . DBCA bootstraps the database with the Oracle Label Security policy information from Oracle Internet Directory, and only policyCreators can perform this bootstrap.
- If the database is already registered with the Oracle Internet Directory using DBCA, use Enterprise Manager to add the user to the OracleDBAdmins group of that database.

Note that the permissions specified earlier are also needed by the administrator who unregisters the database that has Oracle Internet Directory enabled Oracle Label Security configuration.

Registering a Database and Configuring Oracle Internet Directory enabled Oracle Label Security

To achieve this goal, do the following major tasks:

Task 1 Configure Your Oracle Home for Directory Usage.

> **See Also:** "Configuring Your Database to Use the Directory" in the *Oracle Database Enterprise User Security Administrator's Guide*

Task 2 Configure the Database for Oracle Internet Directory enabled Oracle Label Security

1. Register your database in the directory using DBCA (Database Configuration Assistant).

   > **See Also:** "Registering Your Database with the Directory" in the *Oracle Database Enterprise User Security Administrator's Guide*

2. After your database is registered in the directory, configure Label Security:
   a. Start DBCA, select **Configure database options in a database**, and click **Next**.
   b. Select a database and click **Next**.
   c. Regarding the option of unregistering the database or keeping it registered, select **Keep the database registered**.
   d. If the database is registered with Oracle Internet Directory, the **Database options** screen shows a customize button beside the Label Security check box. Select the **Label Security** option and click **Customize**.
   e. This customize dialog has two configuration options, for standalone Oracle Label Security or for Oracle Internet Directory-enabled Oracle Label Security. Click **OID-enabled Label security configuration** and enter the Oracle Internet Directory credentials of an appropriate administrator. Click **Ok**.

f. Continue with the remaining DBCA steps and click **Finish** when it appears.

Note: You can configure a standalone Oracle Label Security on a database that is registered with Oracle Internet Directory. Select the standalone option in step **e**.

When configuring for Oracle Internet Directory-enabled Oracle Label Security, DBCA also does the following things in addition to registering the database:

1. Creates a provisioning profile for propagating Label Security policy changes to to the database. This Directory Integration Platform (DIP) provisioning profile is enabled by default.
2. Installs the required packages on the database side for Oracle Internet Directory-enabled Oracle Label Security.
3. Bootstraps the database with all the existing Label Security policy information in the Oracle Internet Directory.

See Also: Bootstrapping Databases on page 6-11 for more information.

Alternate Method for Task 2, Configuring Database for Oracle Internet Directory enabled Oracle Label Security

Registering the database and configuring Oracle Label Security can be done in one invocation of DBCA.

1. Start DBCA.
2. Select **Configure database options in a database** and click **Next**.
3. Select a database and click **Next**.
4. Click **Register the database**.
5. Enter the Oracle Internet Directory credentials of an appropriate administrator, and the corresponding password for the database wallet that will be created.
6. The Database options screen shows a Customize button beside the Label Security check box. Select the **Label Security** option and click **Customize**.

   The Customize dialog box is displayed, showing two configuration options, for standalone Oracle Label Security or for Oracle Internet Directory-enabled Oracle Label Security.
7. Click **OID-enabled Label Security Configuration**.
8. Continue with the remaining DBCA steps and click **Finish**.

Task3: Set the DIP Password and Connect Data

1. Use the command line tool `oidprovtool` to set the password for the DIP user and update the interface connect information in the DIP provisioning profile for that database with the new password.

   See: Oracle Directory Integration and Provisioning (DIP) Provisioning Profiles on page 6-12 for more details

2. Upon creation, the DIP profile uses a schedule value of 3600 seconds by default, meaning that Oracle Label Security changes are propagated to the database every

hour. You can use `oidprovtool` to change this value if deployment considerations require that.

Once the the database is configured for Oracle Internet Directory-enabled Oracle Label Security, further considerations regarding enterprise user security may apply.

> **See Also:** *Oracle Database Enterprise User Security Administrator's Guide* for further concepts, tools, steps, and procedures

Unregistering a Database with Oracle Internet Directory enabled OLS

To perform this task, you use DBCA, which does the following things:

1. Deletes the DIP provisioning profile for the database created for Oracle Label Security.
2. Installs the required packages for standalone Oracle Label Security, so that at the end of unregistration, Oracle Internet Directory enabled Oracle Label Security becomes standalone Oracle Label Security.

> **Note:** Specific instructions for DB unregistration appear in the *Oracle Database Enterprise User Security Administrator's Guide*. No special steps are required when Oracle Internet Directory-enabled Oracle Label Security is configured.

> **Note:** If a database has standalone Oracle Label Security, it cannot be converted to Oracle Internet Directory-enabled Oracle Label Security. You need to drop Oracle Label Security from the database and then use DBCA again to configure Oracle Internet Directory-enabled Oracle Label Security.

Removing Directory-enabled Oracle Label Security from Database

To remove Oracle Internet Directory-enabled Oracle Label Security from a database, first unregister the database using DBCA, and then run the following script:

```
$ORACLE_HOME/rdbms/admin/catnools.sql
```

Oracle Label Security Profiles

A user profile is a set of user authorizations and privileges. Profiles are maintained as part of each Oracle Label Security policy stored in the Directory.

If a user is added to a profile, then the authorizations and privileges defined in that profile for that particular policy are aquired by the user, which include the following attributes:

- Five label authorizations:
  - maximum read label
  - maximum write label
  - minimum write label
  - default read label
  - default row label

- Privileges
- The list of enterprise users to whom these authorizations apply

See Also:

- Oracle Label Security Policy Attributes in Oracle Internet Directory on page 6-9
- "Getting Started with Enterprise User Security" in the *Oracle Database Enterprise User Security Administrator's Guide* for more infomation on creating and managing enterprise users
- Oracle Enterprise Manager help for information on creating and administering Oracle Label Security profiles and policies

An enterprise user can belong to only one profile, or none.

Integrated Capabilities When Label Security Uses the Directory

The integration of Oracle Label Security and Oracle Internet Directory enables the following capabilities:

- User/administrator actions
  - Storing multiple Oracle Label Security policies in Oracle Internet Directory
  - Managing Oracle Label Security policies and options in the directory, including
    * creating or dropping a policy
    * changing policy options
    * changing audit settings
  - Creating label components for any Oracle Label Security policies by
    * creating or removing levels, compartments, or groups
    * assigning numeric values to levels, compartments, or groups
    * changing long names of levels, compartments, or groups
    * creating children groups
  - Managing enterprise users configured as users of any Oracle Label Security policies, including
    * assigning or removing enterprise users to/from profiles within policies
    * assigning policy-specific privileges to enterprise users, or removing them
    * changing policy label authorizations assigned to enterprise users
  - Managing all user/administrator actions and capabilities by means of an integrated set of command line tools that monitor and manage Oracle Label Security policies in Oracle Internet Directory.
- Automatic results of Oracle Label Security
  - Limiting database policy usage to directory-defined policies only (no local policies defined or applied)
  - Synchronizing changes to policies in the directory with the databases using Oracle Label Security (to apply after enterprise users reconnect)

- After changes are propagated by the Directory Integration Platform, having immediate access to enterprise users' Oracle Label Security attributes when these users log on to any database using Oracle Label Security, assuming they are configured within any Oracle Label Security policies. These attributes include users' label authorizations and users' privileges.

Oracle Label Security Policy Attributes in Oracle Internet Directory

In Oracle Internet Directory, Oracle-related metadata is stored under cn=OracleContext. Within Label Security, each policy holds the information and parameters shown in Figure 6–1, "Diagram of Oracle Label Security Metadata Storage in Oracle Internet Directory":

When Oracle Label Security is used without Oracle Internet Directory, it supports automatic creation of data labels by means of a label function. However, when Oracle Label Security is used with Oracle Internet Directory, such functions can create labels only using data labels that are already defined in the directory.

Table 6–1 Contents of Each Policy

Type of Entry	Contents	Meaning/Sample Usage/References
Policy Name	The name assigned to this policy at its creation	Used in olsadmintool commands such as `olsadmintool createpolicy` (refer to Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory")
Column Name	The name of the column that will hold the label values relevant to this policy	Column is added to database. Refer to "The Policy Label Column and Label Tags" on page 5-1 "Inserting Labeled Data" on page 5-11 "The HIDE Policy Column Option" on page 9-5 Appendix E, "Reference". Used in `olsadmintool createpolicy`
Enforcement Options	Any combination of the following entries: `LABEL_DEFAULT,LABEL_UPDATE, CHECK_CONTROL,READ_CONTROL, WRITE_CONTROL,INSERT_CONTROL, DELETE_CONTROL,UPDATE_CONTROL, ALL_CONTROL, orNO_CONTROL`	Refer to the discussions in Chapter 9, "Implementing Policy Enforcement Options and Labeling Functions" and Appendix E, "Reference". Used in `olsadmintool createpolicy` and `olsadmintool alterpolicy`
Options	Enabled:`TRUE orFALSE`, Type:`ACCESS orSESSION`, Success:`SUCCESSFUL,UNSUCCESSFUL, or BOTH`.	Used in `olsadmintool audit`
Levels	Name and number for each level	Used in `olsadmintool create/alter/droplevel`
Compartments	Name and number for each compartment	Used in `olsadmintool create/alter/drop compartment`

Table 6–1 (Cont.) Contents of Each Policy

Type of Entry	Contents	Meaning/Sample Usage/References
Groups	Name, number, and parent for each group	Used in `olsadmintool create/alter/dropgroup`

Profiles	Maximum and default read labels, maximum and minimum write labels, default row label, list of users, and a set of privileges from this list:
	`READ,FULL,`
	`WRITEUP,WRITEDOWN, WRITEACROSS,`
	`PROFILE_ACCESS, or COMPACCESS`
		Policies can have one or more profiles, each of which can be assigned to many users. Profiles reduce the need to set up label authorizations for individual users.
		All users with the same set of labels and privileges are grouped in a single profile. Each profile represents a different set of labels, privileges, and users. Each profile in a policy is unique.

Data Labels	Full name and number for each valid data label	Refer to Restrictions on New Data Label Creation.
Administrators	Name of each administrator authorized to modify the parameters within this policy.	Policy administrators can modify parameters within a policy. They are not necessarily also policy creators, who have the right to create or remove policies or policy administrators. Refer to Security Roles and Permitted Actions.

Restrictions on New Data Label Creation

When Oracle Label Security is used with Oracle Internet Directory, data labels must be pre-defined in the directory.

They cannot be created dynamically by a label function, as is possible when label security is not integrated with the directory.

Two Types of Administrators

Administrators listed within a policy are those individuals authorized to do the following policy-specific administrative tasks:

- Modify existing policy options and audit settings.
- Enable or disable auditing for a policy.
- Create or remove levels, compartments, groups or children groups.
- Modify full/long names for levels, compartment, or groups.
- Define or modify enterprise user settings, in this policy, for:
  - Privileges
  - Maximum or minimum levels
  - Read, write, or row access for levels, compartments, or groups
  - Label profiles
- Remove enterprise users from a policy.

There is a higher level of administrators, called policy creators, who can create and remove Oracle Label Security policies and the policy administrators named within them.

Bootstrapping Databases

After a new database is registered with Oracle Internet Directory, the administrator can install Oracle Internet Directory enabled Oracle Label Security on that database. This installation process automatically creates a Directory Integration Platform (DIP) provisioning profile enabling policy information to be periodically refreshed in the future by downloading it to the database. Refer to Oracle Directory Integration and Provisioning (DIP) Provisioning Profiles.

When configuring the database for Oracle Internet Directory enabled Oracle Label Security, the DBCA tool puts all the policy information in Oracle Internet Directory into the database. At any point, the administrator can decide to bootstrap the database with the policy information again, using the bootstrap utility script at `$ORACLE_HOME/bin/olsoidsync`. The parameters it requires are as follows:

```
olsoidsync --dbconnectstring <"database connect string in host:port:sid format">
--dbuser <database user> --dbuserpassword <database user password> [-c] [-r]
[-b <admin context>] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>

For example,
olsoidsync --dbconnectstring yippee:1521:ora101 --dbuser lbacsys
--dbuserpassword lbacsys -c
-b "ou=Americas,o=Oracle,c=US" -h yippee -D cn=policycreator -w welcome1
```

The `olsoidsync` command pulls policy information from Oracle Internet Directory and populates the information in the database. Users must provide the database TNS name, the database user name, the database user's password, the administrative context (if any), the Oracle Internet Directory host name, the bind DN and bind password, and optionally the Oracle Internet Directory port number.

The optional `-c` switch causes the command to drop all the existing policies in the database and refresh it with policy information from Oracle Internet Directory.

The optional `-r` switch causes the command to drop all the policy metadata (without dropping the policies themselves) and refresh the policies with new metadata from Oracle Internet Directory.

Without these two switches, the command will only create new policies from Oracle Internet Directory, and will halt on any errors encountered during the refresh.

Synchronizing the Database and Oracle Internet Directory

Oracle Label Security metadata in the directory is synchronized with the databases using the Oracle Directory Provisioning Integration Service of the Directory Integration Platform.

Changes to the label security data in the directory are conveyed by the provisioning integration service in the form of provisioning events. A software agent receives these events and generates appropriate SQL or PL/SQL statements to update the database. After these statements are processed, Oracle Label Security data dictionaries are updated to match the changes already made in the directory.

Oracle Label Security subscribes itself to the Provisioning Integration Service automatically during installation. The provisioning service stores the information associated with each database in the form of a provisioning profile. The software agent

uses the identity of the user "DIP" to connect to the database, and the password "DIP", when synchronizing the changes in Oracle Internet Directory with the database.

If the password for the user DIP is changed, that information must be updated in the provisioning profile of the provisioning integration service.

The steps to change the database connection information in the DIP profile are as follows:

1. Disable the provisioning profile. This temporarily stops the propagation of label security changes in the directory to the database, but no data is lost. Once the profile is enabled, any label security changes that happened in the directory since the profile was disabled are synchronized with the database.
2. Update the database connection information in the profile.
3. Enable the profile.

Note: The database character set must be compatible with Oracle Internet Directory for Oracle Internet Directory-enabled Oracle Label Security to work correctly. Only then can there be successful synchronization of the Label Security metadata in Oracle Internet Directory with the Database.

Please refer to Chapters 2 and 3 of *Oracle Database Globalization Support Guide* for more information about Character sets and Globalization Support parameters.

See Also:

- Disabling, Changing, and Enabling a Provisioning Profile on page 6-14
- *Oracle Internet Directory Administrator's Guide* for more information about enabling and disabling of provisioning profiles

Oracle Directory Integration and Provisioning (DIP) Provisioning Profiles

The DIP server synchronizes policy changes in the directory with the connected databases, using a separate DIP provisioning profile created for each database. This profile is created automatically as part of the installation process for Oracle Internet Directory-enabled Oracle Label Security. The administrator can use the provisioning tool `oidprovtool` to modify the password for a database profile, using the script $ORACLE_HOME/bin/oidprovtool. Each such profile contains the following information:

Table 6–2 Elements in a DIP Provisioning Profile

Element	Name for This Element When Invoking oidprovtool
The LDAP host name	ldap_host
The LDAP port number	ldap_port
The user DN and password to bind to Oracle Internet Directory to retrieve policy information	ldap_user ldap_user_password
The database DN	application_dn

Table 6–2 (Cont.) Elements in a DIP Provisioning Profile

Element	Name for This Element When Invoking oidprovtool
The organization DN, that is, the administrative context in which changes are being made	`organization_dn`
The callback function to be invoked, that is, LBACSYS.OLS_DIP_NTFY	`interface_name`
The database connect information, which is the hostname of the database, the port number used to connect to the database, the database SID, the database user name and password	`interface_connect_info`
Event subscriptions, including all MODIFY, ADD and DELETE events under cn=LabelSecurity in Oracle Internet Directory	operation
The time interval between synchronizations	`schedule`

Here is an example of using `oidprovtool`, followed by an explanation of the parameters in this example:

```
oidprovtool operation=modify ldap_host=yippee ldap_port=389
ldap_user=cn=defense_admin ldap_user_password=welcome1
application_dn="cn=db1,cn=OracleContext,ou=Americas,o=Oracle,c=US"
organization_dn="ou=Americas,o=Oracle,c=US" interface_name=LBACSYS.OLS_DIP_NTFY
interface_type=PLSQL interface_connect_info=yippee:1521:db1:dip:newdip schedule=60
event_subscription= "ENTRY:cn=LabelSecurity,cn=Products,cn=OracleContext,
ou=Americas,o=Oracle,c=US:ADD(*)" event_subscription=
"ENTRY:cn=LabelSecurity,cn=Products, cn=OracleContext,ou=Americas,
o=Oracle,c=US:MODIFY(*)" event_subscription="ENTRY:cn=LabelSecurity,cn=Products,
cn=OracleContext, ou=Americas,o=Oracle,c=US:DELETE"
```

This sample `oidprovtool` command creates and enables a new DIP provisioning profile with the following attributes:

- Oracle Internet Directory in host yippee using port 389
- Oracle Internet Directory user bind DN: cn=defense_admin with password welcome1
- Database DN: cn=db1,cn=OracleContext,ou=Americas,o=Oracle,c=US
- Organization DN (administrative context): ou=Americas,o=Oracle,c=US
- Database on host yippee, listening on port 1521
- Oracle SID: db1
- Database user: dip with new password newdip
- Interval to synchronize directory with connected databases : 60 seconds
- All the ADD, MODIFY and DELETE events under cn=LabelSecurity to be sent to DIP

To start the DIP server, use $ORACLE_HOME/bin/oidctl. For example:

```
oidctl server=odisrv connect=db2 config=0 instance=0 start
```

This command will start the DIP server by connecting to db2 (the Oracle Internet Directory database) with config set 0 and instance number 0.

See also: *Oracle Internet Directory Administrator's Guide* for more information on DIP provisioning profiles

Disabling, Changing, and Enabling a Provisioning Profile

You can change the password for the `interface_connect_info`, which is the database password, by using the oidprovtool modify command, but first you must disable the profile. After changing the password, you then reenable the profile.

You can disable the Oracle Label Security provisioning profile using `oidprovtool`, specifying the disable operation and the first six original parameters shown here. (The other original parameters are not needed.) The command form is:

```
oidprovtool operation=disable ldap_host=< > ldap_port=< > ldap_user_dn=< >
   ldap_user_password=< > application_dn=< >  organization_dn=< >
```

Using parameters from the example given in the previous section, this command would look like this:

```
oidprovtool operation=disable ldap_host=yippee ldap_port=389
ldap_user=cn=defense_admin ldap_user_password=welcome1
application_dn="cn=db1,cn=OracleContext,ou=Americas,o=Oracle,c=US"
organization_dn="ou=Americas,o=Oracle,c=US"
```

To modify the password in the connection information, use the `oidprovtool` command, specifying the `modify` operation, the first six original parameters, and the new DIPuser password given in the connection info. The command form is:

```
oidprovtool operation=modify  ldap_host=< > ldap_port=< >
ldap_user_dn=< >  ldap_user_password=< > application_dn=< >
organization_dn=< >   interface_connect_info=< new_connect _info >
```

Using parameters from the example given in the previous section, this command would look like this:

```
oidprovtool operation=modify ldap_host=yippee ldap_port=389
ldap_user=cn=defense_admin ldap_user_password=welcome1
application_dn="cn=db1,cn=OracleContext,ou=Americas,o=Oracle,c=US"
organization_dn="ou=Americas,o=Oracle,c=US"
interface_connect_info=yippee:1521:db1:dip:NewestDIPpassword
```

Similarly, you can re-enable the Directory Integration Platform provisioning profile using oidprovtool as follows, again specifying the desired operation and the first six original parameters. (The other original parameters are not needed.) The command form is:

```
oidprovtool operation=enable  ldap_host=< > ldap_port=< >  ldap_user_dn=< >
ldap_user_password=< > application_dn=< >   organization_dn=< >
```

Again using parameters from the example given in the previous section, this command would look like this:

```
oidprovtool operation=enable ldap_host=yippee ldap_port=389
ldap_user=cn=defense_admin ldap_user_password=welcome1
application_dn="cn=db1,cn=OracleContext,ou=Americas,o=Oracle,c=US"
organization_dn="ou=Americas,o=Oracle,c=US"
```

Security Roles and Permitted Actions

To manage Oracle Label Security policies in Oracle Internet Directory, certain entities are given access control rights in the directory. The access control mechanisms are provided by Oracle Internet Directory.

Table 6–3 describes, in abstract terms, these entities and the tasks they are enabled to perform.

Table 6–4, " Access Levels Allowed by Users in OID", lists the specific access level operations permitted or disallowed for policy creators, policy administrators, and label security users.

Table 6–3 Tasks That Certain Entities Can Perform

Entity	Tasks This Entity Can Perform
Policy creators	Create new (or delete existing) policies, create new (or remove existing) policy administrators.
Policy administrators	For Policies: modify existing policy options and audit settings, enable or disable auditing for a policy. For Label components: create, modify, or remove levels, compartments and groups, such as by changing their full or long names or (for groups) by creating or deleting their children groups. For enterprise users: remove enterprise users from a policy, modify enterprise users' maximum or minimum levels, their read, write, and row access for compartments or groups, their privileges for a policy, and their label profiles.

Table 6–4 Access Levels Allowed by Users in OID

Entries	Policy Creators	Policy Administrators	Databases
cn=Policies	can modify	no access	no access
cn=Admins,cn=Policy1	can modify	no access	no access
uniqueMember: cn=Policy1	can browse	can browse	can modify
cn=PolicyCreators	no access[1]	no access	no access
cn=Levels,cn=Policy1	can browse and delete	can modify	no access
cn=Compartments,cn=Policy1	can browse and delete	can modify	no access
cn=Groups,cn=Policy1	can browse and delete	can modify	no access
cn=AuditOptions,cn=Policy1	can browse and delete	can modify	no access
cn=Profiles,cn=Policy1	can browse and delete	can modify	no access
cn=Labels,cn=Policy1	can browse and delete	can modify	no access
cn=DBServers	no access[2]	no access	no access

[1] The group cn=OracleContextAdmins is the owner of the group cn=PolicyCreators, so members in cn=OracleContextAdmins can modify cn=PolicyCreators.

[2] The group cn=OracleDBCreators is the owner of the group cn=DBServers, so members in cn=OracleDBCreators can modify cn=DBServers.

Restriction on Policy Creators for Directory-enabled Oracle Label Security

A member of the Policy Creators group can only create, browse, and delete Oracle Label Security policies.

This user cannot perform policy administrative tasks, such as creating label components and adding users, even if explicitly added to the Policy Admins group of that policy. In short, a policy creator cannot be the administrator of any policy.

Superseded PL/SQL Statements

When Oracle Internet Directory is enabled with Oracle Label Security, the procedures listed in the following table are superseded. Only LBACSYS is allowed to run these procedures.

For some of the procedures listed in the table, the functionality they provided is replaced by the `olsadmintool` command named in the second column (and explained in Appendix E, "Reference").

Table 6–5 Procedures Superseded by olsadmintool When Using Oracle Internet Directory

Disabled Procedure	Replaced by olsadmintool Command
SA_SYSDBA.CREATE_POLICY	olsadmintool createpolicy
SA_SYSDBA.ALTER_POLICY	olsadmintool alterpolicy
SA_SYSDBA.DROP_POLICY	olsadmintool droppolicy
SA_COMPONENTS.CREATE_LEVEL	olsadmintool createlevel
SA_COMPONENTS.ALTER_LEVEL	olsadmintool alterlevel
SA_COMPONENTS.DROP_LEVEL	olsadmintool droplevel
SA_COMPONENTS.CREATE_COMPARTMENT	olsadmintool createcompartment
SA_COMPONENTS.ALTER_COMPARTMENT	olsadmintool altercompartment
SA_COMPONENTS.DROP_COMPARTMENT	olsadmintool dropcompartment
SA_COMPONENTS.CREATE_GROUP	olsadmintool creategroup
SA_COMPONENTS.ALTER_GROUP	olsadmintool altergroup
SA_COMPONENTS.ALTER_GROUP_PARENT	olsadmintool altergroup
SA_COMPONENTS.DROP_GROUP	olsadmintool dropgroup
SA_USER_ADMIN.SET_LEVELS	None
SA_USER_ADMIN.SET_COMPARTMENTS	None
SA_USER_ADMIN.SET_GROUPS	None
SA_USER_ADMIN.ADD_COMPARTMENTS	None
SA_USER_ADMIN.ALTER_COMPARTMENTS	None
SA_USER_ADMIN.DROP_COMPARTMENTS	None
SA_USER_ADMIN.DROP_ALL_COMPARTMENTS	None
SA_USER_ADMIN.ADD_GROUPS	None
SA_USER_ADMIN.ALTER_GROUPS	None
SA_USER_ADMIN.DROP_GROUPS	None
SA_USER_ADMIN.DROP_ALL_GROUPS	None
SA_USER_ADMIN.SET_USER_LABELS	olsadmintool createprofile; olsadmintool adduser; olsadmintool dropprofile; olsadmintool dropuser;
SA_USER_ADMIN.SET_DEFAULT_LABEL	None
SA_USER_ADMIN.SET_ROW_LABEL	None

Table 6–5 (Cont.) Procedures Superseded by olsadmintool When Using Oracle Internet Directory

Disabled Procedure	Replaced by olsadmintool Command
SA_USER_ADMIN.DROP_USER_ACCESS	`olsadmintool dropuser`
SA_USER_ADMIN.SET_USER_PRIVS	olsadmintool createprofile; olsadmintool adduser;olsadmintool dropprofile; olsadmintool dropuser;
SA_AUDIT_ADMIN.AUDIT	olsadmintool audit
SA_AUDIT_ADMIN.NOAUDIT	olsadmintool noaudit
SA_AUDIT_ADMIN.AUDIT_LABEL	None
SA_AUDIT_ADMIN.NOAUDIT_LABEL	None

Procedures for Policy Administrators Only

The following procedures are allowed to be run only by policy administrators (enterprise users defined in Oracle Internet Directory):

- SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY
- SA_POLICY_ADMIN.APPLY_TABLE_POLICY
- SA_POLICY_ADMIN.DISABLE_SCHEMA_POLICY
- SA_POLICY_ADMIN.DISABLE_TABLE_POLICY
- SA_POLICY_ADMIN.ENABLE_SCHEMA_POLICY
- SA_POLICY_ADMIN.ENABLE_TABLE_POLICY
- SA_POLICY_ADMIN.GRANT_PROG_PRIVS
- SA_POLICY_ADMIN.POLICY_SUBSCRIBE
- SA_POLICY_ADMIN.POLICY_UNSUBSCRIBE
- SA_POLICY_ADMIN.REMOVE_SCHEMA_POLICY
- SA_POLICY_ADMIN.REMOVE_TABLE_POLICY
- SA_POLICY_ADMIN.SET_PROG_PRIVS
- SA_POLICY_ADMIN.REVOKE_PROG_PRIVS

Part III

Administering an Oracle Label Security Application

This part contains the following chapters:

7

Creating an Oracle Label Security Policy

This chapter explains how to create an Oracle Label Security policy. It contains these sections:

Oracle Label Security Administrative Task Overview

To create and implement an Oracle Label Security policy, you perform the following tasks, which are described in the next few chapters:

Step 1: Create the Policy

Create a policy by defining:

- The policy name
- The column name for policy labels
- The default options for the policy

You can use Oracle Enterprise Manager Database Control or Grid Control interface to create a policy.

To create a policy using Oracle Enterprise Manager:

1. Log in to Oracle Enterprise Manager Database Control using the `LBACSYS` account.
2. Click the **Server** tab.
3. Click **Oracle Label Security** under Security. The Label Security Policies page appears.
4. Click **Create** to start creating a new label security policy.

   The Create Label Security Policy page appears.
5. Define the policy's name, label column, and the default policy enforcement options.
   - **Name**: Enter a name for the policy, for example, `ACCESS_LOCATIONS`.
   - **Label Column**: Enter a name for the label column, for example, `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)`. You can also use an existing table column of the `NUMBER(10)` data type as the label column.
   - **Hide Label Column**: Select to hide the column. When you first create the policy, you may want to disable **Hide Label Column** during the development phase of the policy. When the policy is satisfactory and ready for use by users, hide the column so that it is transparent to applications.
   - **Enabled**: Toggle to enable or disable the policy.
   - **Enforcement Options**: The default policy enforcement options are used when the policy is applied. Ensure that these meet the needs of the application to which you are applying the policy.

     Select from the following options:

     – **Apply No Policy Enforcements (NO_CONTROL)**

     – **Apply Policy Enforcements**

     For all queries (READ_CONTROL)

     For Insert operations (INSERT_CONTROL)

     For Update Operations (UPDATE_CONTROL)

     Use session's default label for label column update (LABEL_DEFAULT)

     Operations that update the label column (LABEL_UPDATE)

     Update and Insert operations so that they are read accessible (CHECK_CONTROL)
6. Click **OK**.

   The new policy appears in the Oracle Label Security Policies page.

Alternatively, you can use the SA_SYSDBA.CREATE_POLICY command-line procedure to create a policy.

See Also: "Creating a Policy with SA_SYSDBA.CREATE_POLICY" on page 7-11

Step 2: Define the Components of the Labels

Define the levels, compartments, and groups that form the components of the new policy's labels.

To create the label components using Oracle Enterprise Manager:

1. In the Oracle Label Security Policies page, select the policy you just created. Click **Edit**.
2. In the Edit Label Security Policy page, select the **Label Components** tab.
3. Click **Add 5 Rows** under Levels to add levels for the policy. Enter a Long Name, Short Name, and Numeric Tag for each level that you create. The numeric tag corresponds to the sensitivity of the level. To create more levels, you can click **Add 5 Rows** again. Use the same steps to create compartments and rows. For compartments and groups, the numeric tags do not correspond to sensitivity.

   At a minimum, you must create one level, such as SECRET. Creating compartments and groups is optional.

   The level numbers indicate the level of sensitivity for their corresponding labels. A greater number implies greater sensitivity. Select a numeric range that can be expanded later on, in case your security policy needs more levels. For example, if you have created levels PUBLIC (7000) and SENSITIVE (8000), and you now want to create an intermediate level called CONFIDENTIAL, then you can assign the numeric value 7500 to this level.

   Compartments identify categories associated with data, providing a finer level of granularity within a level. For example, a single table might have data corresponding to different departments that you might like to separate using compartments. 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.
4. Click **Apply**.

Alternatively, you can use the SA_COMPONENTS package on the command line to create the label components.

> **See Also:** "Using the SA_COMPONENTS Package to Define Label Components" on page 7-14

Step 3: Identify the Set of Valid Data Labels

Specify the set of valid labels to support the policy. From all the possible combinations of levels, compartments, and groups, you must define labels that can be assigned to data.

To create data labels for a policy:

1. In the Label Security Policies page, select the policy that needs to have labels linked to levels.
2. In the **Actions** box, select Data Labels. Click **Go**.

   The Data Labels page appears.
3. Click **Add**.

   The Create Data Label page appears.

4. Enter the following information:
   - **Numeric Tag**: Enter a number that uniquely identifies the label. This number should be unique across all policies.
   - **Level**: Select a level from the list.
5. You can optionally select Compartments to add to the label. To add compartments, click **Add** under Compartments. Select the compartments to be added to the label. Click **Select** to add the compartments.
6. You can optionally select Groups to add to the label. To add groups, click **Add** under Groups. Select the groups to be added to the label. Click **Select** to add the groups.
7. Click **OK** in the Create Data Label page.

   The data label appears in the Data Labels page.
8. Repeat steps 3 to 7 to create more data labels.

Alternatively, you can use the SA_LABEL_ADMIN package to create the label components.

Applications that need to create data labels dynamically at runtime can use the TO_DATA_LABEL function.

Note: When Oracle Label Security is installed to work with Oracle Internet Directory, dynamic label generation is not allowed, because labels are managed centrally in Oracle Internet Directory, using `olsadmintool` commands. Refer to Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory".)

So, when Oracle Label Security is directory-enabled, this function, TO_DATA_LABEL, is not available and will generate an error message if used.

See Also: "Using the SA_LABEL_ADMIN Package to Specify Valid Labels" on page 7-19

"Inserting Labels Using TO_DATA_LABEL" on page 5-13

Step 4: Apply the Policy to Tables and Schemas

Protect individual database tables and schemas by applying the policy to them. In the process, you can customize the level of enforcement of the policy for each table and schema, to reflect your application security requirements.

To apply the policy to a database table:

1. In the Label Security Policies page, select the policy that needs to be applied to a table.
2. Select Apply from the **Actions** box. Click **Go**.

   The Apply page appears.
3. Select the **Tables** tab to apply the policy to a table.

Note: Select the **Schemas** tab if you are applying the policy to a schema.The process is same as applying the policy to a table.

4. Click **Create**.

   The Add Table page appears.

5. Next to the **Table** box, click the flashlight icon.

6. In the Search and Select window, enter the following information under Search:

   - **Schema**: Enter the name of the schema in which the table appears. Leaving this field empty displays tables in all schemas.
   - **Name**: Optionally, enter the name of the table. Leaving this box empty displays all the tables within the schema.

   To narrow the search by using wildcards, use the percent (%) sign. For example, enter `O%` to search for all tables beginning with the letter O.

7. Select the table and click **Select**.

   The Add Table page appears.

8. Enter the following information:

   - **Policy Enforcement Options**: Select enforcement options as needed. These options will apply to the table on top of the enforcement options that you selected when you created the policy in Step 1: Create the Policy.

     To make no change from those enforcement options, that is, to use the same enforcement options created earlier, select **Use Default Policy Enforcement**. To add more enforcement options, select from the other options listed.

   - **Labeling Function**: Optionally, specify a labeling function to automatically compute the label to be associated with a new or updated row. That function is always invoked thereafter to provide the data labels written under that policy, because active labeling functions take precedence over any alternative means of supplying a label.
   - **Predicate**: Optionally, specify an additional predicate to combine (using `AND` or `OR`) with the label-based predicate for `READ_CONTROL`.

9. Click **OK**.

Alternatively, you can use the SA_POLICY_ADMIN package to apply policies to tables and schemas.

See Also: Chapter 10, "Applying Policies to Tables and Schemas"

Step 5: Authorize Users

For individual users, define the authorizations that each person will use for session access. If users do not have appropriate authorizations, they cannot access protected data.

You can optionally assign special privileges that particular users need to do their job. Note that Oracle Label Security privileges may only be necessary to perform special job functions.

To authorize users for the OLS policy:

1. In the Label Security Policies page, select the policy that needs authorization.

2. In the **Actions** box, select Authorization. Click **Go**.

   The Authorization page appears. Make sure that the **Users** tab is selected.

3. Click **Add Users**.

   The Add Users page appears.

4. Add users as follows:

   - Under Database Users, click **Add**. In the Search and Select window, select users that you want and then click **Select**.
   - Under Non Database Users, click **Add 5 Rows**, and then add the user names of the non-database users that you want to add. Most application users are considered non-database users. A non-database 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 non-database 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 real database user.

5. In the Create User page, select the user that you want to authorize. Click **Next**. If you have multiple users that need the same authorizations, then select all users who need the same authorizations. Click **Next**.

   The Privileges step appears.

6. Next, you can assign privileges to the user you selected in the preceding step. Privileges allow a database user to bypass certain controls enforced by the policy. Select the privileges you want to grant. Click **Next**.

   If you do not wish to assign any privilege to the user, click **Next** without selecting any privileges.

   The Labels, Compartments, and Groups step appears.

7. Next, you need to create the user label for the user. Under Levels, use the flashlight icon to select data to enter for the following fields:

   - **Maximum Level**: Enter the highest level for read and write access for this user.
   - **Minimum Level**: Enter the lowest level for write access.
   - **Default Level**: Enter the default level when the user logs in.

     This value is equal to or greater than the minimum level and equal to or less than the maximum level.
   - **Row Level**: Enter the level given to the row when user writes to the table.

8. Click **Add** under Compartments, to add compartments to the user label. Select the compartments to add. Click **Select**.

9. For each compartment that you add, you can select the following properties:

   - **Write**: Allows the user to write to data that has the compartment as part of it's label
   - **Default**: Adds the compartment to the user's default session label
   - **Row**: Adds the compartment to the data label when the user writes to the table

10. Click **Add** under Groups, to add groups to the user label. Select the groups and click **Select**.
11. For each group that you add, you can select the following properties:
    - **Write**: Allows the user to write to data that has the group as part of it's label
    - **Default**: Adds the group to the user's default session label
    - **Row**: Adds the group to the data label when the user writes to the table
12. Click **Next**.

    The Audit step appears.
13. Next, you can choose to set the policy audit options for the selected user. You can set audit options for the following operations:
    - Policy Applied:

      Audit On Success By audits successful application of the policy to a table or schema. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed application of the policy to a table or schema. Select ACCESS to audit by access or SESSION to audit by session.
    - Policy Removed:

      Audit On Success By audits successful removal of the policy from a table or schema. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed removal of the policy from a table or schema. Select ACCESS to audit by access or SESSION to audit by session.
    - Labels And Privileges Set:

      Audit On Success By audits successful setting of user authorizations and privileges. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed setting of user authorizations and privileges. Select ACCESS to audit by access or SESSION to audit by session.
    - All Policy Specific Privileges:

      Audit On Success By audits successful use of policy privileges. Select ACCESS to audit by access or SESSION to audit by session.

      Audit On Failure By audits failed use of policy privileges. Select ACCESS to audit by access or SESSION to audit by session.
14. Click **Next**.
15. You can review the policy authorization settings. Click **Finish** to create the policy authorization. Alternatively, you can click **Back** to modify the authorization settings.

Alternatively, you can use the SA_POLICY_ADMIN package to authorize users.

See Also: Chapter 8, "Administering User Labels and Privileges"

Step 6: Create and Authorize Trusted Program Units (Optional)

Trusted program units are functions, procedures, or packages that are granted Oracle Label Security privileges. You create a trusted stored program unit in the same way that you create a standard procedure, function, or package, that is by using the `CREATE PROCEDURE`, `CREATE FUNCTION`, or `CREATE PACKAGE` and `CREATE`

PACKAGE BODY statements. The program unit becomes trusted when you grant Oracle Label Security privileges to it.

To set privileges for a program unit:

1. In the Label Security Policies page, select the policy that needs authorization.
2. In the **Actions** box, select Authorization. Click **Go**.

   The Authorization page appears.
3. Click the **Trusted Program Units** tab.
4. Click **Add** to add Oracle Label Security privileges for a procedure, function, or package.

   The **Create Program Unit** page appears.
5. Enter the name of the procedure, function, or package, for which the privileges need to be granted, in the **Program Unit** field. You can also use the **Search** icon to search for the procedure, function, or package.
6. Select one or more policy-specific privileges that need to be granted to the program unit. Click **OK**.

   The trusted program unit is added to the Authorizations page.

Alternatively, you can use the SA_USER_ADMIN package to authorize trusted program units.

> **See Also:** Chapter 11, "Administering and Using Trusted Stored Program Units"

Step 7: Configure Auditing (Optional)

Configure monitoring of the administrative tasks and use of privileges, if desired.

To configure audit settings for an existing Oracle Label Security policy:

1. In the Label Security Policies page, select the policy that you need to configure.
2. Click **Edit**.

   The Edit Label Security Policy Settings page appears.
3. Click the **Advanced** tab. You can edit the audit settings under the Audit section.
4. Select **Include Label In Audit trail** under Audit Labels, if you wish to include user session labels in the audit table.
5. Select the **Operation**, to audit, under Audit Settings. You can choose from the following operations:
   - Policy Applied: Audits application of the policy to a table or schema.
   - Policy Removed: Audits removal of the policy from a table or schema.
   - Labels And Privileges Set: Audits setting of user authorizations and privileges.
   - All Policy Specific Privileges: Audits use of policy privileges.
6. Click **Add** under Policy Applied to add users that will be audited for the **Operation** you selected in the preceding step.

   The Search and Select window appears.
7. Select the users that you need to add. Click **Select**.

8. Select values for **Audit on Success By** and **Audit on Failure By**, for each user that you added.

   For each user that you added, you can choose to audit successful and failed instances of the chosen operation. You can also choose to audit by access or session.

9. Repeat steps 5 to 8 for each operation that you choose to audit.

See Also: Chapter 12, "Auditing Under Oracle Label Security"

Organizing the Duties of Oracle Label Security Administrators

You can manage the administration of an Oracle Label Security policy in various ways. The policy_DBA role is created when you create a new policy, and every individual who needs to perform administrative functions must be granted this role. However, you can grant EXECUTE privileges on the administrative packages to different users, so that each administrator can be restricted to a subset of the administrative functions.

For example, you could grant EXECUTE privilege on SA_COMPONENTS and SA_LABEL_ADMIN to one user or role to manage the label definitions, and grant EXECUTE on SA_USER_ADMIN to a different user or role to manage user labels and privileges. Alternatively, you could grant EXECUTE on all of the administrative packages to the policy_DBA role, so that anyone with the policy_DBA role could perform all of the administrative tasks.

Choosing an Oracle Label Security Administrative Interface

You can perform Oracle Label Security development and administrative tasks using either of two interfaces:

- Oracle Label Security Packages
- Oracle Enterprise Manager

Oracle Label Security Packages

Oracle Label Security packages provide a direct, command-line interface for ease of administration. These include:

Table 7–1 Oracle Label Security Administrative Packages

Package	Purpose
SA_SYSDBA	To create, alter, and drop Oracle Label Security policies
SA_COMPONENTS	To define the levels, compartments, and groups for the policy
SA_LABEL_ADMIN	To perform standard label policy administrative functions, such as creating labels
SA_POLICY_ADMIN	To apply policies to schemas and tables
SA_USER_ADMIN	To manage user authorizations for levels, compartments, and groups, as well as program unit privileges. Also to administer user privileges.
SA_AUDIT_ADMIN	To set options to audit administrative tasks and use of privileges

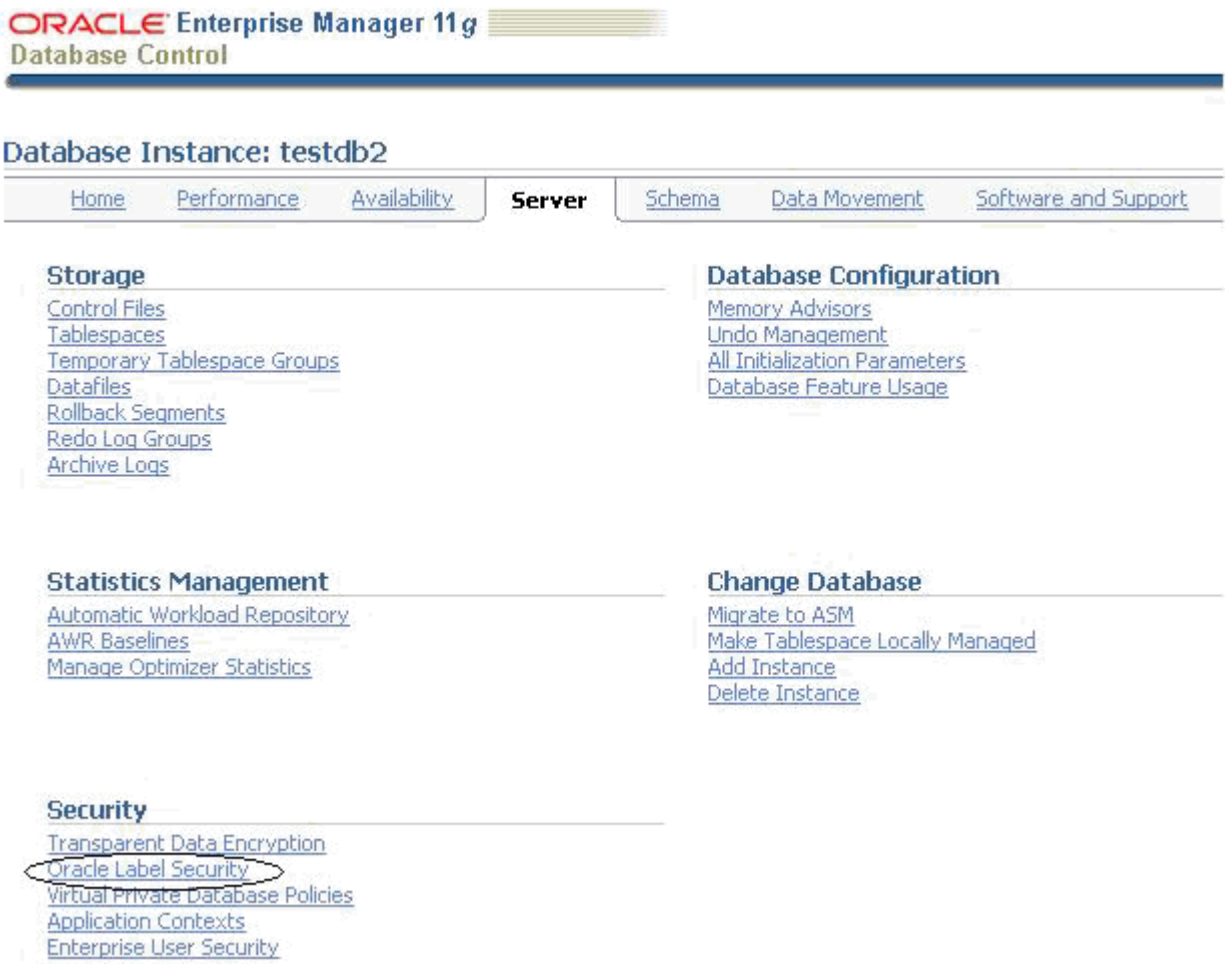
Oracle Label Security Demonstration File

For a demonstration showing how to create and develop an Oracle Label Security policy using the supplied packages, refer to the `olsdemo.sql` file in your `ORACLE_HOME/rdbms/demo` directory.

Oracle Enterprise Manager

You can use the Web interface provided by Oracle Enterprise Manager Database Control or Grid Control to administer Oracle Label Security. Figure 7–1 is a representative screenshot that illustrates the Oracle Enterprise Manager interface.

Figure 7–1 Using Enterprise Manager to Configure Oracle Label Security Policies



See Also:

- Chapter 4, "Getting Started with Oracle Label Security" for details on using Enterprise Manager for adminstering Oracle Label Security
- Enterprise Manager Online Help for details on using the Enterprise Manager Database Control or Grid Control interface

Using the SA_SYSDBA Package to Manage Security Policies

This section explains how to manage a policy using the SA_SYSDBA package. It includes the following topics:

Who Can Use the SA_SYSDBA Package

To use the SA_SYSDBA package to create, alter, and drop policies, a user must have:

- The LBAC_DBA role
- EXECUTE privilege on the SA_SYSDBA package

Who Can Administer a Policy

When you create a policy, a role named *policy*_DBA is automatically created. You can use this role to control the users who are authorized to run the policy's administrative procedures.

For example, after you have created a human resources policy named HR, an HR_DBA role is automatically created. To use any administrative packages, a user would need to have the HR_DBA role. If Joan is the administrator of the HR policy, and David is the administrator of the FIN policy, then Joan has the HR_DBA role and David has the FIN_DBA role. Each person can administer that policy for which he or she has the *policy*_DBA role.

The user who creates the *policy* is automatically granted the *policy*_DBA role with the ADMIN option, and the user can grant the role to others.

Valid Characters for Policy Specifications

Valid characters for all policy specifications include alphanumeric characters and underscores, as well as any valid character from your database character set.

Creating a Policy with SA_SYSDBA.CREATE_POLICY

Use the CREATE_POLICY procedure to create a new Oracle Label Security policy, define a policy-specific column name, and specify a set of default policy options.

Syntax:

```
PROCEDURE CREATE_POLICY (
   policy_name       IN VARCHAR2,
   column_name       IN VARCHAR2 DEFAULT NULL,
   default_options   IN VARCHAR2 DEFAULT NULL);
```

Table 7–2 Parameters for SA_SYSDBA.CREATE_POLICY

Parameter Name	Parameter Description
policy_name	Specifies the policy name, which must be unique within the database. It can have a maximum of 30 characters, but only the first 26 characters in the *policy_name* are significant. Two policies may not have the same first 26 characters in the *policy_name*.
column_name	Specifies the name of the column to be added to tables protected by the policy. If NULL, the default name "SA_LABEL" is used. Two Oracle Label Security policies cannot share the same column name.
default_options	Specifies the default options to be used when the policy is applied and no table- or schema-specific options are specified. Includes enforcement options and the option to hide the label column.

See Also:

- Regarding policy enforcement options for tables: "Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY" on page 10-3
- Regarding HIDE, "Choosing Policy Options" on page 9-1 and "The HIDE Policy Column Option" on page 9-5.
- "SYSDBA.CREATE_POLICY with Inverse Groups" on page 15-12.

Modifying Policy Options with SA_SYSDBA.ALTER_POLICY

Use the ALTER_POLICY procedure to set and modify policy default options.

Syntax:

```
PROCEDURE ALTER_POLICY (
   policy_name       IN  VARCHAR2,
   default_options   IN  VARCHAR2 DEFAULT NULL);
```

Table 7–3 Parameters for SA_SYSDBA.ALTER_POLICY

Parameter Name	Parameter Description
policy_name	Specifies the policy name
default_options	Specifies the default options to be used when the policy is applied and no table- or schema-specific options are specified. Includes enforcement options and the option to hide the label column.

Disabling a Policy with SA_SYSDBA.DISABLE_POLICY

Use the DISABLE_POLICY procedure to turn off enforcement of a policy, without removing it from the database. The policy is not enforced for all subsequent access to the database.

To disable a policy means that no access control is enforced on the tables and schemas protected by the policy. The administrator can continue to perform administrative operations while the policy is disabled.

Syntax:

```
PROCEDURE DISABLE_POLICY (policy_name IN VARCHAR2);
```

Table 7–4 Parameters for SA_SYSDBA.DISABLE_POLICY

Parameter Name	Parameter Description
policy_name	Specifies the policy to be disabled

> **Note:** This feature is extremely powerful, and should be used with caution. When a policy is disabled, anyone who connects to the database can access all the data normally protected by the policy. So, your site should establish guidelines for use of this feature.

Normally, a policy should not be disabled in order to manage data. At times, however, an administrator may need to disable a policy to perform application debugging tasks. In this case, the database should be run in single-user mode. In a development environment, for example, you may need to observe data processing operations without the policy turned on. When you reenable the policy, all of the selected enforcement options become effective again.

Enabling a Policy with SA_SYSDBA.ENABLE_POLICY

Use the ENABLE_POLICY procedure to enforce access control on the tables and schemas protected by the policy. A policy is automatically enabled when it is created. After creation or enabling, the policy is enforced for all subsequent access to tables protected by the policy.

Syntax:

```
PROCEDURE ENABLE_POLICY (policy_name IN VARCHAR2);
```

Table 7–5 Parameters for SA_SYSDBA.ENABLE_POLICY

Parameter Name	Parameter Description
policy_name	Specifies the policy to be enabled

Removing a Policy with SA_SYSDBA.DROP_POLICY

Use the DROP_POLICY procedure to remove the policy and all of its associated user labels and data labels from the database. It purges the policy from the system entirely. You can optionally drop the label column from all tables controlled by the policy.

Syntax:

```
PROCEDURE DROP_POLICY (policy_name IN VARCHAR2,
   drop_column  BOOLEAN DEFAULT FALSE);
```

Table 7–6 Parameters for SA_SYSDBA.DROP_POLICY

Parameter Name	Parameter Description
policy_name	Specifies the policy to be dropped
drop_column	Indicates that the policy column should be dropped from protected tables (TRUE)

Using the SA_COMPONENTS Package to Define Label Components

This package manages the component definitions of an Oracle Label Security label. Each policy defines the components differently. This section contains these topics:

- Creating a Level with SA_COMPONENTS.CREATE_LEVEL
- Modifying a Level with SA_COMPONENTS.ALTER_LEVEL
- Removing a Level with SA_COMPONENTS.DROP_LEVEL
- Creating a Compartment with SA_COMPONENTS.CREATE_COMPARTMENT
- Modifying a Compartment with SA_COMPONENTS.ALTER_COMPARTMENT
- Removing a Compartment with SA_COMPONENTS.DROP_COMPARTMENT
- Creating a Group with SA_COMPONENTS.CREATE_GROUP
- Modifying a Group with SA_COMPONENTS.ALTER_GROUP
- Modifying a Group Parent with SA_COMPONENTS.ALTER_GROUP_PARENT
- Removing a Group with SA_COMPONENTS.DROP_GROUP

See Also:

Chapter 2, "Understanding Data Labels and User Labels"

"Using Oracle Label Security Views" on page 8-12 for information about displaying the label definitions you have set

Using Overloaded Procedures

Oracle Label Security makes use of overloaded subprogram names. That is, the same name is used for several different procedures whose formal parameters differ in number, order, or datatype family.

For example, you can call the SA_COMPONENTS.ALTER_LEVEL procedure this way:

```
PROCEDURE ALTER_LEVEL (policy_name IN VARCHAR2,
   level_num       IN INTEGER,
   new_short_name  IN VARCHAR2 DEFAULT NULL,
   new_long_name   IN VARCHAR2 DEFAULT NULL);
```

or this way:

```
PROCEDURE ALTER_LEVEL (policy_name IN VARCHAR2,
   short_name      IN VARCHAR2,
  new_long_name   IN VARCHAR2);
```

Because the processing in these two procedures is the same, it is logical to give them the same name. PL/SQL determines which of the two procedures is being called by checking their formal parameters. In the preceding example, the version of `initialize` used by PL/SQL depends on whether you call the procedure with a `level_num` or `short_name` parameter.

Creating a Level with SA_COMPONENTS.CREATE_LEVEL

Use the CREATE_LEVEL procedure to create a level and specify its short name and long name. The numeric values assigned to the *level_num* parameter determine the sensitivity ranking (that is, a lower number indicates less sensitive data).

Syntax:

```
PROCEDURE CREATE_LEVEL (policy_name IN VARCHAR2,
   level_num         IN INTEGER,
   short_name        IN VARCHAR2,
   long_name         IN VARCHAR2);
```

Table 7–7 Parameters for SA_COMPONENTS.CREATE_LEVEL

Parameter Name	Parameter Description
policy_name	Specifies the policy
level_num	Specifies the level number (0-9999)
short_name	Specifies the short name for the level (up to 30 characters)
long_name	Specifies the long name for the level (up to 80 characters)

Modifying a Level with SA_COMPONENTS.ALTER_LEVEL

Use the ALTER_LEVEL procedure to change the short name and long name associated with a level.

Once they are defined, level numbers cannot be changed. If a level is used in any existing label, then its short name *cannot* be changed, but its long name *can* be changed.

Syntax:

```
PROCEDURE ALTER_LEVEL (policy_name IN VARCHAR2,
   level_num       IN INTEGER,
   new_short_name  IN VARCHAR2 DEFAULT NULL,
   new_long_name   IN VARCHAR2 DEFAULT NULL);

PROCEDURE ALTER_LEVEL (policy_name IN VARCHAR2,
   short_name      IN VARCHAR2,
   new_long_name   IN VARCHAR2);
```

Table 7–8 Parameters for SA_COMPONENTS.ALTER_LEVEL

Parameter Name	Parameter Description
policy_name	Specifies the policy
level_num	Specifies the number of the level to be altered
short_name	Specifies the short name for the level (up to 30 characters)
new_short_name	Specifies the new short name for the level (up to 30 characters)
new_long_name	Specifies the new long name for the level (up to 80 characters)

Removing a Level with SA_COMPONENTS.DROP_LEVEL

Use the DROP_LEVEL procedure to remove a level. If the level is used in any existing label, then it cannot be dropped.

Syntax:

```
PROCEDURE DROP_LEVEL (policy_name IN VARCHAR2,
   level_num   IN INTEGER);

PROCEDURE DROP_LEVEL (policy_name IN VARCHAR2,
   short_name  IN VARCHAR2);
```

Table 7–9 Parameters for SA_COMPONENTS.DROP_LEVEL

Parameter Name	Parameter Description
policy_name	Specifies the policy
level_num	Specifies the number of an existing level for the policy
short_name	Specifies the short name for the level (up to 30 characters)

Creating a Compartment with SA_COMPONENTS.CREATE_COMPARTMENT

Use the CREATE_COMPARTMENT procedure to create a compartment and specify its short name and long name. The *comp_num* parameter determines the order in which compartments are listed in the character string representation of labels.

Syntax:

```
PROCEDURE CREATE_COMPARTMENT (policy_name IN VARCHAR2,
   comp_num    IN INTEGER,
   short_name  IN VARCHAR2,
   long_name   IN VARCHAR2);
```

Table 7–10 Parameters for SA_COMPONENTS.CREATE_COMPARTMENT

Parameter Name	Parameter Description
policy_name	Specifies the policy
comp_num	Specifies the compartment number (0-9999)
short_name	Specifies the short name for the compartment (up to 30 characters)
long_name	Specifies the long name for the compartment (up to 80 characters)

Modifying a Compartment with SA_COMPONENTS.ALTER_COMPARTMENT

Use the ALTER_COMPARTMENT procedure to change the short name and long name associated with a compartment.

Once set, the *comp_num* parameter cannot be changed. If the *comp_num* parameter is used in any existing label, then its short name *cannot* be changed but its long name *can* be changed.

Syntax:

```
PROCEDURE ALTER_COMPARTMENT (policy_name IN VARCHAR2,
   comp_num          IN INTEGER,
   new_short_name    IN VARCHAR2 DEFAULT NULL,
   new_long_name     IN VARCHAR2 DEFAULT NULL);

PROCEDURE ALTER_COMPARTMENT (policy_name IN VARCHAR2,
   short_name        IN VARCHAR2,
   new_long_name     IN VARCHAR2);
```

Table 7–11 Parameters for SA_COMPONENTS.ALTER_COMPARTMENT

Parameter Name	Parameter Description
policy_name	Specifies the policy
comp_num	Specifies the number of the compartment to be altered

Table 7–11 (Cont.) Parameters for SA_COMPONENTS.ALTER_COMPARTMENT

Parameter Name	Parameter Description
short_name	Specifies the short name of the compartment to be altered (up to 30 characters)
new_short_name	Specifies the new short name of the compartment (up to 30 characters)
new_long_name	Specifies the new long name of the compartment (up to 80 characters).

Removing a Compartment with SA_COMPONENTS.DROP_COMPARTMENT

Use the DROP_COMPARTMENT procedure to remove a compartment. If the compartment is used in any existing label, then it *cannot* be dropped.

Syntax:

```
PROCEDURE DROP_COMPARTMENT (policy_name IN VARCHAR2,
   comp_num    IN INTEGER);

PROCEDURE DROP_COMPARTMENT (policy_name IN VARCHAR2,
   short_name  IN VARCHAR2);
```

Table 7–12 Parameters for SA_COMPONENTS.DROP_COMPARTMENT

Parameter Name	Parameter Description
policy_name	Specifies the policy
comp_num	Specifies the number of an existing compartment for the policy
short_name	Specifies the short name of an existing compartment for the policy

Creating a Group with SA_COMPONENTS.CREATE_GROUP

Use the CREATE_GROUP procedure to create a group and specify its short name and long name, and optionally a parent group.

Syntax:

```
PROCEDURE CREATE_GROUP (policy_name IN VARCHAR2,
   group_num   IN INTEGER,
   short_name  IN VARCHAR2,
   long_name   IN VARCHAR2,
   parent_name IN VARCHAR2 DEFAULT NULL);
```

Table 7–13 Parameters for SA_COMPONENTS.CREATE_GROUP

Parameter Name	Parameter Description
policy_name	Specifies the policy
group_num	Specifies the group number (0-9999)
short_name	Specifies the short name for the group (up to 30 characters)
long_name	Specifies the long name for the group (up to 80 characters)
parent_name	Specifies the short name of an existing group as the parent group. If NULL, then the group is a top-level group.

Note that the group number affects the order in which groups will be displayed when labels are selected.

See Also: "Groups" on page 2-6

Modifying a Group with SA_COMPONENTS.ALTER_GROUP

Use the ALTER_GROUP procedure to change the short name and long name associated with a group.

Once set, the *group_num* parameter cannot be changed. If the group is used in any existing label, then its short name *cannot* be changed, but its long name *can* be changed.

Syntax:

```
PROCEDURE ALTER_GROUP (policy_name IN VARCHAR2,
   group_num      IN INTEGER,
   new_short_name IN VARCHAR2 DEFAULT NULL,
   new_long_name  IN VARCHAR2 DEFAULT NULL);

PROCEDURE ALTER_GROUP (policy_name IN VARCHAR2,
   short_name     IN VARCHAR2,
   new_long_name  IN VARCHAR2);
```

Table 7–14 Parameters for SA_COMPONENTS.ALTER_GROUP

Parameter Name	Parameter Description
policy_name	Specifies the policy
group_num	Specifies the existing group number to be altered
short_name	Specifies the existing group short name to be altered
new_short_name	Specifies the new short name for the group (up to 30 characters)
new_long_name	Specifies the new long name for the group (up to 80 characters)

Modifying a Group Parent with SA_COMPONENTS.ALTER_GROUP_PARENT

The ALTER_GROUP_PARENT procedure changes the parent group associated with a particular group.

Syntax:

```
PROCEDURE ALTER_GROUP_PARENT (policy_name IN VARCHAR2,
   group_num   IN INTEGER,
   parent_name IN VARCHAR2);

PROCEDURE ALTER_GROUP_PARENT (policy_name IN VARCHAR2,
   group_num   IN INTEGER,
   parent_num  IN INTEGER);

PROCEDURE ALTER_GROUP_PARENT (policy_name IN VARCHAR2,
   short_name  IN VARCHAR2,
   parent_name IN VARCHAR2);
```

Table 7–15 Parameters for SA_COMPONENTS.ALTER_GROUP_PARENT

Parameter Name	Parameter Description
policy_name	Specifies the policy
group_num	Specifies the existing group number to be altered
short_name	Specifies the existing group short name to be altered

Table 7–15 (Cont.) Parameters for SA_COMPONENTS.ALTER_GROUP_PARENT

Parameter Name	Parameter Description
parent_num	Specifies the number of an existing group as the parent group
parent_name	Specifies the short name of an existing group as the parent group

Removing a Group with SA_COMPONENTS.DROP_GROUP

Use the DROP_GROUP procedure to remove a group. If the group is used in an existing label, it *cannot* be dropped.

Syntax:

```
PROCEDURE DROP_GROUP (policy_name IN VARCHAR2,
   group_num   IN INTEGER);

PROCEDURE DROP_GROUP (policy_name IN VARCHAR2,
   short_name  IN VARCHAR2);
```

Table 7–16 Parameters for SA_COMPONENTS.DROP_GROUP

Parameter Name	Parameter Description
policy_name	Specifies the policy
group_num	Specifies the number of an existing group for the policy
short_name	Specifies the short name of an existing group

Using the SA_LABEL_ADMIN Package to Specify Valid Labels

The SA_LABEL_ADMIN package provides an administrative interface to manage the labels used by a policy. To do this, a user must have the EXECUTE privilege for the SA_LABEL_ADMIN package and have been granted the *policy*_DBA role.

This section includes:

- Creating a Valid Data Label with SA_LABEL_ADMIN.CREATE_LABEL
- Modifying a Label with SA_LABEL_ADMIN.ALTER_LABEL
- Deleting a Label with SA_LABEL_ADMIN.DROP_LABEL

Creating a Valid Data Label with SA_LABEL_ADMIN.CREATE_LABEL

Use the SA_LABEL_ADMIN.CREATE_LABEL procedure to create a valid data label. You must manually specify a label tag value from 1 to 8 digits long.

Syntax:

```
PROCEDURE CREATE_LABEL (
   policy_name IN VARCHAR2,
   label_tag   IN INTEGER,
   label_value IN VARCHAR2,
   data_label  IN BOOLEAN DEFAULT TRUE);
```

Table 7–17 Parameters for SA_LABEL_ADMIN.CREATE_LABEL

Parameter Name	Parameter Description
policy_name	Specifies the name of an existing policy

Table 7–17 (Cont.) Parameters for SA_LABEL_ADMIN.CREATE_LABEL

Parameter Name	Parameter Description
label_tag	Specifies a unique integer value representing the sort order of the label, relative to other policy labels (0-99999999)
label_value	Specifies the character string representation of the label to be created
data_label	TRUE if the label can be used to label row data. Use this to define the label as valid for data.

When specifying labels, use the short name of the level, compartment, and group.

When you identify valid labels, you specify which of all the possible combinations of levels, compartments, and groups can potentially be used to label data in tables.

Note: If you create a new label by using the TO_DATA_LABEL procedure, a system-generated label tag of 10 digits will be generated automatically.

However, when Oracle Label Security is installed to work with Oracle Internet Directory, dynamic label generation is not permitted, because labels are managed centrally in Oracle Internet Directory, using `olsadmintool` commands. Refer to Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory".

So, when Oracle Label Security is directory-enabled, the TO_DATA_LABEL function is not available and will generate an error message if used.

See Also: "The Policy Label Column and Label Tags" on page 5-1

Modifying a Label with SA_LABEL_ADMIN.ALTER_LABEL

Use the ALTER_LABEL procedure to change the character string label definition associated with a label tag. Note that the label tag itself cannot be changed.

If you change the character string associated with a label tag, the sensitivity of the data in the rows changes accordingly. For example, if the label character string TS:A with an associated label tag value of 4001 is changed to the label TS:B, then access to the data changes accordingly. This is true even when the label tag value (4001) has not changed. In this way, you can change the data's sensitivity without the need to update all the rows.

Ensure that when you specify a label to alter, you can refer to it either by its label tag or by its character string value.

Syntax:

```
PROCEDURE ALTER_LABEL (
   policy_name       IN VARCHAR2,
   label_tag         IN INTEGER,
   new_label_value   IN VARCHAR2 DEFAULT NULL,
   new_data_label    IN BOOLEAN  DEFAULT NULL);

PROCEDURE ALTER_LABEL (
   policy_name       IN VARCHAR2,
   label_value       IN VARCHAR2,
```

```
   new_label_value   IN VARCHAR2 DEFAULT NULL,
   new_data_label    IN BOOLEAN  DEFAULT NULL);
```

Table 7–18 Parameters for SA_LABEL_ADMIN.ALTER_LABEL

Parameter Name	Parameter Description
policy_name	Specifies the name of an existing policy
label_tag	Identifies the integer tag assigned to the label to be altered
label_value	Identifies the existing character string representation of the label to be altered
new_label_value	Specifies the new character string representation of the label value. If NULL, the existing value is not changed.
new_data_label	TRUE if the label can be used to label row data. If NULL, the existing value is not changed.

Deleting a Label with SA_LABEL_ADMIN.DROP_LABEL

Use the SA_LABEL_ADMIN.DROP_LABEL procedure to delete a specified policy label. Any subsequent reference to the label (in data rows, or in user or program unit labels) will raise an invalid label error.

Syntax:

```
PROCEDURE DROP_LABEL (
   policy_name       IN VARCHAR2,
   label_tag         IN INTEGER);

PROCEDURE DROP_LABEL (
   policy_name       IN VARCHAR2,
   label_value       IN VARCHAR2);
```

Table 7–19 Parameters for SA_LABEL_ADMIN.DROP_LABEL

Parameter Name	Parameter Description
policy_name	Specifies the name of an existing policy
label_tag	Specifies the integer tag assigned to the label to be dropped
label_value	Specifies the string value of the label to be dropped

Caution: Do not drop a label that is in use anywhere in the database.

Use this procedure only while setting up labels, prior to data population. If you should inadvertently drop a label that is being used, you can recover it by disabling the policy, fixing the problem, and then re-enabling the policy.

8

Administering User Labels and Privileges

This chapter discusses using Oracle Label Security packages to administer user labels and privileges. You can also use the Web interface provided by Oracle Enterprise Manager Database Control or Grid Control to administer these. This is discussed in Chapter 4, "Getting Started with Oracle Label Security".

This chapter includes the following topics:

- Introduction to User Label and Privilege Management
- Managing User Labels by Component, with SA_USER_ADMIN
- Managing User Labels by Label String, with SA_USER_ADMIN
- Managing User Privileges with SA_USER_ADMIN.SET_USER_PRIVS
- Setting Labels & Privileges with SA_SESSION.SET_ACCESS_PROFILE
- Returning User Name with SA_SESSION.SA_USER_NAME
- Using Oracle Label Security Views

Introduction to User Label and Privilege Management

To manage user labels and privileges, you must have the EXECUTE privilege for the SA_USER_ADMIN package, and must have been granted the *policy*_DBA role.

The SA_USER_ADMIN package provides the functions to manage the Oracle Label Security user security attributes. It contains several procedures to manage user labels by component: that is, specifying user levels, compartments, and groups. For convenience, there are additional procedures that accept character string representations of full labels, rather than components. Note that the level, compartment and group parameters use the short name defined for each component.

All of the label and privilege information is stored in Oracle Label Security data dictionary tables. When a user connects to the database, his session labels are established based on the information stored in the Oracle Label Security data dictionary.

Note that a user can be authorized under multiple policies.

Managing User Labels by Component, with SA_USER_ADMIN

The following SA_USER_ADMIN procedures enable you to manage user labels by label component:

- SA_USER_ADMIN.SET_LEVELS

- SA_USER_ADMIN.SET_COMPARTMENTS
- SA_USER_ADMIN.SET_GROUPS
- SA_USER_ADMIN.ADD_COMPARTMENTS
- SA_USER_ADMIN.ALTER_COMPARTMENTS
- SA_USER_ADMIN.DROP_COMPARTMENTS
- SA_USER_ADMIN.DROP_ALL_COMPARTMENTS
- SA_USER_ADMIN.ADD_GROUPS
- SA_USER_ADMIN.ALTER_GROUPS
- SA_USER_ADMIN.DROP_GROUPS
- SA_USER_ADMIN.DROP_ALL_GROUPS

SA_USER_ADMIN.SET_LEVELS

The SET_LEVELS procedure assigns a minimum and maximum level to a user and identifies default values for the user's session label and row label.

- If the *min_level* is NULL, then it is set to the lowest defined level for the policy.
- If the *def_level* is not specified, then it is set to the *max_level*.
- If the *row_level* is not specified, then it is set to the *def_level*.

Syntax:

```
PROCEDURE SET_LEVELS (policy_name IN VARCHAR2,
   user_name        IN VARCHAR2,
   max_level        IN VARCHAR2,
   min_level        IN VARCHAR2 DEFAULT NULL,
   def_level        IN VARCHAR2 DEFAULT NULL,
   row_level        IN VARCHAR2 DEFAULT NULL);
```

Table 8–1 Parameters for SA_USER_ADMIN.SET_LEVELS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
max_level	The highest level for read and write access
min_level	The lowest level for write access
def_level	Specifies the default level (equal to or greater than the minimum level, and equal to or less than the maximum level)
row_level	Specifies the row level (equal to or greater than the minimum level, and equal to or less than the default level)

SA_USER_ADMIN.SET_COMPARTMENTS

The SET_COMPARTMENTS procedure assigns compartments to a user and identifies default values for the user's session label and row label.

- If *write_comps* are NULL, then they are set to the *read_comps*.
- If the *def_comps* are NULL, then they are set to the *read_comps*.
- If the *row_comps* are NULL, then they are set to the components in *def_comps* that are authorized for write access.

All users must have their levels set before their authorized compartments can be established.

The write compartments, if specified, must be a subset of the read compartments. (The write compartments are those to which the user should have write access.)

Syntax:

```
PROCEDURE SET_COMPARTMENTS (policy_name IN VARCHAR2,
  user_name     IN VARCHAR2,
  read_comps    IN VARCHAR2,
  write_comps   IN VARCHAR2 DEFAULT NULL,
  def_comps     IN VARCHAR2 DEFAULT NULL,
  row_comps     IN VARCHAR2 DEFAULT NULL);
```

Table 8–2 Parameters for SA_USER_ADMIN.SET_COMPARTMENTS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
read_comps	A comma-delimited list of compartments authorized for read access
write_comps	A comma-delimited list of compartments authorized for write access (subset of *read_comps*)
def_comps	Specifies the default compartments. This must be a subset of *read_comps*.
row_comps	Specifies the row compartments. This must be a subset of *write_comps* and *def_comps*.

SA_USER_ADMIN.SET_GROUPS

The SET_GROUPS procedure assigns groups to a user and identifies default values for the user's session label and row label.

- If the *write_groups* are NULL, they are set to the *read_groups*.
- If the *def_groups* are NULL, they are set to the *read_groups*.
- If the *row_groups* are NULL, they are set to the groups in *def_groups* that are authorized for write access.

All users must have their levels set before their authorized groups can be established.

Syntax:

```
PROCEDURE SET_GROUPS (policy_name IN VARCHAR2,
  user_name        IN VARCHAR2,
  read_groups      IN VARCHAR2,
  write_groups     IN VARCHAR2 DEFAULT NULL,
  def_group        IN VARCHAR2 DEFAULT NULL,
  row_groups       IN VARCHAR2 DEFAULT NULL);
```

Table 8–3 Parameters for SA_USER_ADMIN.SET_GROUPS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
read_groups	A comma-delimited list of groups authorized for read

Table 8–3 (Cont.) Parameters for SA_USER_ADMIN.SET_GROUPS

Parameter	Meaning
write_groups	A comma-delimited list of groups authorized for write. This must be a subset of read_groups.
def_groups	Specifies the default groups. This must be a subset of *read_groups*
row_groups	Specifies the row groups. This must be a subset of *write_groups* and *def_groups*.

SA_USER_ADMIN.ALTER_COMPARTMENTS

The ALTER_COMPARTMENTS procedure changes the write access, the default label indicator, and the row label indicator for each of the compartments in the list.

Syntax:

```
PROCEDURE ALTER_COMPARTMENTS (policy_name IN VARCHAR2,
  user_name    IN VARCHAR2,
  comps        IN VARCHAR2,
  access_mode  IN VARCHAR2 DEFAULT NULL,
  in_def       IN VARCHAR2 DEFAULT NULL,
  in_row       IN VARCHAR2 DEFAULT NULL);
```

Table 8–4 Parameters for SA_USER_ADMIN.ALTER_COMPARTMENTS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
comps	A comma-delimited list of compartments to modify
access_mode	One of two public variables that contain string values that can specify the type of access authorized. The variable names, values, and meaning are as follows: SA_UTL.READ_ONLY READ_ONLY Indicates no write access SA_UTL.READ_WRITE READ_WRITE Indicates that write is authorized If *access_mode* is NULL, then access_mode for the compartment is unaltered.
in_def	Specifies whether these compartments should be in the default compartments (Y/N) If *in_def* is NULL, then in_def for the compartment is unaltered.
in_row	Specifies whether these compartments should be in the row label (Y/N) If *in_row* is NULL, then in_row for the compartment is unaltered.

SA_USER_ADMIN.ADD_COMPARTMENTS

This procedure adds compartments to a user's authorizations, indicating whether the compartments are authorized for write as well as read.

Syntax:

```
PROCEDURE ADD_COMPARTMENTS (policy_name IN VARCHAR2,
user_name      IN VARCHAR2,
comps          IN VARCHAR2,
access_model   IN VARCHAR2 DEFAULT NULL,
in_def         IN VARCHAR2 DEFAULT NULL,
in_row         IN VARCHAR2 DEFAULT NULL);
```

Table 8–5 Parameters for SA_USER_ADMIN.ADD_COMPARTMENTS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
comps	A comma-delimited list of read compartments to add
access_mode	One of two public variables that contain string values that can specify the type of access authorized. The variable names, values, and meaning are as follows: SA_UTL.READ_ONLY READ_ONLY Indicates no write access SA_UTL.READ_WRITE READ_WRITE Indicates that write is authorized If *access_mode* is NULL, then it is set to SA_UTL.READ_ONLY.
in_def	Specifies whether these compartments should be in the default compartments (Y/N) If *in_def* is NULL, then it is set to Y.
in_row	Specifies whether these compartments should be in the row label (Y/N) If *in_row* is NULL, then it is set to N.

SA_USER_ADMIN.DROP_COMPARTMENTS

The DROP_COMPARTMENTS procedure drops the specified compartments from a user's authorizations.

Syntax:

```
PROCEDURE DROP_COMPARTMENTS (policy_name IN VARCHAR2,
  user_name       IN VARCHAR2,
  comps           IN VARCHAR2);
```

Table 8–6 Parameters for SA_USER_ADMIN.DROP_COMPARTMENTS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
comps	A comma-delimited list of compartments to drop

SA_USER_ADMIN.DROP_ALL_COMPARTMENTS

The DROP_ALL_COMPARTMENTS procedure drops all compartments from a user's authorizations.

Syntax:

```
PROCEDURE DROP_ALL_COMPARTMENTS (policy_name IN VARCHAR2,
     user_name IN VARCHAR2);
```

Table 8–7 Parameters for SA_USER_ADMIN.DROP_ALL_COMPARTMENTS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name

SA_USER_ADMIN.ADD_GROUPS

The ADD_GROUPS procedure adds groups to a user, indicating whether the groups are authorized for write as well as read.

Syntax:

```
PROCEDURE ADD_GROUPS (policy_name IN VARCHAR2,
  user_name         IN VARCHAR2,
  groups            IN VARCHAR2,
  access_mode       IN VARCHAR2 DEFAULT NULL,
  in_def            IN VARCHAR2 DEFAULT NULL,
  in_row            IN VARCHAR2 DEFAULT NULL);
```

Table 8–8 Parameters for SA_USER_ADMIN.ADD_GROUPS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
groups	A comma-delimited list of read groups to add
access_mode	One of two public variables that contain string values that can specify the type of access authorized. The variable names, values, and meaning are as follows: SA_UTL.READ_ONLY READ_ONLY Indicates no write access SA_UTL.READ_WRITE READ_WRITE Indicates that write is authorized If *access_mode* is NULL, then access_mode is set to SA_UTL.READ_ONLY.
in_def	Specifies whether these groups should be in the default groups (Y/N) If *in_def* is NULL, then it is set to Y.
in_row	Specifies whether these groups should be in the row label (Y/N) If *in_row* is NULL, then it is set to N.

SA_USER_ADMIN.ALTER_GROUPS

The ALTER_GROUPS procedure changes the write access, the default label indicator, and the row label indicator for each of the groups in the list.

Syntax:

```
PROCEDURE ALTER_GROUPS (policy_name IN VARCHAR2,
  user_name        IN VARCHAR2,
  groups           IN VARCHAR2,
  access_mode      IN VARCHAR2 DEFAULT NULL,
  in_def           IN VARCHAR2 DEFAULT NULL,
  in_row           IN VARCHAR2 DEFAULT NULL);
```

Table 8–9 Parameters for SA_USER_ADMIN.ALTER_GROUPS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
groups	A comma-delimited list of groups to alter
access_mode	Two public variables contain string values that can specify the type of access authorized. The variable names, values, and meaning are as follows: SA_UTL.READ_ONLY READ_ONLY Indicates no write access SA_UTL.READ_WRITE READ_WRITE Indicates that write is authorized If *access_mode* is NULL, then access_mode for the group is unaltered.
in_def	Specifies whether these groups should be in the default groups (Y/N) If *in_def* is NULL, then in_def for the group is unaltered.
in_row	Specifies whether these groups should be in the row label (Y/N) If *in_row* is NULL, then in_row for the group is unaltered.

SA_USER_ADMIN.DROP_GROUPS

The DROP_GROUPS procedure drops the specified groups from a user's authorizations.

Syntax:

```
PROCEDURE DROP_GROUPS (policy_name IN VARCHAR2,
  user_name   IN VARCHAR2,
  groups      IN VARCHAR2);
```

Table 8–10 Parameters for SA_USER_ADMIN.DROP_GROUPS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
groups	A comma-delimited list of groups to drop

SA_USER_ADMIN.DROP_ALL_GROUPS

The DROP_ALL_GROUPS procedure drops all groups from a user's authorizations.

Syntax:

```
PROCEDURE DROP_ALL_GROUPS (policy_name IN VARCHAR2,
  user_name  IN VARCHAR2);
```

Table 8–11 Parameters for SA_USER_ADMIN.DROP_ALL_GROUPS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name

Managing User Labels by Label String, with SA_USER_ADMIN

The following SA_USER_ADMIN procedures enable you to manage user labels by specifying the complete character label string:

- SA_USER_ADMIN.SET_USER_LABELS
- SA_USER_ADMIN.SET_DEFAULT_LABEL
- SA_USER_ADMIN.SET_ROW_LABEL
- SA_USER_ADMIN.SET_DEFAULT_LABEL

SA_USER_ADMIN.SET_USER_LABELS

The SET_USER_LABELS procedure sets the user's levels, compartments, and groups using a set of labels, instead of the individual components.

Syntax:

```
PROCEDURE SET_USER_LABELS (
  policy_name      IN VARCHAR2,
  user_name        IN VARCHAR2,
  max_read_label   IN VARCHAR2,
  max_write_label  IN VARCHAR2 DEFAULT NULL,
  min_write_label  IN VARCHAR2 DEFAULT NULL,
  def_label        IN VARCHAR2 DEFAULT NULL,
  row_label        IN VARCHAR2 DEFAULT NULL);
```

Table 8–12 Parameters for SA_USER_ADMIN.SET_USER_LABELS

Parameter	Meaning
max_read_label	Specifies the label string to be used to initialize the user's maximum authorized read label. Composed of the user's maximum level, compartments authorized for read access, and groups authorized for read access.
max_write_label	Specifies the label string to be used to initialize the user's maximum authorized write label. Composed of the user's maximum level, compartments authorized for write access, and groups authorized for write access. If *max_write_label* is not specified, then it is set to *max_read_label*.
min_write_label	Specifies the label string to be used to initialize the user's minimum authorized write label. Contains only the level, with no compartments or groups. If *min_write_label* is not specified, then it is set to the lowest defined level for the policy, with no compartments or groups.
def_label	Specifies the label string to be used to initialize the user's session label, including level, compartments, and groups (a subset of *max_read_label*). If *default_label* is not specified, then it is set to *max_read_label*.
policy_name	Specifies the policy
user_name	Specifies the user name
row_label	Specifies the label string to be used to initialize the program's row label. Includes level, components, and groups: subsets of *max_write_label* and *def_label*. If *row_label* is not specified, then it is set to *def_label*, with only the compartments and groups authorized for write access.

See Also: "Managing Program Unit Privileges with SET_PROG_PRIVS" on page 11-2

SA_USER_ADMIN.SET_DEFAULT_LABEL

The SET_DEFAULT_LABEL procedure sets the user's initial session label to the one specified.

Syntax:

```
PROCEDURE SET_DEFAULT_LABELS (
  policy_name  IN VARCHAR2,
  user_name    IN VARCHAR2,
  def_label    IN VARCHAR2);
```

Table 8–13 Parameters for SA_USER_ADMIN.SET_DEFAULT_LABEL

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
def_label	Specifies the label string to be used to initialize the user's default labels. This label may contain any compartments and groups that are authorized for read access.

As long as the row label will still be dominated by the new write label, the user can set the session label to:

- Any level equal to or less than his maximum, and equal to or greater than his minimum label
- Include any compartments in the authorized compartment list
- Include any groups in the authorized group list. (Subgroups of authorized groups are implicitly included in the authorized list.)

The row label must be dominated by the new write label that will result from resetting the session label. If this condition is not true, then the SET_DEFAULT_LABEL procedure will fail.

For example, suppose the current row label is S:A,B, and that you have write access to both compartments. If you attempt to set the new default label to C:A,B, then the SET_LABEL procedure will fail. This is because the new write label would be C:A,B, which does not dominate the current row label.

To successfully reset the session label in this case, you must first lower the row label to a value that will be dominated by the resulting session label.

See Also: "Changing Your Session and Row Labels with SA_SESSION" on page 5-14

"Session Labels and Inverse Groups" on page 15-9

SA_USER_ADMIN.SET_ROW_LABEL

Use the SET_ROW_LABEL procedure to set the user's initial row label to the one specified.

Syntax:

```
PROCEDURE SET_ROW_LABEL (
  policy_name   IN VARCHAR2,
```

```
  user_name     IN VARCHAR2,
  row_label     IN VARCHAR2);
```

Table 8–14 Parameters for SA_USER_ADMIN.SET_ROW_LABEL

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name
row_label	Specifies the label string to be used to initialize the user's row label. The label must contain only those compartments and groups from the default label that are authorized for write access.

The user can set the row label independently, but only to:

- A level that is less than or equal to the level of the session label, and greater than or equal to the user's minimum level
- Include a subset of the compartments and groups from the session label, for which the user is authorized to have write access

If you try to set the row label to an invalid value, then the operation is disallowed, and the row label value is unchanged.

See Also: "Changing the Row Label with SA_SESSION.SET_ROW_LABEL" on page 5-15

SA_USER_ADMIN.DROP_USER_ACCESS

Use the DROP_USER_ACCESS procedure to remove all Oracle Label Security authorizations and privileges from the specified user. This procedure must be issued from the command line.

Syntax:

```
PROCEDURE DROP_USER_ACCESS (
  policy_name      IN VARCHAR2,
  user_name        IN VARCHAR2);
```

Table 8–15 Parameters for SA_USER_ADMIN.DROP_USER_ACCESS

Parameter	Meaning
policy_name	Specifies the policy
user_name	Specifies the user name

Managing User Privileges with SA_USER_ADMIN.SET_USER_PRIVS

The SET_USER_PRIVS procedure sets policy-specific privileges for users. These privileges do not become effective in the current session. However, they become effective the next time the user logs in. The new set of privileges replaces any existing privileges. A NULL value for the privileges parameter removes the user's privileges for the policy.

To assign policy privileges to users, you must have the EXECUTE privilege for the SA_USER_ADMIN package, and must have been granted the *policy*_DBA role.

Syntax:

```
PROCEDURE SET_USER_PRIVS (
```

```
  policy_name     IN VARCHAR2,
  user_name       IN VARCHAR2,
  privileges      IN VARCHAR2);
```

Table 8–16 Parameters for SA_USER_ADMIN.SET_USER_PRIVS

Parameter	Meaning
policy_name	Specifies the policy name of an existing policy
user_name	The name of the user to be granted privileges
privileges	A character string of policy-specific privileges separated by commas

See Also: "Managing Program Unit Privileges with SET_PROG_PRIVS" on page 11-2

Setting Labels & Privileges with SA_SESSION.SET_ACCESS_PROFILE

The SET_ACCESS_PROFILE procedure sets the Oracle Label Security authorizations and privileges of the database session to those of the specified user. (Note that the originating user retains the PROFILE_ACCESS privilege.)

The user executing the SA_SESSION.SET_ACCESS_PROFILE procedure must have the PROFILE_ACCESS privilege. Note that the logged-in database user (the Oracle userid) does not change. That user assumes only the authorizations and privileges of the specified user. By contrast, the Oracle Label Security user name *is* changed.

This administrative procedure is useful for various tasks:

- With SET_ACCESS_PROFILE, the administrator can see the result of the authorization and privilege settings for a particular user.
- Applications need to have proxy accounts connect as (and assume the identity of) application users, for purposes of accessing labeled data. With the SET_ACCESS_PROFILE privilege, the proxy account can act on behalf of the application users.

Syntax:

```
PROCEDURE SET_ACCESS_PROFILE (policy_name IN VARCHAR2
  user_name   IN VARCHAR2);
```

Table 8–17 Parameters for SA_SESSION.SET_ACCESS_PROFILE

Parameter	Meaning
policy_name	The name of an existing policy
user_name	Name of the user whose authorizations and privileges should be assumed

Returning User Name with SA_SESSION.SA_USER_NAME

The SA_USER_NAME function returns the name of the current Oracle Label Security user, as set by the SET_ACCESS_PROFILE procedure (or as established at login). This is how you can determine the identity of the current user in relation to Oracle Label Security, rather than in relation to your Oracle login name.

Syntax:

```
FUNCTION SA_USER_NAME (policy_name IN VARCHAR2)
RETURN VARCHAR2;
```

Table 8–18 Parameters for SA_SESSION.SA_USER_NAME

Parameter	Meaning
policy_name	The name of an existing policy

Using Oracle Label Security Views

This section describes views you can use to see the user authorization and privilege assignments made by the administrator.

- View to Display All User Security Attributes: DBA_SA_USERS
- Views to Display User Authorizations by Component

View to Display All User Security Attributes: DBA_SA_USERS

The DBA_SA_USERS view displays the values assigned for privileges, levels, compartments, and groups all together, corresponding to how you enter these values through the SA_USER_ADMIN command-line interface. The values include:

USER_PRIVILEGES

MAX_READ_LABEL

MAX_WRITE_LABEL

MIN_WRITE_LABEL

DEFAULT_READ_LABEL

DEFAULT_WRITE_LABEL

DEFAULT_ROW_LABEL

USER_LABELS

MAX_READ_LABEL

MAX_WRITE_LABEL

MIN_WRITE_LABEL

DEFAULT_READ_LABEL

DEFAULT_WRITE_LABEL

DEFAULT_ROW_LABEL

This information is stored in data dictionary tables, and used to establish session and row labels when a user logs in.

Note: The field USER_LABELS in DBA_SA_USERS is retained solely for backward compatibility and will be removed in the next release.

Views to Display User Authorizations by Component

The following views individually display each component of the label:

Table 8–19 Oracle Label Security Views

View	Contents
DBA_SA_USER_LEVELS	Displays the levels assigned to the user: minimum level, maximum level, default level, and level for the row label
DBA_SA_USER_COMPARTMENTS	Displays the compartments assigned to the user
DBA_SA_USER_GROUPS	Displays the groups assigned to the user

9

Implementing Policy Enforcement Options and Labeling Functions

This chapter explains how to customize the enforcement of Oracle Label Security policies and how to implement labeling functions, in the following sections:

Choosing Policy Options

This section introduces the policy options, and discusses their use.

Overview of Policy Enforcement Options

Of all the enforcement controls that Oracle Label Security permits, the administrator must choose those that meet the needs of the given application. This means identifying levels of data sensitivity to exposure, alteration, or misuse, as well as identifying which users have the need or the right to access or alter such data. The policy enforcement options enable administrators to fine-tune users' abilities to read or write data or labels.

These options can operate at three levels:

Table 9–1 When Policy Enforcement Options Take Effect

Level at which option set	Options set at this level affect user operations ...
Policy Level	... only when the policy has been applied to the table or schema
Schema Level	... whenever a user acts in this schema
Table Level	... whenever a user acts in this table

When you apply a policy to a table or schema, you can specify the enforcement options that are to constrain use of that table or schema. If you do not specify enforcement options at that time, then the default enforcement options you specified when you created that policy are used automatically.

See Also:

- "Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY" on page 10-3
- "Creating a Policy with SA_SYSDBA.CREATE_POLICY" on page 7-11

These options customize your policy enforcement to meet your security requirements as to READ access, WRITE access, and label changes. You can also specify whether the label column should be displayed or hidden. You can choose to enforce some or all of the policy options for any protected table by specifying only those you want.

Optionally, you can assign each table a labeling function, which determines the label of any row inserted or updated in that table. You can also specify, optionally, a *SQL* predicate for a table, to control which rows are accessible to users, based on their labels.

See Also:

- Using a Labeling Function on page 9-9.
- Using a SQL Predicate with an Oracle Label Security Policy on page 9-15.

When Oracle Label Security policy enforcement options are applied, they control which rows are accessible to view or to insert, update, or delete.

Table 9–2, " Policy Enforcement Options" lists the options in three categories:

- Label management options, ensuring that data labels written for inserted or updated rows do not violate policies set for such labels
- Access control options, ensuring that only rows whose labels meet established policies are accessible for SELECT, UPDATE, INSERT, or DELETE operations.
- Overriding options, which can suspend or apply all other enforcement options.

Table 9–2 Policy Enforcement Options

Type of Enforcement	Option	Description
The Label Management Enforcement Options	LABEL_DEFAULT	Uses the session's default row label value unless the user explicitly specifies a label on INSERT.
	LABEL_UPDATE	Applies policy enforcement to UPDATE operations that set or change the value of a label attached to a row. The WRITEUP, WRITEDOWN, and WRITEACROSS privileges are enforced only if the LABEL_UPDATE option is active.
	CHECK_ CONTROL	Applies READ_CONTROL policy enforcement to INSERT and UPDATE statements to assure that the new row label is read-accessible.
The Access Control Enforcement Options	READ_CONTROL	Applies policy enforcement to all queries. Only authorized rows are accessible for SELECT, UPDATE, and DELETE operations.
	WRITE_CONTROL	Determines the ability to INSERT, UPDATE, and DELETE data in a row. If this option is active, it enforces INSERT_CONTROL, UPDATE_CONTROL, and DELETE_CONTROL.
	INSERT_ CONTROL	Applies policy enforcement to INSERT operations, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.
	DELETE_ CONTROL	Applies policy enforcement to DELETE operations, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.
	UPDATE_ CONTROL	Applies policy enforcement to UPDATE operations on the data columns within a row, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.
The Overriding Enforcement Options	ALL_CONTROL	Applies all enforcement options.
	NO_CONTROL	Applies no enforcement options. A labeling function or a SQL predicate can nonetheless be applied.

Remember that even when Oracle Label Security is applicable to a table, some DML operations may not be covered by the policies being applied. The policy enforcement options set by the administrator determine both the SQL processing behavior and what an authorized user can actually see in response to a query on a protected table. Except where noted, this chapter assumes that ALL_CONTROL is active, meaning that all enforcement options are in effect. If users attempt to perform an operation for which they are not authorized, then an error message is raised and the SQL statement fails.

> **See Also:** "Implementing Inverse Groups with the INVERSE_ GROUP Enforcement Option" on page 15-3

Understanding the relationships among these policy enforcement options, and what SQL statements they control, is essential to their effective use in designing and implementing your Oracle Label Security policies.

Table 9–2, " Policy Enforcement Options" indicates these relationships.

Table 9–3 What Policy Enforcement Options Control

Specifying This Option in a Policy	Controls These SQL Operations	Using These Criteria and with These Effects
READ_CONTROL	SELECT, UPDATE, and DELETE	Only authorized rows (*) are accessible.
WRITE_CONTROL	INSERT, UPDATE, and DELETE	(a) Only authorized rows (**) are accessible (b) Data labels writable unless LABEL_UPDATE is active.
WRITE_CONTROL is necessary for these three:		
INSERT_CONTROL	INSERT	
UPDATE_CONTROL	UPDATE	
DELETE_CONTROL	DELETE	
CHECK_CONTROL		Applies READ_CONTROL policy enforcement to INSERT and UPDATE statements to assure that the new row label is read-accessible.
The Access Control Enforcement Options		Applies policy enforcement to all queries. Only authorized rows are accessible for operations. Determines the ability to data in a row. If this option is active, then it enforces.
	INSERT_CONTROL	Applies policy enforcement to INSERT operations, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.
	DELETE_CONTROL	Applies policy enforcement to DELETE operations, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.
	UPDATE_CONTROL	Applies policy enforcement to UPDATE operations on the data columns within a row, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.
The Overriding Enforcement Options	ALL_CONTROL	Applies all enforcement options.
	NO_CONTROL	Applies no enforcement options. A labeling function or a SQL predicate can nonetheless be applied.

(*) A row is authorized for READ access if the following three criteria are all met:

(user-minimum-level) < = (data-row-level) < = (session-level)

(any-data-group) is a child of (any-user-group-or-childgroup)

(every-data-compartment) is also in (the user's compartments)

Refer to Figure 3–7, "Label Evaluation Process for Read Access" on page 3-9

(**) A row is authorized for READ access if the following three criteria are all met:

(user-minimum-level) < = (data-row-level) < = (session-level)

(any-data-group) is a child of (any-user-group-or-childgroup)

(every-data-compartment) is also in (the user's compartments)

Refer to Figure 3–7, "Label Evaluation Process for Read Access" on page 3-9.

The HIDE Policy Column Option

You can specify the HIDE policy configuration option when you initially apply an Oracle Label Security policy to a table, that is, when adding the policy column to the table. This prevents display of the column containing the policy's labels.

> **See Also:** "Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY" on page 10-3.

Once the policy has been applied, the hidden (or not hidden) status of the column cannot be changed unless the policy is removed with the DROP_COLUMN parameter set to TRUE. Then, the policy can be reapplied with a new hidden status.

> **See Also:** Removing a Policy with SA_POLICY_ADMIN.REMOVE_TABLE_POLICY on page 10-4.

INSERT statements doing all-column inserts do not require the values for hidden label columns.

SELECT statements do not automatically return the values of hidden label columns. Such values must be explicitly retrieved.

A DESCRIBE on a table may or may not display the label column. If the administrator sets the HIDE option, then the label column will not be displayed. If HIDE is not specified for a policy, then the label column is displayed in response to a SELECT.

> **See Also:** "Retrieving All Columns from a Table When the Policy Label Column Is Hidden" on page 5-7

The Label Management Enforcement Options

The three label enforcement options control the data label written when a row is inserted or updated. When a policy specifies these options and is applied to a table or schema, these options apply to the situations described in this section.

A user inserting a row can specify any data label within the range of the user's label authorizations. If the user does not specify a label for the row being written, LABEL_DEFAULT can do so. Updates can be restricted by LABEL_UPDATE. Inserts or updates that use a labeling function can need CHECK-CONTROL to prevent assigning a data label outside the user's authorizations. Such a label would prevent the user from accessing the row just written, and could enable the user to make data available inappropriately.

Any labeling function in force on a table overrides these options. Such a function can be named in the call that applies the policy to the table. If the administrator named such a function when applying a policy, but then disables or removes that policy, then that function is no longer applied.

> **See Also:**
>
> - Chapter 10, "Applying Policies to Tables and Schemas" regarding applying policies to tables or schemas (or removing them).
> - "Disabling a Policy with SA_SYSDBA.DISABLE_POLICY" on page 7-12

LABEL_DEFAULT: Using the Session's Default Row Label

A user can update a row without specifying a label value, because the updated row uses its original label. However, to insert a new row, the user must supply a valid label unless a labeling function is in force or LABEL_DEFAULT applies for the table. LABEL_DEFAULT causes the user's session default row label to be used as the new row label.

If neither LABEL_DEFAULT nor a labeling function is in force and the user attempts to INSERT a row, then an error occurs.

Note that any labeling function in force on a table overrides the LABEL_DEFAULT option.

LABEL_UPDATE: Changing Data Labels

A user updating a row can normally change its label to any label within his authorized label range. However, if LABEL_UPDATE applies, then to modify a label, the user must have one or more of these privileges: WRITEUP, WRITEDOWN, and WRITEACROSS.

The LABEL_UPDATE option uses an Oracle after-row trigger which is called only on an update operation affecting the label. Note that any labeling function in force on a table overrides the LABEL_UPDATE option.

See Also: "Special Row Label Privileges" on page 3-15.

CHECK_CONTROL: Checking Data Labels

If a row being inserted or updated gets its label from a labeling function, then that label could conceivably be outside the user's authorizations, preventing future access by that user.

CHECK_CONTROL causes READ_CONTROL to apply to the new label, ensuring that this user will be authorized to read the inserted or updated row after the operation. If not, then the insert or update operation is canceled and has no effect.

In other words, if CHECK_CONTROL is included as an option in a policy being enforced on a row, then the user modifying that row must still be able to access it after the operation. CHECK_CONTROL prevents a user or a labeling function from modifying a row's label to include a level, group, or compartment that the modifying user would be prevented from accessing.

Note that CHECK_CONTROL overrides any labeling function in force on a table.

The Access Control Enforcement Options

Access control options limit the rows accessible for SELECT, UPDATE, INSERT, or DELETE operations to only those rows whose labels meet established policies:

- READ_CONTROL: Reading Data
- WRITE_CONTROL: Writing Data
- INSERT_CONTROL, UPDATE_CONTROL, and DELETE_CONTROL

READ_CONTROL: Reading Data

READ_CONTROL uses Oracle virtual private database (VPD) technology to enforce the read access mediation algorithm illustrated in Figure 3–7, "Label Evaluation Process for Read Access" on page 3-9.

READ_CONTROL limits the set of records accessible to a session for SELECT, UPDATE and DELETE operations. If READ_CONTROL is not active, then even rows in the table protected by the policy are accessible to all users.

WRITE_CONTROL: Writing Data

WRITE_CONTROL uses Oracle after-row triggers to enforce the write access mediation algorithm illustrated in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11. When an Oracle Label Security policy specifying the WRITE_CONTROL option is applied to a table, triggers are generated and the algorithm is enforced.

Note: The protection implementation for WRITE_CONTROL is the same for all write operations, but you need not apply all write options across the board. You can apply WRITE_CONTROL selectively for INSERT, UPDATE, and DELETE operations by using the corresponding policy enforcement option (INSERT_CONTROL, UPDATE_CONTROL, and DELETE_CONTROL) instead of WRITE_CONTROL.

If WRITE_CONTROL is on but LABEL_UPDATE is not specified, then the user can change both data and labels. If you want to control updating the row labels, then specify the LABEL_UPDATE option in addition to WRITE_CONTROL when creating your policies.

INSERT_CONTROL, UPDATE_CONTROL, and DELETE_CONTROL

These options apply policy enforcement during the corresponding operations on the data columns within a row, according to the algorithm for write access described in Figure 3–8, "Label Evaluation Process for Write Access" on page 3-11.

Specifying WRITE_CONTROL limits all insert, update, and delete operations. However,

- specifying INSERT_CONTROL limits insertions but not updates or deletes;
- specifying UPDATE_CONTROL limits updates but not insertions or deletes; and
- specifying DELETE_CONTROL limits deletes but not insertions or updates.

> **See Also:** For inserts, Inserting Labeled Data Using Policy Options and Labeling Functions on page 9-12;
>
> for updates, Updating Labeled Data Using Policy Options and Labeling Functions on page 9-13;
>
> and for deletions, Deleting Labeled Data Using Policy Options and Labeling Functions on page 9-14.

The Overriding Enforcement Options

Whereas ALL_CONTROL applies all of the label management and access control enforcement options, NO_CONTROL applies none of them. In either case, labeling functions and SQL predicates can be applied. Note that the ALL_CONTROL option can be used only on the command line.

If you apply a policy with NO_CONTROL specified, then a policy label column is added to the table, but the label values are NULL. Because no access controls are

operating on the table, you can proceed to enter labels as desired. You can then set the policy enforcement options as you wish.

NO_CONTROL can be a useful option if you have a labeling function in force to label the data correctly, but want to let all users access all the data.

Guidelines for Using the Policy Enforcement Options

You can customize policy enforcement for a schema or table through the Oracle Enterprise Manager as described in Chapters 4 & 7, or by using the SA_POLICY_ADMIN package as described in Chapter 8.

See Also:

- "Authorized Levels" on page 3-4
- "Oracle Enterprise Manager" on page 7-10
- Chapter 10, "Applying Policies to Tables and Schemas"

This section documents the supported keywords.

Note that when you create a policy, you can specify a string of default options to be used whenever the policy is applied without schema or table options being specified.

If a policy is first applied to a table, and then also applied to the schema containing that table, then the options on the table are not affected by the schema policy. The options of the policy originally applied to the table remain in force.

In general, administrators use the LABEL_DEFAULT policy option, causing data written by a user to be labeled with that user's row label. Alternatively, a labeling function can be used to label the data. If neither of these two choices is used, then a label must be specified in every INSERT statement. (Updates retain the row's original label.)

The following table suggests that certain combinations of policy enforcement options are useful when implementing an Oracle Label Security policy. As the table indicates, you might typically enforce READ_CONTROL and WRITE_CONTROL, choosing from among several possible combinations for setting the data label on writes.

Table 9–4 Suggested Policy Enforcement Option Combinations

Options	Access Enforcement
READ_CONTROL, WRITE_CONTROL, LABEL_DEFAULT	Read and write access based on session label. Default label provided; users can insert/update both data and labels.
READ_CONTROL, WRITE_CONTROL, Labeling Function	Read and write access based on session label. Users can set/change only row data; all row labels are set explicitly by the labeling function. Add CHECK_CONTROL to restrict new labels (on insert or update) to visible range of labels.
READ_CONTROL, WRITE_CONTROL, LABEL_UPDATE	Read and write access based on session label. Changing but users cannot change labels without privileges. Add CHECK_CONTROL to restrict new labels (on insert or update) to visible range.

Exemptions from Oracle Label Security Policy Enforcement

1. Oracle Label Security is not enforced during DIRECT path export.

   See Also: "Using the Export Utility with Oracle Label Security" on page 14-1

2. By design, Oracle Label Security policies cannot be applied to objects in schema SYS. As a consequence, the SYS user, and users making a DBA-privileged connection to the database (such as `CONNECT AS SYSDBA`) do not have Oracle Label Security policies applied to their actions. DBAs need to be able to administer the database. It would make no sense, for example, to export part of a table due to an Oracle Label Security policy being applied. The database user SYS is thus always exempt from Oracle Label Security enforcement, regardless of the export mode, application, or utility used to extract data from the database.

   See Also: For other DBA-related considerations, see Chapter 14, "Performing DBA Functions Under Oracle Label Security".

3. Similarly, database users granted the EXEMPT ACCESS POLICY privilege, either directly or through a database role, are exempted from some Oracle Label Security policy enforcement controls such as READ_CONTROL and CHECK_CONTROL, regardless of the export mode, application or utility used to access the database or update its data. Refer to Table 9–2, " Policy Enforcement Options" on page 9-3. The following policy enforcement options remain in effect even when EXEMPT ACCESS POLICY is granted:

   - INSERT_CONTROL, UPDATE_CONTROL, DELETE_CONTROL, WRITE_CONTROL, LABEL_UPDATE, and LABEL_DEFAULT.
   - If the Oracle Label Security policy specifies the ALL_CONTROL option, then all enforcement controls are applied except READ_CONTROL and CHECK_CONTROL.

   EXEMPT ACCESS POLICY is a very powerful privilege and should be carefully managed.

   Note that this privilege does not affect the enforcement of standard *Oracle Database* object privileges such as SELECT, INSERT, UPDATE, and DELETE. These privileges are enforced even if a user has been granted the EXEMPT ACCESS POLICY privilege.

Viewing Policy Options on Tables and Schemas

Use the following views to show the policy enforcement options currently applied to tables and schemas:

- DBA_SA_TABLE_POLICIES
- DBA_SA_SCHEMA_POLICIES

Using a Labeling Function

Application developers can create labeling functions. These programs can compute and return a label using a wide array of resources, including context variables (such as date or username) and data values.

The following sections describe how to use labeling functions.

Labeling Data Rows under Oracle Label Security

There are three ways to label data that is being inserted or updated:

- Explicitly specify a label in every INSERT or UPDATE to the table.
- Set the LABEL_DEFAULT option, which causes the session's row label to be used if an explicit row label is not included in the INSERT or UPDATE statement.
- Create a labeling function, automatically calls on every INSERT or UPDATE statement and independently of any user's authorization.

The recommended approach is to write a labeling function to implement your rules for labeling data. If you specify a labeling function, then Oracle Label Security embeds a call to that function in INSERT and UPDATE triggers to compute a label.

For example, you could create a labeling function named `my_label` to use the contents of COL1 and COL2 of the new row to compute and return the appropriate label for the row. Then, you could insert, into your INSERT or UPDATE statements, the following reference:

```
my_label(:new.col1,:new.col2) J
```

If you do not specify a labeling function, then specify the LABEL_DEFAULT option. Otherwise, you must explicitly specify a label on every INSERT or UPDATE statement.

Understanding Labeling Functions in Oracle Label Security Policies

Labeling functions enable you to consider, in your rules for assigning labels, information drawn from the application context. For example, you can use as a labeling consideration the IP address to which the user is attached. There are many opportunities to use SYS_CONTEXT in this way.

Note: If the SQL statement is invalid, then an error will occur when you apply the labeling function to the table or policy. You should thoroughly test a labeling function before using it with tables.

Labeling functions override the LABEL_DEFAULT and LABEL_UPDATE options.

A labeling function is called in the context of a before-row trigger. This enables you to pass in the old and new values of the data record, as well as the old and new labels.

You can construct a labeling function to permit an explicit label to be passed in by the user.

All labeling functions must have return types of the LBACSYS.LBAC_LABEL data type. The TO_LBAC_DATA_LABEL function can be used to convert a label in character string format to a data type of LBACSYS.LBAC_LABEL. Note that LBACSYS must have the EXECUTE privilege on your labeling function. The owner of the

labeling function must have the EXECUTE privilege on the TO_LBAC_DATA_LABEL function, with the GRANT option.

> **Note:** LBACSYS is a unique schema providing opaque types for Oracle Label Security. Refer to the discussions in Chapter 14, "Performing DBA Functions Under Oracle Label Security".

Creating a Labeling Function for a Policy

The following example shows how to create a labeling function.

```
SQL> CREATE OR REPLACE FUNCTION sa_demo.gen_emp_label
          (Job varchar2,
           Deptno number,
           Total_sal number)
      Return LBACSYS.LBAC_LABEL
    as
      i_label varchar2(80);
    Begin
      /************* Determine Class Level *************/
      if total_sal > 2000 then
      i_label := 'L3:';
      elsif total_sal > 1000 then
      i_label := 'L2:';
      else
      i_label := 'L1:';
      end if;

      /************* Determine Compartment *************/
      IF Job in ('MANAGER','PRESIDENT') then
      i_label := i_label||'M:';
      else
      i_label := i_label||'E:';
      end if;
      /************* Determine Groups *************/
      i_label := i_label||'D'||to_char(deptno);
      return TO_LBAC_DATA_LABEL('human_resources',i_label);
    End;
    /
```

> **Note:** When Oracle Label Security is configured to work directly with Oracle Internet Directory, dynamic label generation is disabled, because labels are managed centrally in Oracle Internet Directory, using olsadmintool commands. Refer to Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory". So, if the label function generates a data label using a string value that is not already established in Oracle Internet Directory, then an error message results.

Specifying a Labeling Function in a Policy

The following example uses the sa_demo.gen_emp_label label from the example in the previous section to show how to specify a labeling function.

```
sa_policy_admin.remove_table_policy('human_resources','sa_demo','emp');
sa_policy_admin.apply_table_policy (
POLICY_NAME => 'human_resources',
```

```
SCHEMA_NAME => 'sa_demo',
TABLE_NAME  => 'emp',
TABLE_OPTIONS => 'READ_CONTROL,WRITE_CONTROL,CHECK_CONTROL',
LABEL_FUNCTION => 'sa_demo.gen_emp_label(:new.job,:new.deptno,:new.sal)',
PREDICATE => NULL);
```

Inserting Labeled Data Using Policy Options and Labeling Functions

This section explains how enforcement options and labeling functions affect the insertion of labeled data.

- Evaluating Enforcement Control Options and INSERT
- Inserting Labels When a Labeling Function Is Specified
- Inserting Child Rows into Tables with Declarative Referential Integrity Enabled

Evaluating Enforcement Control Options and INSERT

When you attempt to insert or update data based on your authorizations, the outcome depends upon what policy enforcement controls are active.

- If INSERT_CONTROL is active, then rows you insert can only have labels within your write authorizations. If you attempt to update data that you can read, but for which you do not have write authorization, an error is raised. For example, if you can read compartments A and B, but you can only write to compartment A, then if you attempt to insert data with compartment B, then the statement will fail.
- If INSERT_CONTROL is *not* active, then you can use any valid label on rows you insert.
- If the CHECK_CONTROL option is active, then rows you insert can only have labels you are authorized to read, even if the labels are generated by a labeling function.

Inserting Labels When a Labeling Function Is Specified

A labeling function takes precedence over labels entered by the user. If the administrator has set up an automatic labeling function, then no data label a user enters will have effect (unless the labeling function itself makes use of the user's proposed label). New row labels are always determined by an active labeling function, if present.

Note that a labeling function can set the label of a row being inserted to a value outside the range that the user writing that row can see. If such a function is in use, then the user can potentially insert a row but not be authorized to see that row. You can prevent this situation by specifying the CHECK_CONTROL option in the policy. If this option is active, then the new data label is checked against the user's read authorization, and if the user cannot read it, then the insert operation is not performed.

Inserting Child Rows into Tables with Declarative Referential Integrity Enabled

If a parent table is protected by declarative referential integrity, then inserting a child row is constrained by the requirement that the parent row be visible. The user must be able to see the parent row for the insert operation to succeed, that is, the user must have read access to the parent row.

If READ_CONTROL is active on the parent table, then the user's read authorization must be sufficient to authorize a SELECT operation on the parent row. For example, a user who cannot read department 20 cannot insert child rows for department 20. Note that all records will be visible if the user has FULL or READ privileges on the table or schema.

Updating Labeled Data Using Policy Options and Labeling Functions

The rules for updates in Oracle Label Security are almost identical to those for inserts, as long as the user is authorized to change the rows in question. This section contains these topics:

Updating Labels Using CHAR_TO_LABEL

If you need to change a row's label from SENSITIVE to CONFIDENTIAL, then you can change the label by using the CHAR_TO_LABEL FUNCTION as follows:

```
UPDATE emp
SET hr_label = char_to_label ('HR', 'CONFIDENTIAL')
WHERE ename = 'ESTANTON';
```

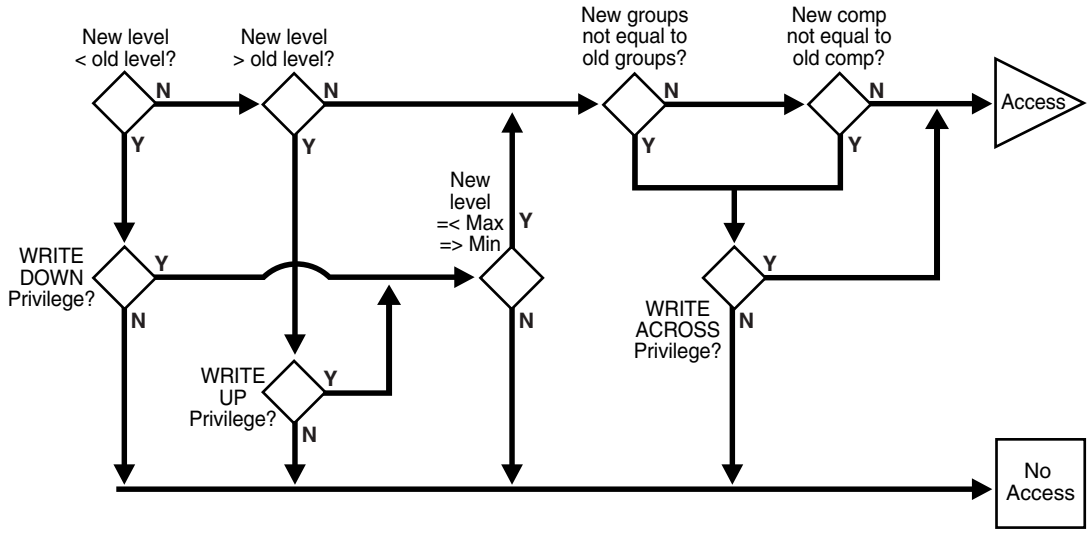
Evaluating Enforcement Control Options and UPDATE

When you attempt to update data based on your authorizations, the outcome depends on what enforcement controls are active.

- If UPDATE_CONTROL is active, then you can only update rows whose labels fall within your write authorizations. If you attempt to update data that you can read, but for which you do not have write authorization, then an error is raised. Assume, for example, that you can read compartments A and B, but you can only write to compartment A. In this case, if you attempt to update data with compartment B, then the statement will fail.
- If UPDATE_CONTROL is not active, then you can update all rows to which you have read access.
- If LABEL_UPDATE is active, then you must have the appropriate privilege (WRITEUP, WRITEDOWN, or WRITEACROSS) to change a label by raising or lowering its sensitivity level, or altering its groups or compartments.
- If LABEL_UPDATE is *not* active but UPDATE_CONTROL *is* active, then you can update a label to any new label value within your write authorization.
- If CHECK_CONTROL is active, then you can only write labels you are authorized to read.

The following figure illustrates the label evaluation process for LABEL_UPDATE.

Figure 9–1 Label Evaluation Process for LABEL_UPDATE



Updating Labels When a Labeling Function Is Specified

A labeling function takes precedence over labels entered by the user. If the administrator has set up an automatic labeling function, then no label a user enters will have effect (unless the labeling function itself makes use of the user's proposed label). New row labels are always determined by an active labeling function, if present.

Note that the security administrator can establish a labeling function that sets the label of a row being updated to a value outside the range that you can see. If this is the case, then you can update a row, but not be authorized to see the row. If the CHECK_CONTROL option is on, then you will not be able to perform such an update. The CHECK_CONTROL option verifies your read authorization on the new label.

Updating Child Rows in Tables with Declarative Referential Integrity Enabled

If a child row is in a table that has a referential integrity constraint, then the update can succeed only if the parent row is visible that is the user must be able to see the parent row. If the parent table has READ_CONTROL on, then the user's read authorization must be sufficient to authorize a SELECT on the parent row.

For example, a user who cannot read department 20 in a parent table cannot update an employee's department to department 20 in a child table. (If the user has FULL or READ privilege, then all records will be visible.)

See Also: *Oracle Database Advanced Application Developer's Guide*

Deleting Labeled Data Using Policy Options and Labeling Functions

This section covers the deletion of labeled data.

- If DELETE_CONTROL is active, then you can delete only rows within your write authorization.
- If DELETE_CONTROL is *not* active, then you can delete only rows that you can read.

- With DELETE_CONTROL active, and declarative referential integrity defined with cascading deletes, you must have write authorization on *all* the rows to be deleted, or the statement will fail.

You cannot delete a parent row if there are any child rows attached to it, regardless of your write authorization. To delete such a parent row, you must first delete each of the child rows. If DELETE_CONTROL is active on any of the child rows, then you must have write authorization to delete the child rows.

Consider, for example, a situation in which the user is UNCLASSIFIED and there are three rows as follows:

Row	Table	Sensitivity
Parent row:	DEPT	UNCLASSIFIED
Child row:	EMP	UNCLASSIFIED
Child row:	EMP	UNCLASSIFIED

In this case, the UNCLASSIFIED user cannot delete the parent row.

DELETE_CONTROL has no effect when DELETE_RESTRICT is active. DELETE_RESTRICT is always enforced. In some cases (depending on the user's authorizations and the data's labels) it may look as though a row has no child rows, when it actually does have children but the user cannot see them. Even if a user cannot see child rows, he still cannot delete the parent row.

See Also: *Oracle Database Advanced Application Developer's Guide*

Using a SQL Predicate with an Oracle Label Security Policy

You can use a SQL predicate to provide extensibility for selective enforcement of data access rules.

This section contains these topics:

- Modifying an Oracle Label Security Policy with a SQL Predicate
- Affecting Oracle Label Security Policies with Multiple SQL Predicates

Modifying an Oracle Label Security Policy with a SQL Predicate

A SQL predicate is a condition, optionally preceded by AND or OR. It can be appended for READ_CONTROL access mediation. The following predicate, for example, adds an application-specific test based on COL1 to determine if the session has access to the row.

```
AND my_function(col1)=1
```

The combined result of the policy and the user-specified predicate limits the rows that a user can read. So, this combination affects the labels and data that CHECK_CONTROL will permit a user to change. An OR clause, for example, increases the number of rows a user can read.

A SQL predicate can be useful if you want to avoid performing label-based filtering. In certain situations, a SQL predicate can easily implement row-level security on tables. Used instead of READ_CONTROL, a SQL predicate will filter the data for SELECT, UPDATE, and DELETE operations.

Similarly, in a typical, Web-enabled human resources application, a user might have to be a manager to access rows in the employee table. In such cases, the user's user label would have to dominate the label on the employee's row. A SQL predicate like the following could be added, so that an employee could bypass label-based filtering if he wanted to view his own record in the employee table. (An `OR` is used so that *either* the label policy will apply, *or* this statement will apply.)

```
OR SYS_CONTEXT ('USERENV', 'SESSION_USER') = employee_name
```

This predicate enables you to have additional access controls so that each employee can access his or her own record.

You can use such a predicate in conjunction with READ_CONTROLs or as a standalone predicate even if READ_CONTROL is not implemented.

Note: Verify that the predicate accomplishes your security goals before you implement it in an application.

If a syntax error occurs in a predicate under Oracle Label Security, then an error will *not* arise when you try to apply the policy to a table. Rather, a predicate error message will arise when you first attempt to reference the table.

Affecting Oracle Label Security Policies with Multiple SQL Predicates

A predicate applied to a table by means of an Oracle Label Security policy is appended to any other predicates that may be applied by other Oracle Label Security policies, or by Oracle fine grain access control/VPD policies. The predicates are ANDed together.

Consider the following predicates applied to the EMP table in the SCOTT schema:

- A predicate generated by an Oracle VPD policy, such as `deptno=10`
- A label-based predicate generated by an Oracle Label Security policy, such as `label=100`, with a user-specified predicate such as

  ```
  OR SYS_CONTEXT ('USERENV', 'SESSION_USER') = ename
  ```

Correct: These predicates would be ANDed together as follows:

```
WHERE deptno=10 AND (label=100 OR SYS_CONTEXT ('USERENV', 'SESSION_USER') = ename)
```

Incorrect: The predicates would *not* be combined in the following way:

```
WHERE deptno=10 AND label=100 OR SYS_CONTEXT ('USERENV', 'SESSION_USER') = ename
```

10

Applying Policies to Tables and Schemas

This chapter describes the SA_POLICY_ADMIN package, which enables you to administer policies on tables and schemas. It contains these sections:

- Policy Administration Terminology
- Subscribing Policies in Directory-Enabled Label Security
- Policy Administration Functions for Tables and Schemas
- Administering Policies on Tables Using SA_POLICY_ADMIN
- Administering Policies on Schemas with SA_POLICY_ADMIN

Policy Administration Terminology

When you *apply* a policy to a table, the policy is automatically enabled. To *disable* a policy is to turn off its protections, although it is still applied to the table. To *enable* a policy is to turn on and enforce its protections for a particular table or schema.

To *remove* a policy is to take it entirely away from the table or schema. Note, however, that the policy label column and the labels remain in the table unless you explicitly drop them.

You can *alter* the default policy enforcement options for future tables that may be created in a schema. This does not, however, affect policy enforcement options on existing tables in the schema.

To change the enforcement options on an existing table, you must first *remove* the policy from the table, make the desired changes, and then *reapply* the policy to the table.

See Also: "Choosing Policy Options" on page 9-1

Subscribing Policies in Directory-Enabled Label Security

In an Oracle Internet Directory-enabled Oracle Label Security, a policy must be subscribed before it can be applied (by APPLY_TABLE_POLICY or APPLY_SCHEMA_POLICY). In a standalone Oracle Label Security installation, the latter functions can be used directly without the need to subscribe.

You subscribe a policy by using SA_POLICY_ADMIN.POLICY_SUBSCRIBE, as described in the next section.

Such a policy cannot be dropped unless it has been removed from any table or schema to which it was applied, and then has been unsubscribed.

You unsubscribe a policy by using SA_POLICY_ADMIN.POLICY_UNSUBSCRIBE as described in a subsequent section.

Subscribing to a Policy with SA_POLICY_ADMIN.POLICY_SUBSCRIBE

In an Oracle Internet Directory-enabled Oracle Label Security configuration, use the POLICY_SUBSCRIBE procedure to subscribe to the policy for usage in APPLY_TABLE_POLICY and APPLY_SCHEMA_POLICY. This procedure must be called for a policy before that policy can be applied to a table or schema. Subscribing is needed only once, not for each use of the policy in a table or schema.

Syntax

```
PROCEDURE POLICY_SUBSCRIBE(
  policy_name     IN VARCHAR2);
```

where policy_name specifies an existing policy.

> **Note::** This procedure needs to be used before policy usage only in the case of Oracle Internet Directory-enabled Oracle Label Security configuration. In the standalone Oracle Label Security case, the policy can be used in APPLY_TABLE_POLICY and APPLY_SCHEMA_POLICY directly without the need to subscribe.

Example: The following statement subscribes the database to the HUMAN_RESOURCES policy so that it can used by applying on tables and schema.

```
SA_POLICY_ADMIN.POLICY_SUBSCRIBE('human_resources');
```

Unsubscribing to a Policy with SA_POLICY_ADMIN.POLICY_UNSUBSCRIBE

In an Oracle Internet Directory enabled Oracle Label Security configuration, use the POLICY_UNSUBSCRIBE procedure to unsubscribe to the policy. This procedure can be used only if the policy is not in use, that is, it has not been applied to any table or schema. (If it has been applied to tables or schemas, then it must be removed from all of them before it can be unsubscribed.) A policy can be dropped in Oracle Internet Directory (olsadmintool droppolicy in Appendix B) only if is not subscribed in any of the databases that have registered with that Oracle Internet Directory.

Syntax

```
PROCEDURE POLICY_UNSUBSCRIBE(
  policy_name  IN VARCHAR2);
```

where policy_name specifies an existing policy.

Example: The following statement unsubscribes the database to the HUMAN_RESOURCES policy.

```
SA_POLICY_ADMIN.POLICY_UNSUBSCRIBE('human_resources');
```

Policy Administration Functions for Tables and Schemas

Two sets of functions are available to administer Oracle Label Security policies:

- functions to administer policies at the table level

- functions to administer policies at the schema level

Schema-level functions are provided for convenience. Note, however, that administrative operations that you perform at the table level will override operations performed at the schema level.

Table 10–1 Policy Administration Functions

Purpose	Table-Level Function	Schema-Level Function
Apply policy	APPLY_TABLE_POLICY	APPLY_SCHEMA_POLICY
Alter policy	Not applicable	ALTER_SCHEMA_POLICY
Disable policy	DISABLE_TABLE_POLICY	DISABLE_SCHEMA_POLICY
Reenable policy	ENABLE_TABLE_POLICY	ENABLE_SCHEMA_POLICY
Remove policy	REMOVE_TABLE_POLICY	REMOVE_SCHEMA_POLICY

Administering Policies on Tables Using SA_POLICY_ADMIN

To administer policies on tables, a user must have the EXECUTE privilege for the SA_POLICY_ADMIN package, and must have been granted the *policy*_DBA role. This section contains these topics:

- Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY
- Removing a Policy with SA_POLICY_ADMIN.REMOVE_TABLE_POLICY
- Disabling a Policy with SA_POLICY_ADMIN.DISABLE_TABLE_POLICY
- Reenabling a Policy with SA_POLICY_ADMIN.ENABLE_TABLE_POLICY

Applying a Policy with SA_POLICY_ADMIN.APPLY_TABLE_POLICY

Use the APPLY_TABLE_POLICY procedure to add the specified policy to a table. A policy label column is added to the table if it does not exist, and is set to NULL. When a policy is applied, it is automatically enabled. To change the table options, labeling function, or predicate, you must first remove the policy, and then reapply it.

Syntax

```
PROCEDURE APPLY_TABLE_POLICY (
  policy_name       IN VARCHAR2,
  schema_name       IN VARCHAR2,
  table_name        IN VARCHAR2,
  table_options     IN VARCHAR2 DEFAULT NULL,
  label_function    IN VARCHAR2 DEFAULT NULL,
  predicate         IN VARCHAR2 DEFAULT NULL);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
table_name	The table to be controlled by the policy
table_options	A comma-delimited list of policy enforcement options to be used for the table. If NULL, then the policy's default options are used.

Parameter	Specifies
label_function	A string calling a function to return a label value to use as the default. For example, `my_label(:new.dept,:new.status)` computes the label based on the new values of the DEPT and STATUS columns in the row.
predicate	An additional predicate to combine (using AND or OR) with the label-based predicate for READ_CONTROL

Example: The following statement applies the HUMAN_RESOURCES policy to the EMP table in the SA_DEMO schema.

```
SA_POLICY_ADMIN.APPLY_TABLE_POLICY('human_resources',
'sa_demo','emp','no_control');
```

Removing a Policy with SA_POLICY_ADMIN.REMOVE_TABLE_POLICY

The REMOVE_TABLE_POLICY procedure removes the specified policy from a table. The policy predicate and any DML triggers will be removed from the table, and the policy label column can optionally be dropped. Policies can be removed from tables belonging to a schema that is protected by the policy.

Syntax

```
PROCEDURE REMOVE_TABLE_POLICY (
policy_name        IN VARCHAR2,
schema_name        IN VARCHAR2,
table_name         IN VARCHAR2,
  drop_column      IN BOOLEAN DEFAULT FALSE);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
table_name	The table
drop_column	Whether the column is to be dropped: if TRUE, then the policy's column will be dropped from the table, otherwise, it will remain

Example: The following statement removes the HUMAN_RESOURCES policy from the EMP table in the SA_DEMO schema:

```
SA_POLICY_ADMIN.REMOVE_TABLE_POLICY('human_resources','sa_demo','emp');
```

Disabling a Policy with SA_POLICY_ADMIN.DISABLE_TABLE_POLICY

The DISABLE_TABLE_POLICY procedure disables the enforcement of the policy for the specified table without changing the enforcement options, labeling function, or predicate values. It removes the RLS predicate and DML triggers from the table.

Syntax

```
PROCEDURE DISABLE_TABLE_POLICY (
  policy_name      IN VARCHAR2,
  schema_name      IN VARCHAR2,
  table_name       IN VARCHAR2);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
table_name	The table

Example: The following statement disables the HUMAN_RESOURCES policy on the EMP table in the SA_DEMO schema:

```
SA_POLICY_ADMIN.DISABLE_TABLE_POLICY('human_resources','sa_demo','emp');
```

Reenabling a Policy with SA_POLICY_ADMIN.ENABLE_TABLE_POLICY

The ENABLE_TABLE_POLICY procedure reenables the current enforcement options, labeling function, and predicate for the specified table by reapplying the RLS predicate and DML triggers.

Syntax

```
PROCEDURE ENABLE_TABLE_POLICY (
  policy_name     IN VARCHAR2,
  schema_name     IN VARCHAR2,
  table_name      IN VARCHAR2);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
table_name	The table

Example: The following statement reenables the HUMAN_RESOURCES policy on the EMP table in the SA_DEMO schema:

```
SA_POLICY_ADMIN.ENABLE_TABLE_POLICY('human_resources','sa_demo','emp');
```

Administering Policies on Schemas with SA_POLICY_ADMIN

To administer policies on schemas, a user must have the EXECUTE privilege on the SA_POLICY_ADMIN package, and must have been granted the *policy*_DBA role.

This section contains these topics:

- Applying a Policy with SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY
- Altering Enforcement Options: SA_POLICY_ADMIN.ALTER_SCHEMA_POLICY
- Removing a Policy with SA_POLICY_ADMIN.REMOVE_SCHEMA_POLICY
- Disabling a Policy with SA_POLICY_ADMIN.DISABLE_SCHEMA_POLICY
- Reenabling a Policy with SA_POLICY_ADMIN.ENABLE_SCHEMA_POLICY
- Policy Issues for Schemas

Applying a Policy with SA_POLICY_ADMIN.APPLY_SCHEMA_POLICY

In addition to applying a policy to individual tables, you can apply a policy at the schema level. The APPLY_SCHEMA_POLICY procedure applies the specified policy

to all of the existing tables in a schema (that is, to those which do not already have the policy applied) and enables the policy for these tables. Then, whenever a new table is created in the schema, the policy is automatically applied to that table, using the schema's default options. No changes are made to existing tables in the schema that already have the policy applied.

Syntax

```
PROCEDURE APPLY_SCHEMA_POLICY (
  policy_name        IN VARCHAR2,
  schema_name        IN VARCHAR2,
  default_options    IN VARCHAR2 DEFAULT NULL);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
default_options	The default options to be used for tables in the schema

If the *default_options* parameter is NULL, then the policy's default options will be used to apply the policy to the tables in the schema.

Altering Enforcement Options: SA_POLICY_ADMIN.ALTER_SCHEMA_POLICY

The ALTER_SCHEMA_POLICY procedure changes the default enforcement options for the policy. Any new tables created in the schema will automatically have the new enforcement options applied. The existing tables in the schema are not affected.

Syntax

```
PROCEDURE ALTER_SCHEMA_POLICY (
  policy_name         IN VARCHAR2,
  schema_name         IN VARCHAR2,
  default_options     IN VARCHAR2);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
default_options	The default options to be used for new tables in the schema

To change enforcement options on a table (rather than a schema), you must first drop the policy from the table, make the change, and then reapply the policy.

If you alter the enforcement options on a schema, then this will take effect the next time a table is created in the schema. As a result, different tables within a schema may have different policy enforcement options in force.

Removing a Policy with SA_POLICY_ADMIN.REMOVE_SCHEMA_POLICY

The REMOVE_SCHEMA_POLICY procedure removes the specified policy from a schema. The policy will be removed from all the tables in the schema and, optionally, the label column for the policy will be dropped from all the tables.

Syntax

```
PROCEDURE REMOVE_SCHEMA_POLICY (
  policy_name     IN VARCHAR2,
  schema_name     IN VARCHAR2,
  drop_column     IN BOOLEAN DEFAULT FALSE);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table
drop_column	If TRUE, then the policy's column will be dropped from the tables, otherwise, the column will remain.

Disabling a Policy with SA_POLICY_ADMIN.DISABLE_SCHEMA_POLICY

The DISABLE_SCHEMA_POLICY procedure disables the enforcement of the policy for all of the tables in the specified schema, without changing the enforcement options, labeling function, or predicate values. It removes the RLS predicate and DML triggers from all the tables in the schema.

Syntax

```
PROCEDURE DISABLE_SCHEMA_POLICY (
  policy_name    IN VARCHAR2,
  schema_name    IN VARCHAR2);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table

Reenabling a Policy with SA_POLICY_ADMIN.ENABLE_SCHEMA_POLICY

The ENABLE_SCHEMA_POLICY procedure reenables the current enforcement options, labeling function, and predicate for the tables in the specified schema by re-applying the RLS predicate and DML triggers.

Syntax

```
PROCEDURE ENABLE_TABLE_POLICY (
  policy_name    IN VARCHAR2,
  schema_name    IN VARCHAR2);
```

Parameter	Specifies
policy_name	An existing policy
schema_name	The schema that contains the table

The result is like enabling a policy for a table, but it covers all the tables in the schema.

Policy Issues for Schemas

Note the following aspects of using Oracle Label Security policies with schemas:

- If you apply a policy to an empty schema, then every time you create a table within that schema, the policy is applied. Once the policy is applied to the schema, the default options you choose are applied to every table added.
- If you remove the policy from a table so that it is unprotected, and then run SA_POLICY_ADMIN.ENABLE_SCHEMA_POLICY, then the table will remain unprotected. If you wish to protect the table once again, then you must apply the policy to the table, or re-apply the policy to the schema.

If you apply a policy to a schema that already contains tables protected by the policy, then all future tables will have the new options that were specified when you applied the policy. The existing tables will retain the options they already had.

11

Administering and Using Trusted Stored Program Units

This chapter explains how to use trusted stored program units to enhance system security. It contains these topics:

Introduction to Trusted Stored Program Units

Oracle Database 11g Release 1 (11.1) *stored procedures*, *functions*, and *packages* are sets of PL/SQL statements stored in a database in compiled form. The single difference between functions and procedures is that functions return a value and procedures do not. Trusted stored program units are like any other stored program units in *Oracle Database*: the underlying logic is the same.

A *package* is a set of procedures and functions, together with the cursors and variables they use, stored as a unit.

There are two parts to a package, the package specification and the package body. The package specification declares the external definition of the public procedures, functions, and variables that the package contains. The package body contains the actual text of the procedures and functions, as well as any private procedures and variables.

A *trusted stored program unit* is a stored procedure, function, or package that has been granted one or more Oracle Label Security privileges. Trusted stored program units are typically used to let users perform privileged operations in a controlled manner, or update data at several labels. This is the optimal approach to permit users to access data beyond their authorization.

Trusted stored program units provide fine-grained control over the use of privileges. Although you can potentially grant privileges to many users, the granting of privileges should be done with great discretion because it might violate the security policy established for your application. Rather than assigning privileges to users, you can identify any application operations requiring privileges, and implement them as trusted program units. When you grant privileges to these stored program units, you effectively restrict the Oracle Label Security privileges required by users. This approach employs the principle of *least privilege*.

For example, if a user with the label CONFIDENTIAL needs to insert data into SENSITIVE rows, then you can grant the WRITEUP privilege to a trusted stored program to which the user has access. In this way, the user can perform the task by means of the trusted stored program, while staying at the CONFIDENTIAL level.

The trusted program unit performs all the actions on behalf of the user. You can thus effectively encapsulate the security policy into a module that can be verified to make sure that it is valid.

How a Trusted Stored Program Unit Runs

A trusted stored program unit runs using its own privileges, and the caller's labels. In this way, it can perform privileged operations on the set of rows constrained by the user's labels.

Oracle Database system and object privileges are intended to be bundled into roles. Users are then granted roles as necessary. By contrast, Oracle Label Security privileges can only be assigned to users or to stored program units. These trusted stored program units provide a more manageable mechanism than roles to control the use of Oracle Label Security privileges.

Trusted Stored Program Unit Example

A trusted stored program unit with the READ privilege can read all unprotected data and all data protected by this policy in the database. Consider, for example, a user who is responsible for creating purchasing forecast reports. The user must perform a summation operation on the amount of all purchases. Regardless of whether or not user's own labels authorize access to the individual purchase orders. The syntax for creating the summation procedure in this example is as follows:

```
CREATE FUNCTION sum_purchases RETURN NUMBER IS
  psum NUMBER;
BEGIN
  SELECT SUM(amount) INTO psum
  FROM purchase_orders;
RETURN psum;
END sum_purchases;
```

In this way, the program unit can gather information the end user is not able to gather, and can make it available by means of a summation.

Note that to run SUM_PURCHASES, the user would need to be granted the standard *Oracle Database* EXECUTE object privilege upon this procedure.

> **See Also:** Chapter 3, "Understanding Access Controls and Privileges"

Managing Program Unit Privileges with SET_PROG_PRIVS

To grant privileges to a stored program unit, you must have the *policy*_DBA role, and the EXECUTE permission on the SA_USER_ADMIN package. You can use either the SA_USER_ADMIN package or Oracle Enterprise Manager to manage Oracle Label Security privileges.

Use the SA_USER_ADMIN.SET_PROG_PRIVS procedure to set policy-specific privileges for program units. If the *privileges* parameter is NULL, then the program unit's privileges for the policy are removed.

Syntax:

```
PROCEDURE SET_PROG_PRIVS (
  policy_name           IN VARCHAR2,
  schema_name           IN VARCHAR2,
  program_unit_name     IN VARCHAR2,
  privileges            IN VARCHAR2);
```

Parameter	Specifies
policy_name	The policy name of an existing policy
program_unit_name	Specifies the program unit to be granted privileges
privileges	A comma-delimited character string of policy-specific privileges

For example, to give the READ privilege to the SUM_PURCHASES function (described in "Trusted Stored Program Unit Example" on page 11-2), you could enter:

```
EXECUTE sa_user_admin.set_prog_privs (
'HR','myschema','sum_purchases','READ');
```

When the SUM_PURCHASES procedure is then called, it runs with the READ privilege as well as the current user's Oracle Label Security privileges. Using this technique, the user can be allowed to find the value of the total corporate payroll, without learning what salary any individual employee receives.

Warning: When you create a trusted stored program unit, have the Oracle Label Security administrator review it carefully and evaluate the privileges you are granting to it. Ensure, for example, that procedures in trusted packages do not perform privileged database operations and then write result or status information into a public variable of the package. In this way, you can make sure that no violations of your site's Oracle Label Security policy can occur.

Creating and Compiling Trusted Stored Program Units

This section contains these topics:

- Creating Trusted Stored Program Units
- Setting Privileges for Trusted Stored Program Units
- Recompiling Trusted Stored Program Units
- Re-creating Trusted Stored Program Units
- Running Trusted Stored Program Units

Creating Trusted Stored Program Units

You create a trusted stored program unit in the same way that you create a standard procedure, function, or package, that is by using the CREATE PROCEDURE, CREATE FUNCTION, or CREATE PACKAGE and CREATE PACKAGE BODY statements. The program unit becomes trusted when you grant it Oracle Label Security privileges.

See Also: *Oracle Database SQL Language Quick Reference*

Setting Privileges for Trusted Stored Program Units

When a developer creates a stored program unit, the Oracle Label Security administrator can verify the correctness of the code before granting the necessary privileges to the stored program unit. Whenever the trusted stored program unit is re-created or replaced, its privileges are removed. The Oracle Label Security administrator must then verify the code again and grant the privileges once again.

Recompiling Trusted Stored Program Units

Recompiling a trusted stored program unit, either automatically or manually (using ALTER PROCEDURE), does not affect its Oracle Label Security privileges. You must, however, grant the EXECUTE privilege on the program unit again after recompiling.

Re-creating Trusted Stored Program Units

Oracle Label Security privileges are revoked if you perform a CREATE OR REPLACE operation on a trusted stored program unit. This limits the potential for misuse of a procedure's Oracle Label Security privileges. Note that the procedure, function, or package can still run even if the Oracle Label Security privileges have been removed.

If you re-create a procedure, function, or package, then you should carefully review its text. When you are certain that the re-created program unit does not violate your site's Oracle Label Security policy, you can then grant it the required privileges again.

In a development environment where trusted stored program units must frequently be replaced (for example, during the first few months of a live system), it is advisable to create a script that can grant the proper Oracle Label Security privileges, as required.

Running Trusted Stored Program Units

Under Oracle Label Security all the standard *Oracle Database* controls on procedure call (regarding access to tables and schemas) are still in force. Oracle Label Security complements these security mechanisms by controlling access to rows. When a trusted stored program unit is carried out, the policy privileges in force are a union of the invoking user's privileges and the program unit's privileges. Whether a trusted stored program unit calls another trusted program unit or a non-trusted program unit, the program unit called runs with the same privileges as the original program unit.

If a sequence of non-trusted and trusted stored program units is carried out, the first trusted program unit will determine the privileges of the entire calling sequence from that point on. Consider the following sequence:

Procedure A (non-trusted)
Procedure B with WRITEUP
Procedure C with WRITEDOWN
Procedure D (non-trusted)
Here, Procedures B, C, and D all runs with the WRITEUP privilege, because B was the first trusted procedure in the sequence. When the sequence ends, the privilege pertaining to Procedure B is no longer in force for subsequent procedures.

Note: Unhandled exceptions raised in trusted program units are caught by Oracle Label Security. This means that error messages may not be displayed to the user. For this reason, you should always thoroughly test and debug any program units before granting them privileges.

Using SA_UTL Functions to Set and Return Label Information

The SA_UTL package provides several functions for use within PL/SQL programs. These functions return information about the current values of the session security attributes, in the form of numeric label values. Although they can be used in program units that are not trusted, these functions are primarily for use in trusted stored program units.

Note that these are public functions; you do not need the *policy*_DBA role to use them. In addition, each of the functions has a parallel SA_SESSION function that returns the same labels in character string format.

- Viewing Session Label and Row Label Using SA_UTL
- Checking Rights to Read and Update Table Row Data
- Setting the Session Label and Row Label Using SA_UTL
- Returning Greatest Lower Bound and Least Upper Bound

See Also: "Viewing Session Attributes with SA_SESSION Functions" on page 5-16

Viewing Session Label and Row Label Using SA_UTL

SA_UTL provides the following procedures for viewing session label and row label.

SA_UTL.NUMERIC_LABEL

This procedure returns the current session label. It takes a policy name as the input parameter and returns a NUMBER value.

```
SA_UTL.NUMERIC_LABEL (policy_name) RETURN NUMBER;
```

SA_UTL.NUMERIC_ROW_LABEL

This procedure returns the current row label. It takes a policy name as the input parameter and returns a NUMBER value.

```
SA_UTL.NUMERIC_ROW_LABEL (policy_name) RETURN NUMBER;
```

SA_UTL.DATA_LABEL

This function returns TRUE if the label is a *data* label.

```
FUNCTION DATA_LABEL(label IN NUMBER)
RETURN BOOLEAN;
```

Checking Rights to Read and Update Table Row Data

SA_UTL provides the following functions for checking the current session user rights to policy labeled data.

SA_UTL.CHECK_READ

Use this function to check if the user can read a policy protected table row. This function returns 1 if the user can read the table row. It returns 0 if the user cannot read the table row. The input values are the policy name and the row data label.

```
FUNCTION CHECK_READ (
  policy_name     IN VARCHAR2,
  label       IN NUMBER)
RETURN NUMBER;
```

Note: The user should already have read privileges on the table to read any data from the table.

SA_UTL.CHECK_WRITE

Use this function to check if the user can insert, update, or delete data in a policy protected table row. This function returns 1 if the user can write to the table row. It returns 0 if the user cannot write to the table row. The input values are the policy name and the row data label.

```
FUNCTION CHECK_WRITE (
  policy_name     IN VARCHAR2,
  label       IN NUMBER)
RETURN NUMBER;
```

Note: The user should already have update privileges on the table to write any data into the table.

SA_UTL.CHECK_LABEL_CHANGE

Use this function to check if the user can change the data label for a policy protected table row. This function returns 1 if the user can change the data label. It returns 0 if the user cannot change the data label. The input values are the policy name, the current data label, and the new data label.

```
FUNCTION CHECK_LABEL_CHANGE (
  policy_name     IN VARCHAR2,
  current_label   IN NUMBER,
  new_label       IN NUMBER)
RETURN NUMBER;
```

Note: The user should already have update privileges on the table to write any data into the table.

Setting the Session Label and Row Label Using SA_UTL

These procedures use numeric labels instead of character strings as input values. Available SA_SESSION procedures perform the same functions as these, but in character string format.

SA_UTL.SET_LABEL

Use this procedure to set the label of the current database session. The session's write label and row label are set to the subset of the label's compartments and groups that are authorized for write access.

```
PROCEDURE SET_LABEL (policy_name IN VARCHAR2,
                     label IN NUMBER);
```

Parameter	Specifies
policy_name	The name of an existing policy
label	The label to set as the session label

SA_UTL.SET_ROW_LABEL

Use this procedure to set the row label of the current database session. The compartments and groups in the label must be a subset of compartments and groups in the session label that are authorized for write access.

```
PROCEDURE SET_ROW_LABEL (policy_name IN VARCHAR2,
                         row_label IN NUMBER);
```

Parameter	Specifies
policy_name	The name of an existing policy
row_label	The label to set as the session default row label

> **See Also:** "Changing Your Session and Row Labels with SA_SESSION" on page 5-14

Returning Greatest Lower Bound and Least Upper Bound

Functions for greatest lower bound and least upper bound are available.

GREATEST_LBOUND

This function returns a label that is the greatest lower bound of the two label arguments.

Syntax:

```
FUNCTION GREATEST_LBOUND (label1 IN NUMBER,
                          label2 IN NUMBER)
RETURN NUMBER;
```

LEAST_UBOUND

This function returns an Oracle Label Security label that is the least upper bound of the label arguments.

Syntax:

```
FUNCTION LEAST_UBOUND (label1 IN NUMBER,
                       label2 IN NUMBER)
RETURN NUMBER;
```

> **See Also:** "Determining Upper and Lower Bounds of Labels" on page 5-9. The functions described here are the same as those described in Chapter 4, except that these return a number instead of a character string.

12

Auditing Under Oracle Label Security

The *Oracle Database* 11*g* Release 1 (11.1) audit facility lets you hold database users accountable for the operations they perform. It can track specific database objects, operations, users, and privileges. Oracle Label Security supplements this by tracking use of its own administrative operations and policy privileges. It provides the SA_AUDIT_ADMIN package to set and change the policy auditing options.

This chapter explains how to use Oracle Label Security auditing. It contains these topics:

Overview of Oracle Label Security Auditing

Oracle Label Security auditing supplements standard *Oracle Database* auditing by tracking use of its own administrative operations and policy privileges. You can use either the SA_AUDIT_ADMIN package or Oracle Enterprise Manager to set and change the auditing options for an Oracle Label Security policy.

When you create a new policy, a label column for that policy is added to the database audit trail. The label column is created regardless of whether auditing is enabled or disabled, and independent of whether database auditing or operating system auditing is used. Whenever a record is written to the audit table, each policy provides a label for that record to indicate the session label. The administrator can create audit views to display these labels. Note that in the audit table, the label does not control access to the row, instead it only records the sensitivity of the row.

The auditing options that you specify apply only to subsequent sessions, not to the current session. You can specify audit options even if auditing is disabled. No overhead is created by making only these specifications. When you do enable Oracle Label Security auditing, the options come into effect, and overhead is created beyond that created by standard *Oracle Database* auditing.

Note that Oracle Label Security does not provide labels for audit data written to the operating system audit trial. All Oracle Label Security audit records are written directly to the database audit trail, even if operating system auditing is enabled. If auditing is disabled, then no Oracle Label Security audit records are generated.

Enabling Systemwide Auditing: AUDIT_TRAIL Initialization Parameter

For Oracle Label Security to generate audit records, you must first enable systemwide auditing by setting the *Oracle Database* AUDIT_TRAIL initialization parameter in the database's parameter file. The parameter can be set to one of the following values:

Table 12–1 AUDIT_TRAIL Parameter Settings

Setting	Explanation
DB	Enables database auditing and directs all audit records to the database audit trail. This approach is recommended by Oracle. Note that even with AUDIT_TRAIL set to DB, some records are always sent to the operating system audit trail. These include STARTUP and SHUTDOWN statements, as well as CONNECT AS SYSOPER or SYSDBA.
DB_EXTENDED	Does all actions of AUDIT_TRAIL=DB and also populates the SqlBind and SqlText CLOB-type columns of the AUD$ table.
OS	Enables operating system auditing. This directs most of your *Oracle Database* audit records to the operating system, rather than to the database; the records will not contain Oracle Label Security labels. By contrast, any Oracle Label Security auditing will go to the database, with labels. If you set AUDIT_TRAIL to OS, then the Oracle Label Security-specific audit records are written to the database audit trail and the other *Oracle Database* audit records are written to the operating system audit trail (with no policy column in the operating system data).
NONE	Disables auditing. This is the default.

After you have edited the parameter file, restart the database instance to enable or disable database auditing as specified.

Set the AUDIT_TRAIL parameter before you set audit options. If you do not set this parameter, then you are still able to set audit options. However, audit records are not written to the database until the parameter is set and the database instance is restarted.

> **See Also:** For information about enabling and disabling systemwide auditing, setting audit options, and managing the audit trail, refer to *Oracle Database Administrator's Guide*
>
> For information about editing initialization parameters, refer to *Oracle Database Reference*
>
> For details about systemwide AUDIT and NOAUDIT functioning, see *Oracle Database SQL Language Reference*

Enabling Oracle Label Security Auditing with SA_AUDIT_ADMIN

After you have enabled systemwide auditing, you can use SA_AUDIT_ADMIN procedures to enable or disable Oracle Label Security auditing. To use Oracle Label Security auditing, you must have the *policy_type* role.

- Auditing Options for Oracle Label Security
- Enabling Oracle Label Security Auditing with SA_AUDIT_ADMIN.AUDIT
- Disabling Oracle Label Security Auditing with SA_AUDIT_ADMIN.NOAUDIT
- Examining Audit Options with the DBA_SA_AUDIT_OPTIONS View

Auditing Options for Oracle Label Security

The AUDIT and NOAUDIT options are as follows:

Table 12–2 Auditing Options for Oracle Label Security

Option	Description
APPLY	Audits application of specified Oracle Label Security policies to tables and schemas
REMOVE	Audits removal of specified Oracle Label Security policies from tables and schemas
SET	Audits the setting of user authorizations, and user and program privileges
PRIVILEGES	Audits use of all policy-specific privileges

Enabling Oracle Label Security Auditing with SA_AUDIT_ADMIN.AUDIT

Use the AUDIT procedure to enable policy-specific auditing.

Syntax:

```
PROCEDURE AUDIT (
     policy_name     IN VARCHAR2,
     users           IN VARCHAR2 DEFAULT NULL,
     option          IN VARCHAR2 DEFAULT NULL,
     type            IN VARCHAR2 DEFAULT NULL,
     success         IN VARCHAR2 DEFAULT NULL);
```

Parameter	Description	Default Behavior
policy_name	Required. Specifies the name of an existing policy. Auditing of each policy is independent of all others.	None
users	Optional. A comma-delimited list of user names to audit. If not specified, then all users are audited.	All users
option	Optional. A comma-delimited list of options to be audited. Refer to Table 12–1. If not specified, then all default options (that is, options not including privileges) are audited. Audit options for privileged operations should be set explicitly by specifying the PRIVILEGES option, which sets audit options for all privileges.	All options
type	Optional. BY ACCESS or BY SESSION. If not specified, then audit records are written by session.	BY SESSION
success	Optional. SUCCESSFUL or NOT SUCCESSFUL. If not specified, then audit is written for both.	SUCCESSFUL and NOT SUCCESSFUL

If the administrator does not specify any audit options, then all options except the privilege-related ones are audited. Auditing of privileges must be specified explicitly. For example, if the administrator enters

```
SA_AUDIT_ADMIN.AUDIT ('HR');
```

then default auditing options are set for the HR policy. When the administrator enables auditing, it will be performed on all users by session, whether successful or not.

When you set auditing parameters and options, the new values apply only to subsequent sessions, not to the current session.

Consider also a case in which one AUDIT call (with no users specified) enables auditing for APPLY operations for all users, and then a second call enables auditing of REMOVE operations for a specific user. For example:

```
SA_AUDIT_ADMIN.AUDIT ('HR', NULL, 'APPLY');
SA_AUDIT_ADMIN.AUDIT ('HR', 'SCOTT', 'REMOVE');
```

In this case, SCOTT is audited for both APPLY and REMOVE operations.

Disabling Oracle Label Security Auditing with SA_AUDIT_ADMIN.NOAUDIT

To disable policy-specific auditing, use the SA_AUDIT_ADMIN.NOAUDIT procedure.

Syntax:

```
PROCEDURE NOAUDIT (
     policy_name     IN VARCHAR2,
     users           IN VARCHAR2 DEFAULT NULL,
     option          IN VARCHAR2 DEFAULT NULL);
```

Parameter	Description	Default Behavior
policy_name	Required. Specifies the name of an existing policy.	None
users	Optional. A comma-delimited list of user names to audit. If not specified, then auditing is disabled for all users.	All users
option	Optional. A comma-delimited list of options to be disabled.Refer to Table 12–2. If not specified, then all default options are disabled. Privileges must be disabled explicitly.	All options

You can disable auditing for all enabled options, or only for a subset of enabled options. All auditing for the specified options is disabled for all specified users (or all users, if the *users* parameter is NULL). For example, the following statement disables auditing of the APPLY and REMOVE operations for users John, Mary, and Scott:

```
SA_AUDIT_ADMIN.NOAUDIT ('HR', 'JOHN, MARY, SCOTT', 'APPLY, REMOVE');
```

Consider also a case in which one AUDIT call enables auditing for a specific user, and a second call (with no user specified) enables auditing for all users. For example:

```
SA_AUDIT_ADMIN.AUDIT ('HR', 'SCOTT');
SA_AUDIT_ADMIN.AUDIT ('HR');
```

In this case, a subsequent call to NOAUDIT with no users specified (such as the following)

```
SA_AUDIT_ADMIN.NOAUDIT ('HR');
```

does not reverse the auditing that was set for SCOTT explicitly in the first call. So, auditing continues to be performed on SCOTT. In this way, even if NOAUDIT is set for all users, Oracle Label Security still audits any users for whom auditing was explicitly set.

Auditing of privileged operations must be specified explicitly. If you run NOAUDIT with no options, the Oracle Label Security will nonetheless continue to audit privileged operations. For example, if auditing is enabled and you enter

```
SA_AUDIT_ADMIN.NOAUDIT ('HR');
```

then auditing will continue to be performed on the privileged operations (such as WRITEDOWN).

NOAUDIT parameters and options that you set apply only to subsequent sessions, not to current sessions.

If you try to enable an audit option that has already been set, or if you try to disable an audit option that has not been set, then Oracle Label Security processes the statement without indicating an error. An attempt to specify an invalid option results in an error message.

Examining Audit Options with the DBA_SA_AUDIT_OPTIONS View

This section describes the view that displays the Oracle Label Security auditing options and privileges.

The DBA_SA_AUDIT_OPTIONS view contains the following columns:

Table 12–3 Columns in the DBA_SA_AUDIT_OPTIONS View

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
APY		VARCHAR2(3)
REM		VARCHAR2(3)
SET_		VARCHAR2(3)
PRV		VARCHAR2(30

Output is similar to the following:

Table 12–4 DBA_SA_AUDIT_OPTIONS Sample Output

POLICY_ NAME	USER_NAME	APY	REM	SET	PRV
HR	SCOTT	-/-	-/-	-/-	A/A
HR	LBACSYS	S/S	S/S	S/S	-/-

See Also: Chapter 11 of the *Oracle Database Security Guide*

Managing Policy Label Auditing

This section describes procedures available to manage policy label auditing:

- Policy Label Auditing with SA_AUDIT_ADMIN.AUDIT_LABEL
- Disabling Policy Label Auditing with SA_AUDIT_ADMIN.NOAUDIT_LABEL
- Finding Label Audit Status with AUDIT_LABEL_ENABLED

Policy Label Auditing with SA_AUDIT_ADMIN.AUDIT_LABEL

Use the AUDIT_LABEL procedure to record policy labels during auditing. It causes the user's session label to be stored in the audit table.

Syntax:

```
PROCEDURE AUDIT_LABEL (
     policy_name     IN VARCHAR2);
```

Parameter	Description	Default
policy_name	Required. Specifies the name of an existing policy.	None

Disabling Policy Label Auditing with SA_AUDIT_ADMIN.NOAUDIT_LABEL

Use the NOAUDIT_LABEL procedure to disable auditing of policy labels.

Syntax:

```
PROCEDURE NOAUDIT_LABEL (
     policy_name     IN VARCHAR2);
```

Parameter	Description	Default
policy_name	Required. Specifies the name of an existing policy.	None

Finding Label Audit Status with AUDIT_LABEL_ENABLED

Use the AUDIT_LABEL_ENABLED function to show whether labels are being recorded in audit records for the policy.

Syntax:

```
FUNCTION AUDIT_LABEL_ENABLED (policy_name IN VARCHAR2)
     RETURN boolean;
```

Creating and Dropping an Audit Trail View for Oracle Label Security

This section contains these topics:

- Creating a View with SA_AUDIT_ADMIN.CREATE_VIEW
- Dropping a View with SA_AUDIT_ADMIN.DROP_VIEW

Creating a View with SA_AUDIT_ADMIN.CREATE_VIEW

The CREATE_VIEW procedure creates an audit trail view named DBA_*policyname*_AUDIT_TRAIL, which contains the specified policy's label column as well as all the entries in the audit trail written on behalf of this policy. If the view name exceeds the database limit of 30 characters, then the user can optionally specify a shorter view name.

Syntax (either of two):

1. A one-parameter procedure:

   ```
   PROCEDURE CREATE_VIEW (
        policy_name     IN VARCHAR2);
   ```

   where *policy_name* specifies the name of an existing policy.

   or

2. A two-parameter procedure:

```
PROCEDURE CREATE_VIEW (
     policy_name     IN VARCHAR2,
     view_name       IN VARCHAR2    DEFAULT NULL);
```

where *policy_name* specifies the name of an existing policy and *view_name* is an optional parameter, maximum 14 characters, specifying the desired view name.

Dropping a View with SA_AUDIT_ADMIN.DROP_VIEW

The DROP_VIEW procedure drops the audit trail view for the specified policy.

Syntax (either of two):

1. A one-parameter procedure:

```
PROCEDURE DROP_VIEW (
     policy_name     IN VARCHAR2);
```

where *policy_name* specifies the name of an existing policy.

or

2. A two-parameter procedure:

```
PROCEDURE DROP_VIEW (
     policy_name     IN VARCHAR2,
     view_name       IN VARCHAR2    DEFAULT NULL);
```

where *policy_name* specifies the name of an existing policy and *view_name* is an optional parameter, maximum 14 characters, specifying an existing view's name .

Note: When sa_audit_admin.create_view was used to create a pre-10i audit view, that view did not show the timestamp field for the audit records in 10i. Oracle Label Security recommends that all pre-10i Oracle Label Security audit views be dropped and re-created, by using sa_audit_admin.drop_view and sa_audit_admin.create_view.

Oracle Label Security Auditing Tips

This section contains these topics:

- Strategy for Setting SA_AUDIT_ADMIN Options
- Auditing Privileged Operations

Strategy for Setting SA_AUDIT_ADMIN Options

Before setting any audit options, you must devise an auditing strategy that monitors events of interest, without recording extraneous events. You should periodically review this strategy, because applications, user base, configurations, and other external factors can change.

The Oracle Label Security options, and those provided by the *Oracle Database* audit facility, might not directly address all of your specific or application-dependent auditing requirements. However, through use of database triggers, you can audit specific events and record specific information that you cannot audit and record using the more generic audit facility.

See Also: For more information about using triggers for auditing, see *Oracle Database Concepts*

Auditing Privileged Operations

Consider auditing any operations that require Oracle Label Security privileges. Because these privileges perform sensitive operations, and because their abuse could jeopardize security, you should closely monitor their dissemination and use.

13

Using Oracle Label Security with a Distributed Database

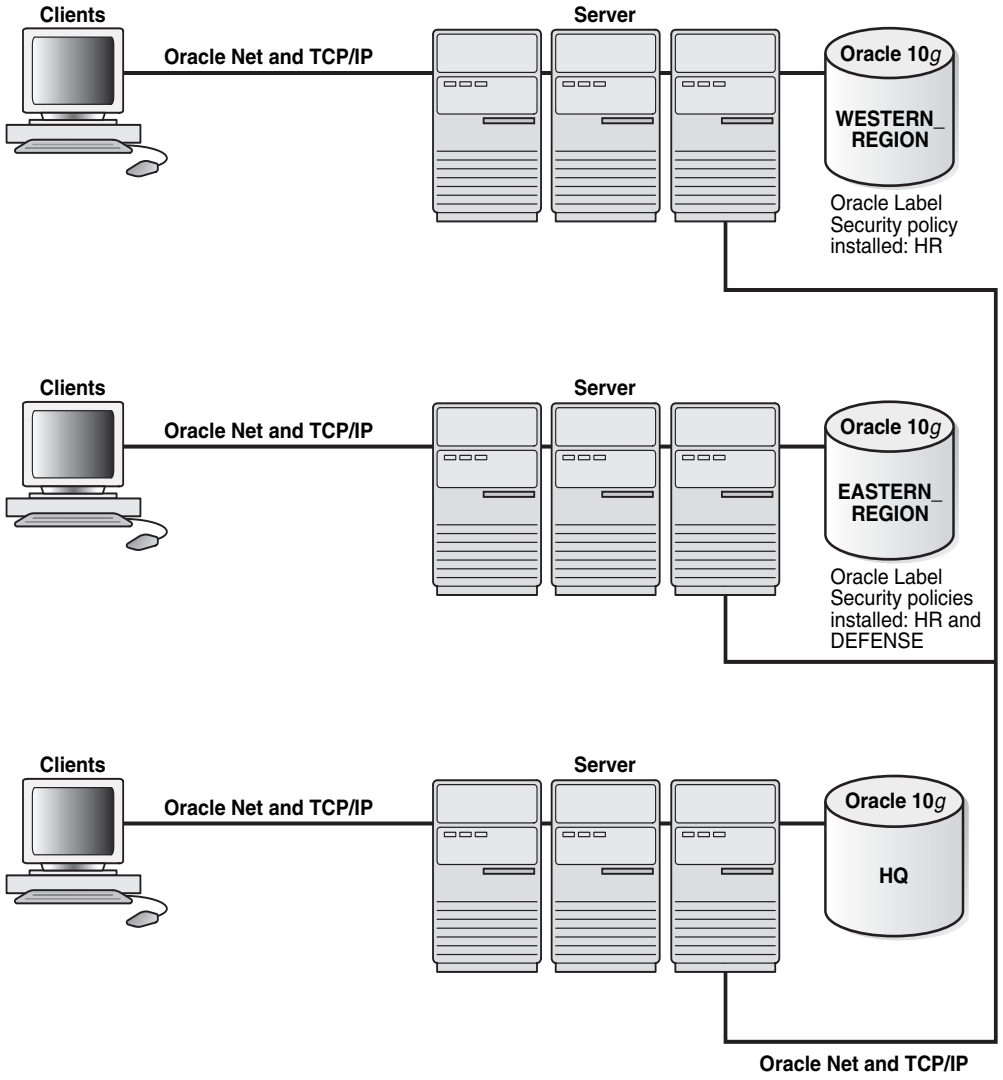
This chapter describes special considerations for using Oracle Label Security in a distributed configuration. It contains the following sections:

An Oracle Label Security Distributed Configuration

A network configuration that supports distributed databases can include multiple *Oracle Database* servers, or other database servers, running on the same or different operating systems. Each cooperative server in a distributed system communicates with other clients and servers over a network.

Figure 13–1, "Using Oracle Label Security with a Distributed Database" illustrates a distributed database that includes clients and servers with and without Oracle Label Security. As described in this chapter, if you establish database links from the WESTERN_REGION database to the EASTERN_REGION database, then you can access data if your userid on EASTERN_REGION is authorized to see it, even if locally (on WESTERN_REGION) you do not have this access.

Figure 13–1 Using Oracle Label Security with a Distributed Database



Connecting to a Remote Database Under Oracle Label Security

Distributed databases act in the standard way with Oracle Label Security: the local user ends up connected as a particular remote user. Oracle Label Security protects the labeled data, whether you connect locally or remotely. If the remote user has the proper labels, then you can access the data. If not, then you cannot access the data.

The database link sets up the connection to the remote database and identifies the user who will be associated with the remote session. Your Oracle Label Security authorizations on the remote database are based on those of the remote user identified in the database link.

For example, local user JANE might connect as remote user AUSTEN, in the database referenced by the connect string `sales`, as follows:

```
CREATE DATABASE LINK sales
  CONNECT TO austen IDENTIFIED BY pride
  USING 'sales'
```

When JANE connects, her authorizations are based on the labels and privileges of remote user AUSTEN, because AUSTEN is the user identified in the database link. When JANE makes the first reference to the remote database, the remote session is actually established. For example, the remote session would be created if JANE enters:

```
SELECT * FROM emp@sales
```

You need not be an Oracle Label Security policy user in the local database. If you connect as a policy user on the remote database, you can access protected data.

Establishing Session Label and Row Label for a Remote Session

When connecting remotely, you can directly control the session label and row label in effect when you establish the connection. When you connect, Oracle Label Security passes these values (for all policies) over to the remote database. Notice that:

- The local session label and row label are used as the default for the remote session, if they are valid for the remote user.
- The remote session is constrained by the minimum and maximum authorizations of the remote user.
- Although the local user's session labels are passed to the remote database, the local user's privileges are not passed. The privileges for the remote session are those associated with the remote user.

Consider a local user, Diana, with a maximum level of HS, and a session level of S. On the remote database, the remote user identified in the database link has a maximum level of S.

- If Diana's session label is S when the database link is established, then the S label is passed over. This is a valid label. Diana can connect and read SENSITIVE data.
- If Diana's session label is HS when the database link is established, then the HS level is passed across, but it is not valid for the remote user. Diana will pick up the remote user's default label (S).

Be aware of the label at which you are running the first time you connect to the remote database. The first time you reference a database link, your local session labels are sent across to the remote system when a connection is made. Later, you can change the label, but to do so, you must run the SA_SESSION.SET_LABEL procedure on the remote database.

Diana can connect at level HS, set the label to S, and then perform a remote access. Connection is implicitly made when the database link is established. Her default label is S on the remote database.

On the local database, Diana can set her session label to her maximum level of HS, but if the label of the remote user is set to S, then she can only retrieve S data from the remote database. If she performs a distributed query, then she will get HS data from the local database, and S data from the remote database.

Setting Up Labels in a Distributed Environment

It is advisable to use the same label component definitions and label tags on any database that is to be protected by the policy.

- Setting Label Tags in a Distributed Environment
- Setting Numeric Form of Label Components in a Distributed Environment

Setting Label Tags in a Distributed Environment

In a distributed environment, you may choose to use the same label tags across multiple databases. However, if you choose *not* to use the same tags across multiple databases, then you should retrieve the character form of the label when performing remote operations. This will ensure that the labels are consistent.

In the following example, the character string representation of the label string is the same. However, the label tag does not match. If the retrieved label tag has a value of 11 on the WESTERN_REGION database but a tag of 2001 on the EASTERN_REGION database, then the tags have no meaning. Serious consequences can result.

Figure 13–2 Label Tags in a Distributed Database

EASTERN_REGION

Label	Label Tag
S:A	3001
C:A	2001
U	10

WESTERN_REGION

Label	Label Tag
S:A	11
C:A	6
U	5

When retrieving labels from a remote system, you should return the character string representation (rather than the numeric label tag), unless you are using the same numeric labels on both databases.

If you allow Oracle Label Security to automatically generate labels on different databases, then the label tags will not be identical. Character strings will have meaning, but the numeric values will not, unless you have predefined labels with the same label tags on both instances.

To avoid the complexities of label tags, you can convert labels to strings on retrieval (using LABEL_TO_CHAR) and use CHAR_TO_LABEL when you store labels. Operations will succeed as long as the component names are the same.

Setting Numeric Form of Label Components in a Distributed Environment

In a distributed environment, you should use the same relative ranking of the numeric form of the level component, to ensure proper sorting of the labels.

In the following example, the levels in the two databases are effectively the same. Although the numeric form is different, the relative ranking of the levels numeric form is the same. As long as the relative order of the components is the same, the labels are perceived as identical.

Figure 13–3 Label Components in a Distributed Database

EASTERN_REGION

Level	Numeric Form
S	30
C	20
U	10

WESTERN_REGION

Level	Numeric Form
S	6
C	5
U	4

Using Oracle Label Security Policies in a Distributed Environment

Oracle Label Security supports all standard *Oracle Database* distributed configurations. Whether or not you can access protected data depends on the policies installed in each distributed database.

Be sure to take into account the relationships between databases in a distributed environment:

- If the same application runs on two databases and you want them to have the same protection, then syou must apply the same Oracle Label Security policy to both the local and the remote databases.
- If the local and remote databases have a policy in common, then your local session label and row label will override the default labels for the remote user.
- If the remote database has a different policy than the local database, then the remote policy can restrict access to the data independent of your local policies. On the other hand, when you make a connection as a remote user who has authorization on the remote policy, you can access any data to which the remote user has access to, regardless of your local authorizations.

If the remote database has no policy applied to it, you can access its data just as you would with a standard distributed database.

Consider a situation in which three databases exist, with different Oracle Label Security policies in force:

Database 1 has Policy A and Policy B
Database 2 has Policy A
Database 3 had Policy C

Users authorized for Policy A can obtain protected data from Database 1 and Database 2. If the remote user is authorized for Policy C, then this user can obtain data from Database 3 as well.

Using Replication with Oracle Label Security

This section explains how to use the replication option with tables protected by Oracle Label Security policies. It contains these topics:

- Introduction to Replication Under Oracle Label Security
- Contents of a Materialized View
- Requirements for Creating Materialized Views Under Oracle Label Security
- How to Refresh Materialized Views

See Also:

- For a complete explanation of replication in *Oracle Database* 11*g* Release 1 (11.1) and how to set up the replication environment, refer to *Oracle Database Advanced Replication*.
- For general information about using materialized views, refer to *Oracle Database Concepts* and*Oracle Database Data Warehousing Guide*.

Introduction to Replication Under Oracle Label Security

This section introduces the use of replication under Oracle Label Security. It contains the following topics:

- Replication Functionality Supported by Oracle Label Security
- Row-Level Security Restriction on Replication Under Oracle Label Security

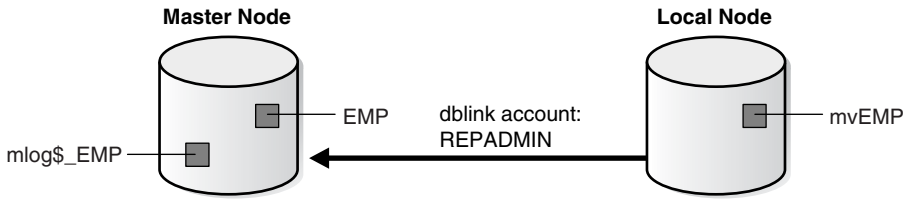
Replication Functionality Supported by Oracle Label Security

Oracle Label Security supports standard replication and Advanced Replication, including multimaster replication and updatable materialized views (snapshots).

Oracle Database uses materialized views for replicating data. A *materialized view* is a local copy of a local or remote master table that reflects a recent state of the master table.

As illustrated in Figure 13–4, "Use of Materialized Views for Replication", a master table is a table you wish to replicate, on a node that you designate as the master node. Using a dblink account (such as REPADMIN), you can create a materialized view of the table in a different database. (This can also be done in the same database, and on the same system.) You can select rows from the remote master table, and copy them into the local materialized view. Here, `mvEMP` represents the materialized view of table `EMP`, and `mlog$_EMP` represents the materialized view log.

Figure 13–4 Use of Materialized Views for Replication



In a distributed environment, a materialized view alleviates query traffic over the network and increases data availability when a node is not available.

Row-Level Security Restriction on Replication Under Oracle Label Security

An Oracle Label Security policy applies Row Level Security (RLS) to a table if READ_CONTROL is specified as one of the policy options. Problems occur if *both* of the following conditions are true:

- The Oracle Label Security policy is applied to any table relevant to replication (such as the master table, materialized view, or materialized view log), and
- The policy returns a predicate in the WHERE clause of SELECT statements.

To avoid the additional predicate (and therefore avoid this problem), the users involved in a replication environment should be given the necessary Oracle Label Security privileges. To be specific, the designated users in the database link (such as REPADMIN and the materialized view owner) must have the READ or the FULL privilege. As a result, the queries used to perform the replication will not be modified by RLS.

See Also: *Oracle Database Concepts*

Contents of a Materialized View

This section discusses the contents of materialized views.

How Materialized View Contents Are Determined

Oracle Label Security performs the following steps when creating materialized views. These steps determine the contents of the view.

1. It reads the definition of the master table in the remote database.
2. It reads the rows in the master table that meet the conditions defined in the materialized view definition.
3. It writes these rows to the materialized view in the local database.

Because Oracle Label Security writes only those rows to which you have write access in the local database, the contents of the materialized view vary according to:

- The policy options in effect
- The privileges you have defined in the local database
- The session label

Complete Materialized Views

If you read all of the rows in the master table and have write access in the local database to each label in the materialized view, then the result is a complete materialized view of the master table. To ensure that the materialized view is complete, ensure that you have read access to all of the data in the master table and write access in the local database to all labels at which data is stored in the master table.

Note: Never revoke privileges that you granted when you created the materialized view. If you do, then you may not be able to perform a replication refresh.

Partial Materialized Views

A partial materialized view is created when you specify a WHERE clause in the materialized view definition. This is a convenient way to pass subsets of data to a remote database.

Note: To create a partial materialized view, you must have write access to all the rows being replicated.

Requirements for Creating Materialized Views Under Oracle Label Security

Requirements for creating a materialized view depend on the type of materialized view you are creating.

- Requirements for Creating Partial Multilevel Materialized Views
- Requirements for Creating Complete Multilevel Materialized Views

Requirements for the REPADMIN Account

Requirements for the REPADMIN account vary depending on the configuration. In general, however, it should meet the following requirements:

- It must have the FULL Oracle Label Security privilege (mandatory for all configurations).
- It must have the SELECT privilege on the master table.
- It must be the account that establishes the database link from the remote node to the database containing the master table.

See Also: *Oracle Database Advanced Replication*

Requirements for the Owner of the Materialized View

Remember that the privileges belonging to the owner of the materialized view are used during the refresh of the materialized view. If these privileges are not sufficient, then there are two options:

- The materialized view can be created in the REPADMIN account, or
- Additional privileges must be granted to the owner of the materialized view.

Consider, for example, the following materialized view created by user SCOTT:

```
CREATE MATERIALIZED VIEW mvemp as
SELECT *
FROM EMP@link_to_master
WHERE label_to_char(sa_label) = 'HS';
```

Here, SCOTT should have permission to insert records at the HS level in the local database. If Oracle Label Security policies are applied on the materialized view, then SCOTT must have the FULL privilege to avoid the RLS restriction.

Different configurations can be set up depending on whether Oracle Label Security policies are applied on the materialized view, what privileges are granted to the owner of the materialized view, and so on. If Oracle Label Security policies are applied to the materialized view, but SCOTT should not be granted the FULL privilege, then the REPADMIN account must be used to create the materialized view. SCOTT can then be granted the SELECT privilege on that table.

If no policies are applied to the materialized view, then the view can be created in SCOTT's schema without any additional privileges. In this case, the materialized view should be created in such a way that a WHERE condition limits the records to those which SCOTT can read.

Finally, if SCOTT can be granted the FULL privilege, then the materialized view can be created in SCOTT's schema, and Oracle Label Security policies can also be applied on the materialized view.

Note that the master table can have Oracle Label Security policies containing any set of policy options. If SCOTT has the FULL or the READ privilege, he can select all rows, regardless of policy options.

Requirements for Creating Partial Multilevel Materialized Views

To create a partial materialized view that includes only some of the rows in a remote master table protected by Oracle Label Security, you must have sufficient privileges to WRITE in the local database at every label retrieved by your query.

Requirements for Creating Complete Multilevel Materialized Views

To create a complete materialized view that includes every row in a remote master table protected by Oracle Label Security, you must be able to WRITE in the local database at the labels of all of the rows retrieved by the defined materialized view query.

How to Refresh Materialized Views

If the contents or definition of a master table changes, then refresh the materialized view so that it accurately reflects the contents of the master table. To refresh a materialized view of a remote multilevel table, you must also have privileges to write in the local database at the labels of all of the rows that the materialized view query retrieves

---

Warning: A materialized view can potentially contain outdated rows if you refresh a partial or full materialized view but do not have READ access to all the rows in the master table, and consequently do not overwrite the rows in the original materialized view with the updated rows from the master table.

To ensure an accurate materialized view refresh, use the optional materialized view background processes, SNP*n*, to refresh the views automatically. These processes must have sufficient privileges both to read all of the rows in the master table and to write those rows to the materialized view, ensuring that the view is completely refreshed. Remember that the privileges used by these processes are those of the materialized view owner.

> **See Also:** For information about SNP*n* background processes, refer to *Oracle Database Administrator's Guide*

14

Performing DBA Functions Under Oracle Label Security

The standard *Oracle Database* 11*g* Release 1 (11.1) utilities can be used under Oracle Label Security, but certain restrictions apply, and extra steps may be required to get the expected results. This chapter describes these special considerations. It assumes that you are using policy label columns of the NUMBER data type.

The chapter contains these sections:

Using the Export Utility with Oracle Label Security

The Export utility functions in the standard way under Oracle Label Security. There are, however, a few differences resulting from the enforcement of Oracle Label Security policies.

- For any tables protected by an Oracle Label Security policy, only rows with labels authorized for read access will be exported. Unauthorized rows will not be included in the export file. Consequently, to export *all* the data in protected tables, you must have a privilege (such as FULL or READ) that gives you complete access.
- SQL statements to reapply policies are exported along with tables and schemas that are exported. These statements are carried out during import to reapply policies with the same enforcement options as in the original database.
- The HIDE property is not exported. When protected tables are exported, the label columns in those tables are also exported (as numeric values). However, if a label column is hidden, then it is exported as a normal, unhidden column.
- The LBACSYS schema cannot be exported due to the use of opaque types in Oracle Label Security. An export of the entire database (parameter FULL=Y) with Oracle Label Security installed can be done, except that the LBACSYS schema would not be exported.

See Also: *Oracle Database Utilities*

Using Datapump Export Utility with Oracle Label Security

The user must have EXEMPT ACCESS POLICY in order to export all rows in the table, or else no rows are exported. This restriction is in addition to the export restrictions discussed earlier.

Using the Import Utility with Oracle Label Security

This section explains how the Import utility functions under Oracle Label Security:

- Requirements for Import Under Oracle Label Security
- Defining Data Labels for Import
- Importing Labeled Data Without Installing Oracle Label Security
- Importing Unlabeled Data
- Importing Tables with Hidden Columns

See Also: *Oracle Database Utilities*

Requirements for Import Under Oracle Label Security

To use the Import utility under Oracle Label Security, you must prepare the import database and ensure that the import user has the proper authorizations.

Preparing the Import Database

Before you can use the Import utility with Oracle Label Security, you must prepare the import database, as follows:

1. Install Oracle Label Security.
2. Create any Oracle Label Security policies that protect the data to be imported. The policies must use the same column names as in the export database.
3. Define in the import database all of the label components and individual labels used in tables being imported. Tag values assigned to the policy labels in each database must be the same. (Note that if you are importing into a database from which you exported, then the components are most likely already defined.)

Verifying Import User Authorizations

To successfully import data under Oracle Label Security, the user running the import operation must be authorized for all of the labels required to insert the data and labels contained in the export file. Errors will be raised upon import if the following requirements are not met:

Requirement 1: To assure that all rows can be imported, the user must have the *policy*_DBA role for all policies with data being imported. After each schema or table is imported, any policies from the export database are reapplied to the imported objects.

Requirement 2: The user must also have the ability to write all rows that have been exported. This can be accomplished by one of the following methods:

- The user can be granted the FULL privilege.
- A user-defined labeling function can be applied to the table.
- The user can be given sufficient authorization to write all labels contained in the import file.

Defining Data Labels for Import

The label definitions at the time of import must include all the policy labels used in the export file. You can use the views DBA_SA_LEVELS, DBA_SA_COMPARTMENTS, DBA_SA_GROUPS, and DBA_SA_LABELS in the export database to design SQL scripts that re-create the label components and labels for each policy in the import database. The following example shows how to generate a PL/SQL block that re-creates the individual labels for the HR policy:

```
set serveroutput on
BEGIN
   dbms_output.put_line('BEGIN');
   FOR l IN (SELECT label_tag, label
                FROM dba_sa_labels
                WHERE policy_name='HR'
                ORDER BY label_tag) LOOP
       dbms_output.put_line
           ('  SA_LABEL_ADMIN.CREATE_LABEL(''HR'', ' ||
            l.label_tag || ', ''' || l.label || ''');');
   END LOOP;
   dbms_output.put_line ('END;');
   dbms_output.put_line ('/');
END;
/
```

If the individual labels do not exist in the import database with the same numeric values and the same character string representations as in the export database, then the label values in the imported tables will be meaningless. The numeric label value in the table may refer to a different character string representation, or it may be a label value that has not been defined at all in the import database.

If a user attempts to access rows containing invalid numeric labels, then the operation will fail.

Importing Labeled Data Without Installing Oracle Label Security

When policy label columns are defined as a NUMBER data type, they can be imported into databases that do not have Oracle Label Security installed. In this case, the values in the policy label column are imported as numbers. Without the corresponding Oracle Label Security label definitions, the numbers will not reference any specific label.

Note that errors will be raised during the import if Oracle Label Security is not installed, because the SQL statements to reapply the policy to the imported tables and schemas will fail.

Importing Unlabeled Data

You can import unlabeled data into an existing table protected by an Oracle Label Security policy. Either the LABEL_DEFAULT option or a labeling function must be specified for each table being imported, so that the labels for the rows can be automatically initialized as they are inserted into the table.

Importing Tables with Hidden Columns

A hidden column is exported as a normal column, but the fact that it was hidden is lost. If you want to preserve the hidden property of the label column, you must precreate the table in the import database.

1. Before you perform the import, create the table and apply the policy with the HIDE option. This causes the policy label column to be added to the table as a hidden column.
2. Then remove the policy from the table, so that the enforcement options specified in the export file can be reapplied to the table during the import operation.
3. Perform the import with `IGNORE=Y`. Setting the `IGNORE` parameter to `Y` ignores errors during import.
4. Manually apply the policy to the table with the HIDE option.

Using SQL*Loader with Oracle Label Security

SQL*Loader moves data from external files into tables in Oracle Database. This section contains these topics:

- Requirements for Using SQL*Loader Under Oracle Label Security
- Oracle Label Security Input to SQL*Loader

> **See Also:** For information about SQL*Loader, including log files, discard files, and bad files, see *Oracle Database Utilities*

Requirements for Using SQL*Loader Under Oracle Label Security

You can use SQL*Loader with the conventional path to load data into a database protected by Oracle Label Security. Because SQL*Loader performs INSERT operations, all of the standard requirements apply when using SQL*Loader on tables protected by Oracle Label Security policies.

Oracle Label Security Input to SQL*Loader

If the policy column for a table is hidden, then you must use the HIDDEN keyword to convey this information to SQL*Loader.

To specify row labels in the input file, include the policy label column in the INTO TABLE clause in the control file.

To load policy labels along with the data for each row, you can specify the CHAR_TO_LABEL function or the TO_DATA_LABEL function in the SQL*Loader control file.

Note: When Oracle Label Security is installed to work with Oracle Internet Directory, dynamic label generation is not allowed, because labels are managed centrally in Oracle Internet Directory, using olsadmintool commands. Refer to Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory".

So, when Oracle Label Security is directory-enabled, this function, TO_DATA_LABEL, is not available and will generate an error message if used.

You can use the following variations when loading Oracle Label Security data with SQL*Loader:

*Table 14–1 Input Choices for Oracle Label Security Input to SQL*Loader*

Form of Data	Explanation of Results
`col1 hidden integer external`	Hidden column loaded with tag value of data directly from data file
`col2 hidden char(5) "func(:col2)"`	Hidden column loaded with character value of data from data file. func() used to translate between the character label and its tag value. Note: func() might be char_to_label().
`col3 hidden "func(:col3)"`	Same as in col2 specified earlier, field type defaults to char
`col4 hidden expression "func(:col4)"`	Hidden column not mapped to input data. func() will be called to provide the label value. This could be a user function.

For example, the following is a valid INTO TABLE clause in a control file that is loading data into the DEPT table:

```
INTO TABLE dept
(hr_label HIDDEN POSITION (1:22) CHAR "CHAR_TO_LABEL('HR',:hr_label)",
deptno    POSITION (23:26) INTEGER EXTERNAL,
dname     POSITION (27:40) CHAR,
loc       POSITION(41,54)  CHAR)
```

The following could be an entry in the data file specified by this control file:

```
HS:FN                 231 ACCOUNTING  REDWOOD SHORES
```

Performance Tips for Oracle Label Security

This section explains how to achieve optimal performance with Oracle Label Security.

- Using ANALYZE to Improve Oracle Label Security Performance
- Creating Indexes on the Policy Label Column
- Planning a Label Tag Strategy to Enhance Performance
- Partitioning Data Based on Numeric Label Tags

Using ANALYZE to Improve Oracle Label Security Performance

Run the ANALYZE command on the Oracle Label Security data dictionary tables in the LBACSYS schema, so that the cost-based optimizer can improve execution plans on queries. This will improve Oracle Label Security performance.

Running ANALYZE on application tables improves the application SQL performance.

Creating Indexes on the Policy Label Column

By creating the appropriate type of index on the policy label column, you can improve the performance of user-raised queries on protected tables.

If you have applied an Oracle Label Security policy on a database table in a particular schema, then you should compare the number of different labels to the amount of data. Based on this information, you can decide which type of index to create on the policy label column.

- If the cardinality of data in the policy label column (that is, the number of labels compared to the number of rows) is low, then consider creating a bitmapped index.

- If the cardinality of data in the policy label column is high, then consider creating a B-tree index.

Example 1:

Consider the following case, in which the EMP table is protected by an Oracle Label Security policy with the READ_CONTROL enforcement option set, and HR_LABEL is the name of the policy label column. A user raises the following query:

```
SELECT COUNT (*) FROM scott.emp;
```

In this situation, Oracle Label Security adds a predicate based on the label column. For example:

```
SELECT COUNT (*) FROM scott.emp
  WHERE hr_label=100;
```

In this way, Oracle Label Security uses the security label to restrict the rows that are processed, based on the user's authorizations. To improve performance of this query, you could create an index on the HR_LABEL column.

Example 2:

Consider a more complex query (once again, with READ_CONTROL applied to the EMP table):

```
SELECT COUNT (*) FROM scott.emp
  WHERE deptno=10
```

Again, Oracle Label Security adds a predicate based on the label column:

```
SELECT COUNT (*) FROM scott.emp
  WHERE deptno=10
  AND hr_label=100;
```

In this case, you might want to create a composite index based on the DEPTNO and HR_LABEL columns, to improve application performance.

See Also: *Oracle Database Performance Tuning Guide*

Planning a Label Tag Strategy to Enhance Performance

For optimal performance, you can plan a strategy for assigning values to label tags. In general, it is best to assign higher numeric values to labels with higher sensitivity levels. This is because, typically, many more users can see data at comparatively low levels and fewer users at higher levels can see many levels of data.

In addition, with READ_CONTROL set, Oracle Label Security generates a predicate that uses a BETWEEN clause to restrict the rows to be processed by the query. As illustrated in the following example, if the higher-sensitivity labels do not have a higher label tag than the lower-sensitivity labels, then the query will potentially examine a larger set of rows. This will affect performance.

Consider, for example, label tags assigned as follows:

Table 14–2 Label Tag Performance Example: Correct Values

Label	Label Tag
TS:A,B	100
S:A	50
S	20

Table 14–2 (Cont.) Label Tag Performance Example: Correct Values

Label	Label Tag
U:A	10

Here, a user whose maximum authorization is S:A can potentially access data at labels S:A, S, and U:A. Consider what happens when this user raises the following query:

```
SELECT COUNT (*) FROM scott.emp;
```

Oracle Label Security adds a predicate that includes a BETWEEN clause (based on the user's maximum and minimum authorizations) to restrict the set of rows this user can see:

```
SELECT COUNT (*) FROM scott.emp
  WHERE hr_label BETWEEN 10 AND 50;
```

Performance improves, because the query examines only a subset of data based on the user's authorizations. It does not fruitlessly process rows that the user is not authorized to access.

By contrast, unnecessary work would be performed if tag values were assigned as follows:

Table 14–3 Label Tag Performance Example: Incorrect Values

Label	Label Tag
TS:A,B	50
S:A	100
S	20
U:A	10

In this case, the user with S:A authorization can see only some of the labels between 100 and 10. Although the user cannot see TS:A,B labels (that is, rows with a label tag of 50). A query would nonetheless pick up and process these rows, even though the user ultimately will not have access to them.

Partitioning Data Based on Numeric Label Tags

If you are using a numeric ordering strategy with the numeric label tags that you have applied to the labels, then you can use this as a basis for *Oracle Database* data partitioning. Depending on the application, partitioning data based on label values may or may not be useful.

Consider, for example, a business-hosting CRM application to which many companies subscribe. In the same EMP table, there might be rows (and labels) for Subscriber 1 and Subscriber 2. That is, information for both companies can be stored in the same table, as long as it is labeled differently. In this case, employees of Subscriber 1 will never need to access data for Subscriber 2, so it might make sense to partition based on label. You could put rows for Subscriber 1 in one partition, and rows for Subscriber2 in a different partition. When a query is raised, it will access only one or the other partition, depending on the label. Performance improves because partitions that are not relevant are not examined by the query.

The following example shows this is done. It places labels in the 2000 series on one partition, labels in the 3000 series on another partition, and labels in the 4000 series on a third partition.

```
CREATE TABLE EMPLOYEE
       (EMPNO NUMBER(10) CONSTRAINT PK_EMPLOYEE PRIMARY KEY,
    ENAME VARCHAR2(10),
    JOB VARCHAR2(9),
    MGR NUMBER(4),
    HIREDATE DATE,
    SAL NUMBER(7,2),
    COMM NUMBER(7,2),
    DEPTNO NUMBER(4),
    HR_LABEL NUMBER(10))
    TABLESPACE PERF_DATA
    STORAGE (initial 2M
    NEXT 1M
    MINEXTENTS 1
    MAXEXTENTS unlimited)
    PARTITION BY RANGE (hr_label)
    (partition sx1 VALUES LESS THAN (2000) NOLOGGING,
     partition sx2 VALUES LESS THAN (3000),
     partition sx3 VALUES LESS THAN (4000) );
```

Creating Additional Databases After Installation

When you install the *Oracle Database* 11*g* Release 1 (11.1) Enterprise Edition and Oracle Label Security, an initial Oracle Database is created. You can then install Oracle Label Security, as described in the *Oracle Label Security Installation Notes* for your platform.

If you wish to create additional databases, then Oracle recommends that you do this using the Database Configuration Assistant. Alternatively, you can create additional databases by following the steps listed in Chapter 2 of the *Oracle Database Administrator's Guide*

Each time you create a new database, you must install into it the Oracle Label Security data dictionary tables, views, and packages, and create the LBACSYS account. For the first database, this is done automatically when you install Oracle Label Security. For additional databases, you must perform the following tasks manually.

> **Note:** If you have not installed Oracle Label Security at least once in your target Oracle environment, then you must first do so using the Oracle Universal Installer.

1. In your init*sid*.ora file, set the COMPATIBLE parameter to the current *Oracle Database* release that you are running. (This must be no lower than 8.1.7.)

   Shut down and restart your database so that this change will take effect.

2. Connect to the *Oracle Database* instance as user SYS, using the `AS SYSDBA` syntax.

3. Run the script `$ORACLE_HOME/rdbms/admin/catols.sql`.

   This script installs the label-based framework, data dictionary, data types, and packages. After the script is run, the LBACSYS account exists, with the password LBACSYS. All the Oracle Label Security packages exist under this account.

4. Change the default password of the LBACSYS user.

Now, you can proceed to create an Oracle Label Security policy.

See Also: For a complete discussion of Oracle database creation, see *Oracle Database Administrator's Guide*

15

Releasability Using Inverse Groups

This chapter discusses the Oracle Label Security implementation of releasability using inverse groups. It contains the following sections:

Introduction to Inverse Groups and Releasability

Inverse groups indicate *releasability* of information. They are used to mark the dissemination of data. When you add an inverse group to a data label, the data becomes less classified. For example, a user with inverse groups UK and US cannot access data that only has inverse group UK. Adding US to that data makes it accessible to all users with the inverse groups UK and US.

When you assign releasabilities to a user, you mark the communication channel to the user. For data to flow across the communication channel, the data releasabilities must dominate the releasabilities assigned to the user. In other words, releasabilities assigned to a data record must contain all the releasabilities assigned to a user.

The advantage of releasabilities lies in their power to broadly disseminate information. Releasing data to the entire marketing organization becomes as simple as adding the Marketing releasability to the data record.

Comparing Standard Groups and Inverse Groups

Groups in Oracle Label Security identify organizations that own or access data. Like standard groups, inverse groups control the dissemination of information. However, the behavior of inverse groups differs from Oracle Label Security standard group behavior. By default, all policies created in Oracle Label Security use the standard group behavior.

The term, *releasabilities* is sometimes used to refer to the behavior provided by inverse groups. When you include inverse groups in a data label, the effect is similar to assigning label compartment authorizations to a user. When Oracle Label Security evaluates whether a user can view a row of data assigned to a label with inverse groups, it checks to see whether the data, not the user, has the appropriate group authorizations. It checks whether the data has *all* the inverse groups assigned to the user. With standard groups, by contrast, Oracle Label Security checks to see whether a user is authorized for *at least one* of the groups assigned to a row of data.

Consider a policy that contains three standard groups such as, Eastern, Western, and Southern. User1's label authorizations include the groups Eastern and Western. Assuming that User1 has been assigned the appropriate level and compartment authorizations in the policy, then:

- With standard Oracle Label Security groups, User1 can view *all* data records that have the group Eastern, or the group Western, or both Eastern and Western.
- With inverse groups, User1 can only view data records that have, *at a minimum, all* the groups assigned to the user, that is, both Eastern and Western. User1 *cannot* view records that have only the Eastern group, only the Western group, or that have no groups at all.

Table 15–1 shows all the rows that User1 can potentially access, given the type of group that is used in the policy.

Table 15–1 Access to Standard Groups and Inverse Groups

If row label contains groups:	User1 access with standard groups?	User1 access with inverse groups?
None	Y	N
Eastern	Y	N
Western	Y	N
Southern	N	N
Eastern, Western	Y	Y
Eastern, Southern	Y	N
Western, Southern	Y	N
Eastern, Western, Southern	Y	Y

Standard groups indicate *ownership* of information. In this way, all data pertaining to a certain department can have that department's group in the label. When you add a group to a data label, the data becomes more classified. For example, a user with no groups can access data that has no groups in its label. If you add the group US to the data label, the user can no longer access the data.

See Also: "Groups" on page 2-6

How Inverse Groups Work

This section explains how inverse groups are implemented and how they work. It contains these topics:

- Inverse Groups and Hierarchical Structure
- Inverse Groups and User Privileges

Implementing Inverse Groups with the INVERSE_GROUP Enforcement Option

When creating an Oracle Label Security policy, the administrator can specify whether the policy can use inverse group functionality to implement releasability. To do this, the administrator specifies INVERSE_GROUP as one of the *default_options* in the CREATE_POLICY statement.

The INVERSE_GROUP option can be set only at policy creation time. Once a policy is created, this option cannot be changed.

The INVERSE_GROUP option is thus policywide. It cannot be turned on or off when the policy is applied to a table or schema. If you attempt to do so, using the procedure APPLY_TABLE_POLICY or APPLY_SCHEMA_POLICY, then an error will be generated.

While other policy enforcement options can be dropped from a policy, the INVERSE_GROUP policy configuration option cannot be dropped once it is set. To remove the option, you must drop and then re-create the policy.

The administrator can give individual users authorization for one or more inverse groups.

Inverse Groups and Label Components

When an Oracle Label Security policy is created with the inverse group option, the components in the policy label (levels, compartments, and groups) are the same as with standard groups. With inverse groups, however, the user's read groups and write groups have a different meaning and role in data access.

Consider the following policy example, with three levels, one compartment, and three groups:

Table 15–2 Policy Example

Policy Component	Abbreviation
Levels:	
UNCLASSIFIED	UN
CONFIDENTIAL	CON
SECRET	SE
Compartments:	
FINANCIAL	FIN
Groups:	
EASTERN	EAS
WESTERN	WES
SOUTHERN	SOU

Two user labels have been assigned, CON:FIN and SE:FIN:EAS,WES

Two data labels have been assigned, CON:FIN:EAS and SE:FIN:EAS

User access to the data differs, depending on the type of group being used:

- If the policy uses standard groups, then:

  The user with the label CON: FIN *cannot* read CON:FIN:EAS data.

  The user with the label SE:FIN:EAS,WES *can* read SE:FIN:EAS data.

- If the policy has the INVERSE GROUPS policy enforcement option, then:

  The user with the label CON: FIN *can* read CON:FIN:EAS data.

  The user with the label SE:FIN:EAS,WES *cannot* read SE:FIN:EAS data.

Computed Labels with Inverse Groups

This section explains how inverse groups affect computed label values. It contains these topics:

- Computed Session Labels with Inverse Groups
- Inverse Groups and Computed Max Read Groups and Max Write Groups

Computed Session Labels with Inverse Groups

After the administrator assigns label authorizations to a user, Oracle Label Security automatically computes a number of labels. With inverse groups, these labels are as follows:

Table 15–3 Computed Session Labels with Inverse Groups

Computed Label	Definition
Max Read Label	The user's maximum level combined with his or her authorized compartments and the minimum set of inverse groups that should be in the user label (session label)
Max Write Label	The user's maximum level combined with the compartments for which the user has been granted write access. Contains the maximum authorized inverse groups that can be set in any label. The user has write authorizations on all these inverse groups.
Min Write Label	The user's minimum level.
Default Read Label	The default level, combined with compartments and inverse groups that have been designated as default for the user.
Default Write Label	A subset of the default read label, containing the compartments and inverse groups for which the user has been granted write access. However the inverse groups component has no significance as it is the Max Write Groups that is always used for write access.
Default Row Label	The combination of components between the user's minimum write label and the maximum write label, which has been designated as the default for the data label for inserted data. The Inverse groups should be a superset of inverse groups in the default label and a subset of Max Write Groups.

See Also: "Computed Session Labels" on page 3-7

Inverse Groups and Computed Max Read Groups and Max Write Groups

From the computed values in Table 15–3, two sets of groups are identified for label evaluation of read and write access:

Table 15–4 Sets of Groups for Evaluating Read and Write Access

Sets of Groups	Meaning
Max Read Groups	Max Read Groups are the groups contained in the Max Read Label, identifying the *minimum* set of inverse groups that can be set in any user label.
Max Write Groups	Max Write Groups are the groups contained in the Max Write Label, identifying the *maximum* authorized inverse groups that can be set in any user label. This set of groups is checked at the time of write access, and also when setting session labels. Note that Max Write Groups is a superset of Max Read Groups.

As shown in Table 15–5, for standard groups you can have READ ONLY and READ/WRITE authorizations; for inverse groups you can have WRITE ONLY and READ/WRITE authorizations.

Table 15–5 Read and Write Authorizations for Standard Groups and Inverse Groups

Type of Group	READ ONLY	READ/WRITE	WRITE ONLY
Standard Groups	The group is present only in Max Read Label, not in Max Write Label.	The group is present in both Max Read Label and Max Write Label.	Not supported
Inverse Groups	Not supported	The group is present in both Max Read Label and Max Write Label.	The group is present only in Max Write Label, not in Max Read Label.

Although Max Read Groups identifies the set of groups contained in the Max Read Label, this value represents the *minimum* set of inverse groups that can be set. For example:

Max Read Groups: S:C1:G1,G2

Max Write Groups: S:C1:G1,G2,G3,G4,G5

Here, the user can read data that contains at least the two groups listed in Max Read Groups.

Note that in standard groups, there can never be a situation in which there are more groups in the Max Write Label than in the Max Read Label.

Inverse Groups and Hierarchical Structure

Standard groups in Oracle Label Security are hierarchical, so that a group can be associated with a parent group. For example, the EASTERN region can be the parent of two subordinate groups: EAS_SALES, and EAS_HR.

In a policy with standard groups, if the user label has the parent group, then it can access all data of the subordinate groups.

With inverse groups, parent-child relationships are not supported.

Inverse Groups and User Privileges

With inverse groups implemented, the meaning of user privileges remains the same.

When the user has no special privileges, then the read algorithm and the write algorithm are different for standard groups and inverse groups. The differences are described

later, in "Algorithm for Read Access with Inverse Groups" on page 15-6 and "Algorithm for Write Access with Inverse Groups" on page 15-7.

The effect of inverse groups on the COMPACCESS privilege is described later, in "Algorithms for COMPACCESS Privilege with Inverse Groups" on page 15-7.

Inverse groups have no impact upon the following user privileges:

- PROFILE_ACCESS
- WRITEUP
- WRITEDOWN
- WRITEACROSS

Algorithm for Read Access with Inverse Groups

This section describes the algorithm for read access with inverse groups.

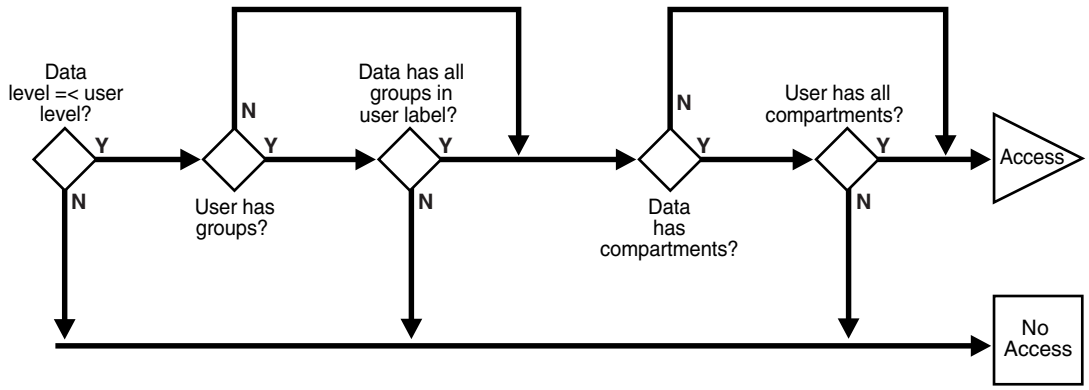
To read data in a table with the INVERSE GROUP option in effect, the label evaluation process proceeds from levels to groups to compartments, as illustrated in Figure 15–1, "Read Access Label Evaluation with Inverse Groups". (Note that the current session label is the label being evaluated.)

1. The user's level must be greater than or equal to the level of data.
2. The user's label must include all the compartments assigned to the data
3. The groups in the data label must be a superset of the groups in the user label.

If the user's label passes these tests, then the user can access the data. If not, the user is denied access. Note that if the data label is null or invalid, then the user is denied access.

Note: This flow diagram is true only when the user has no special privileges.

Figure 15–1 Read Access Label Evaluation with Inverse Groups



See Also: "The Oracle Label Security Algorithm for Read Access" on page 3-9

Algorithm for Write Access with Inverse Groups

This section describes the algorithm for write access with inverse groups.

To write data in a table with the INVERSE GROUP option, the label evaluation process proceeds from levels to groups to compartments, as illustrated in Figure 15–2, "Write Access Label Evaluation with Inverse Groups". (Note that the current session label is the label being evaluated.)

1. The level in the data label must be greater than or equal to the user's minimum level, and less than or equal to the user's session level.
2. One of the following conditions must be met:

   The groups in the data label must be a superset of the groups in the user label.

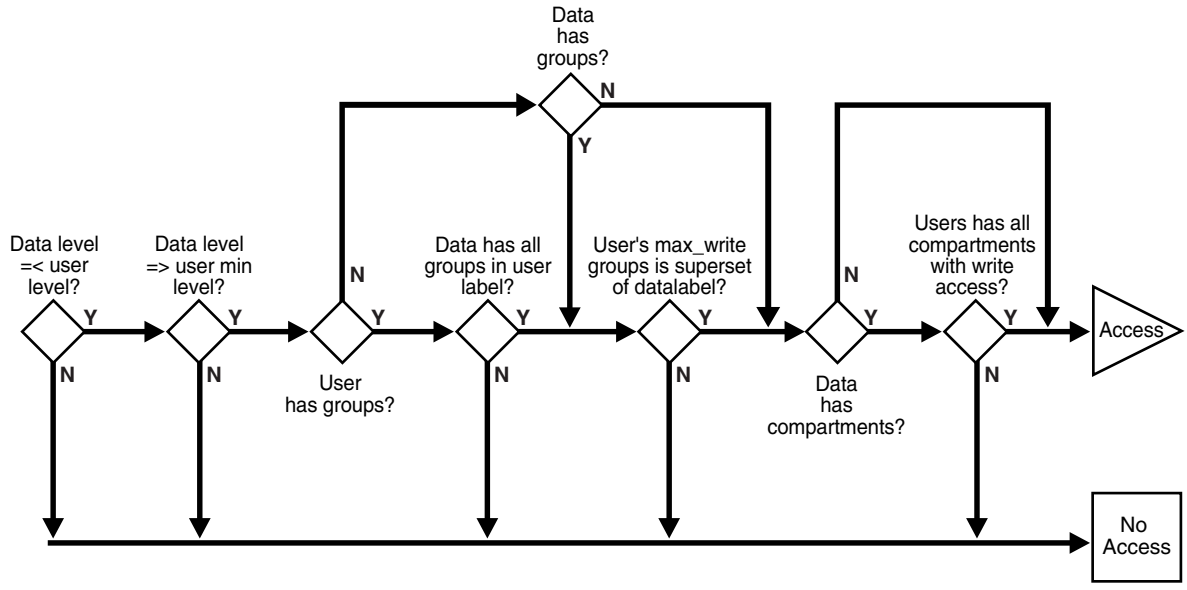
   or

   The user has READ access privilege on the policy.
3. The user's Max Write Groups must be a superset of the data label groups.
4. The user label must have write access on all of the compartments in the data label.

Note that if the data label is null or invalid, then the user is denied access.

Note: This flow diagram is true only when the user has no special privileges.

Figure 15–2 Write Access Label Evaluation with Inverse Groups



See Also: "The Oracle Label Security Algorithm for Write Access" on page 3-10

Algorithms for COMPACCESS Privilege with Inverse Groups

This section describes the algorithms for read and write access with inverse groups, for users who have COMPACCESS privilege.

The COMPACCESS privilege allows a user to access data based on the row's compartments, independent of the row's groups.

- When compartments exist and access to them is authorized, then the group authorization is bypassed.
- If a row has no compartments, then access is determined by the inverse group authorizations.

Figure 15–3, "Read Access Label Evaluation: COMPACCESS Privilege and Inverse Groups" and Figure 15–4, "Write Access Label Evaluation: COMPACCESS Privilege and Inverse Groups" show the label evaluation process for read access and write access for a user with the COMPACCESS privilege. If the data label is null or invalid, then the user is denied access.

(Note that the current session label is the label being evaluated.)

Figure 15–3 Read Access Label Evaluation: COMPACCESS Privilege and Inverse Groups

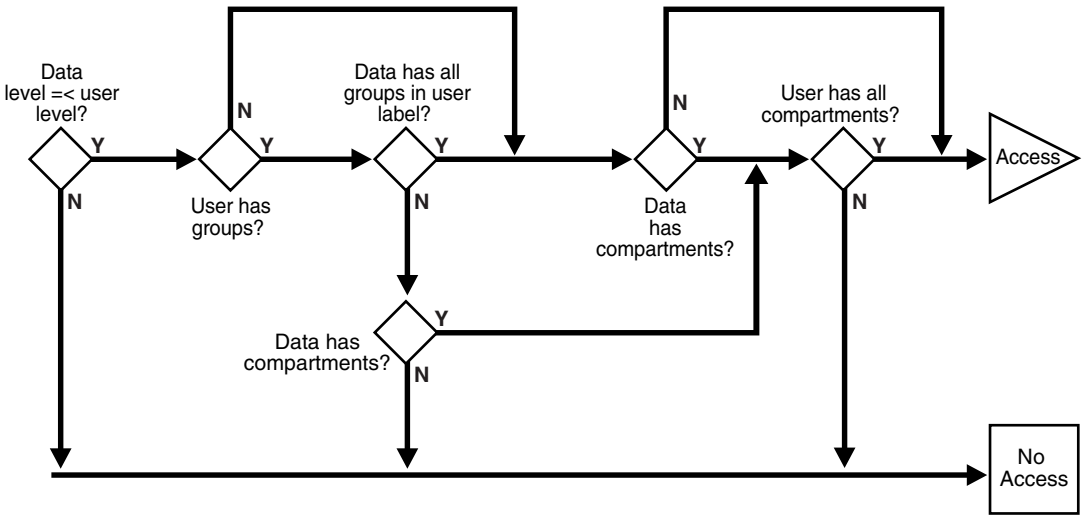
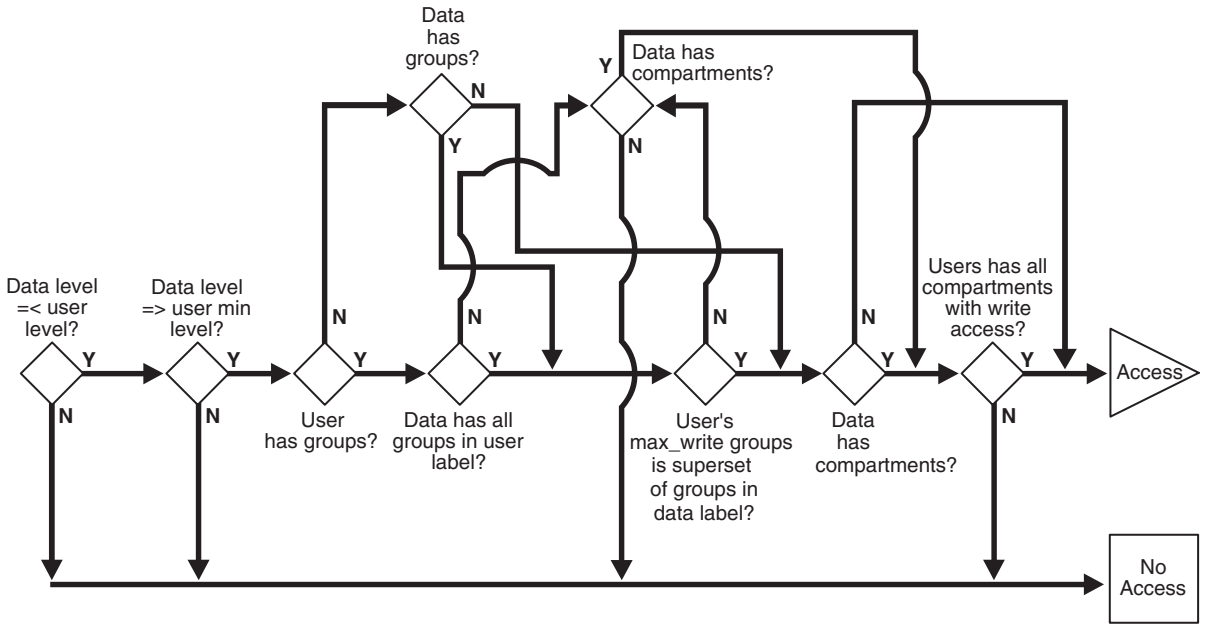


Figure 15–4 Write Access Label Evaluation: COMPACCESS Privilege and Inverse Groups



Session Labels and Inverse Groups

This section describes how inverse groups affect session labels and row labels.

- Setting Initial Session/Row Labels for Standard or Inverse Groups
- Setting Current Session/Row Labels for Standard or Inverse Groups
- Examples of Session Labels and Inverse Groups

Setting Initial Session/Row Labels for Standard or Inverse Groups

The use of inverse groups affects the behavior of Oracle Label Security procedures that determine the session label. The SA_USER_ADMIN.SET_DEFAULT_LABEL and SA_USER_ADMIN.SET_ROW_LABEL procedures set the user's initial session label and row label, respectively, to the one specified.

Standard Groups: Rules for Changing Initial Session/Row Labels

A user's default session label can be changed using SA_USER_ADMIN.SET_DEFAULT_LABEL. In the case of standard groups, the default session label can be set to include any groups in the authorized list, as long as the current default row label will still be dominated by the new write label. That is, the row label will have *the same or fewer standard groups* than the new write label.

The same rule applies for SA_USER_ADMIN.SET_ROW_LABEL.

Inverse Groups: Rules for Changing Initial Session/Row Labels

In the case of inverse groups, the default session label can be set to include any groups in the authorized list, as long as the current default row label will still be dominated by the new write label. That is, the row label will have *the same or more inverse groups* than the new write label.

The same rule applies for SA_USER_ADMIN.SET_ROW_LABEL.

See Also: "SA_USER_ADMIN.SET_DEFAULT_LABEL" on page 8-9

"SA_USER_ADMIN.SET_ROW_LABEL" on page 8-9

"Dominance Rules for Labels with Inverse Groups" on page 15-17

Setting Current Session/Row Labels for Standard or Inverse Groups

The use of inverse groups affects the behavior of the SA_SESSION.SET_LABEL and SA_SESSION.SET_ROW_LABEL procedures, which can be used to set the user's current session label and row label, respectively.

Standard Groups: Rules for Changing Current Session/Row Labels

With standard groups, the SA_SESSION.SET_LABEL procedure can be used to set the session label to include any groups in the user's authorized group list. (Subgroups of authorized groups are implicitly included in the authorized list.) Note that if you change the session label, then this may affect the value of the session's row label.

Use the SET_ROW_LABEL procedure to set the row label value for the current database session. The compartments and groups in the label must be a subset of compartments and groups in the session label to which the user has write access.

Inverse Groups: Rules for Changing Current Session/Row Labels

With inverse groups, the addition of groups to the session label *decreases* a user's ability to access sensitive data with fewer groups. The removal of groups enables the user to access *more* sensitive information. So, the user should be allowed to add groups to the session label, as long as Max Read Groups is a subset of the groups in the session label, and Max Write Groups is a superset of groups in the session label. The same restriction applies when a user removes groups from the session label.

Note that there are no subgroups of authorized groups when using inverse groups. This is because parent groups are not allowed in policies using inverse groups.

Use the SET_ROW_LABEL procedure to set the row label value for the current database session. The compartments in the label must be a subset of compartments in the session label to which the user has write access.

The user is allowed to add inverse groups to the row label, as long as the session label inverse groups are a subset of the row label inverse groups, and Max Write Groups is a superset of inverse groups in the row label.

For example:

- If the user has the inverse groups UK and US as his Max Read Groups, and UK,US,CAN as his Max Write Groups. The user can set his session label to C:ALPHA:UK,US,CAN but not to C:ALPHA:UK.
- If the user has the inverse group UK as his Max Read Groups, and UK,CAN as his Max Write Groups.assigned to him. The user can set the session label to C:ALPHA:UK,CAN but cannot change it to either C:ALPHA or C:ALPHA:UK,US,CAN.

See Also: "Changing the Session Label with SA_SESSION.SET_LABEL" on page 5-14

"Changing the Row Label with SA_SESSION.SET_ROW_LABEL" on page 5-15

Examples of Session Labels and Inverse Groups

This section presents examples to illustrate the use of inverse groups.

Inverse Groups Example 1

Consider a User1, of a policy implementing inverse groups, with the following labels:

Table 15–6 Labels for Inverse Groups Example 1

Name	Definition
Max Read Label	SE:ALPHA,BETA:G1,G2
Max Write Label	SE:ALPHA:G1,G2,G3
Default Read Label	SE:ALPHA,BETA:G1,G2
Default Write Label	SE:ALPHA:G1,G2
Default Row Label	SE:ALPHA:G1,G2
From which the following values are derived:	
Max Read Groups	G1,G2
Max Write Groups	G1,G2,G3

The following conclusions can be drawn:

- User1 can update data with label SE:ALPHA:G1,G2 as well as data with label SE:ALPHA:G1,G2,G3. User1 *cannot*, however, update label SE:ALPHA:G1.

  If standard groups were being used, rather than inverse groups, then User1 could update data with label SE:ALPHA:G1.

- Data that User1 inserts has the label SE:ALPHA:G1,G2. (This is the same as with standard groups.)

- If User1 leaves the default label as is, and sets the row label to SE:ALPHA:G1,G2,G3, then user1 will insert SE:ALPHA:G1,G2,G3 in new rows of data that is written. (In standard groups, User1 can never set more groups in the row label than in the default label.)

Inverse Groups Example 2

Consider a User01, of a policy implementing inverse groups, with the following labels:

Table 15–7 Labels for Inverse Groups Example 2

Name	Definition
Max Read Label	C:ALPHA:
Max Write Label	C:ALPHA:G1,G2,G3
Default Read Label	C:ALPHA:
Default Write Label	C:ALPHA:
Default Row Label	C:ALPHA:
From which the following values are derived:	
Max Read Groups	(an empty set)
Max Write Groups	G1,G2,G3

The following conclusions can be drawn:

- User01 can update any data with level C, compartment ALPHA, and any combination of groups G1, G2, G3, or no groups. User01 inserts the label C:ALPHA: in new data that User01 writes.
- User02, who has Max Read Groups of G1,G2 or G1,G3, and so on, will not be able to view the data written by User01. This is because User01's Default Row Label contains no groups.
- User01 can choose to set inverse groups in the row label, as long as the inverse groups in the session label dominates the row label (that is, User01's session label contains the same or fewer groups than contained in the row label).

  This is true because the row label must have at least the groups in the session label, and can at most have the Maximum Write Groups. If the session label is G1, then you can set the groups in the row label from G1 to the Max Write Groups (G1,G2,G3).
- If User01 sets his session label and row label to C:ALPHA:G1:G2:G3, then his data becomes accessible to anyone who has any combination of G1,G2,G3 in his Max Read Groups.

See Also: "Computed Session Labels" on page 3-7

Changes in Behavior of Procedures with Inverse Groups

When the INVERSE_GROUP option is specified at the time the policy is created, a change occurs in the algorithms that determine the read and write access of the user to labeled data. This section describes how inverse groups affect the behavior of the following procedures:

- SYSDBA.CREATE_POLICY with Inverse Groups
- SYSDBA.ALTER_POLICY with Inverse Groups
- SA_USER_ADMIN.ADD_GROUPS with Inverse Groups
- SA_USER_ADMIN.ALTER_GROUPS with Inverse Groups
- SA_USER_ADMIN.SET_GROUPS with Inverse Groups
- SA_USER_ADMIN.SET_USER_LABELS with Inverse Groups
- SA_USER_ADMIN.SET_DEFAULT_LABEL with Inverse Groups
- SA_USER_ADMIN.SET_ROW_LABEL with Inverse Groups
- SA_COMPONENTS.CREATE_GROUP with Inverse Groups
- SA_COMPONENTS.ALTER_GROUP_PARENT with Inverse Groups
- SA_SESSION.SET_LABEL with Inverse Groups
- SA_SESSION.SET_ROW_LABEL with Inverse Groups
- LEAST_UBOUND with Inverse Groups
- GREATEST_LBOUND with Inverse Groups

SYSDBA.CREATE_POLICY with Inverse Groups

The CREATE_POLICY procedure under the SYSDBA package creates the policy, defines an optional policy-specific column name, and specifies a set of default policy

options. With inverse group support the, user has one more policy enforcement option, INVERSE_GROUP. For example:

```
PROCEDURE CREATE_POLICY (
 HR IN VARCHAR2,
 SA_LABEL IN VARCHAR2 DEFAULT NULL,
```
INVERSE_GROUP IN VARCHAR2 DEFAULT NULL);

> **See Also:** "Creating a Policy with SA_SYSDBA.CREATE_POLICY" on page 7-11
>
> "Overview of Policy Enforcement Options" on page 9-1

SYSDBA.ALTER_POLICY with Inverse Groups

The ALTER_POLICY procedure under the SYSDBA package enables you to change a policy's default enforcement options, *except for* the INVERSE_GROUP option. Once a policy is configured for inverse groups, it cannot be changed.

> **See Also:** "Modifying Policy Options with SA_SYSDBA.ALTER_POLICY" on page 7-12

SA_USER_ADMIN.ADD_GROUPS with Inverse Groups

The ADD_GROUPS procedure adds groups to a user, indicating whether the groups are authorized for write as well as read.

> **See Also:** Syntax for "SA_USER_ADMIN.ADD_GROUPS" on page 8-6.

The type of access authorized depends on the *access_mode* parameter.

Table 15–8 Access Authorized by Values of access_mode Parameter

Access_Mode Parameter	Meaning
READ_WRITE	Indicates that write is authorized. (That is, the group is contained in both Max Read Groups and Max Write Groups.)
WRITE_ONLY	Indicates that the group is contained in Max Write Groups and not in Max Read Groups
access_mode	If access_mode is set to READ_WRITE, then the group is added to both Max Read Groups and Max Write Groups. If access_mode is set to SA_UTL.WRITE_ONLY, then the group is added only to the Max Write Groups. If access_mode is NULL, then it is set to SA_UTL.READ_WRITE.
in_def	Specifies whether these groups should be in the default groups (Y/N). If in_def is NULL, then it will be set to Y or N as follows: If access mode is READ_WRITE, in_def is set to Y. If access mode is WRITE_ONLY, in_def is set to N.
in_row	Specifies whether these groups should be in the row label (Y/N), using the identical criteria as for in_def. However, if in_def is Y, then in_row must also be Y.

Note that if in_def is Y in a row, then in_row must also be set to Y, but not the other way round.

The same is the case with the in_row field.

> **See Also:** "Inverse Groups and Computed Max Read Groups and Max Write Groups" on page 15-4

SA_USER_ADMIN.ALTER_GROUPS with Inverse Groups

The ALTER_GROUPS procedure changes the write access, the default label indicator, and the row label indicator for each of the groups in the list.

The behavior of inverse groups is the same as described in the case of ADD_GROUPS.

> **See Also:** Syntax for "SA_USER_ADMIN.ALTER_GROUPS" on page 8-6.

SA_USER_ADMIN.SET_GROUPS with Inverse Groups

The SET_GROUPS procedure assigns groups to a user and identifies default values for the user's session label and row label.

> **See Also:** Syntax for "SA_USER_ADMIN.SET_GROUPS" on page 8-3.

Inverse groups are handled differently than standard groups, as follows:

Table 15–9 Assigning Groups to a User

Group Set Name	Meaning
read_groups	A comma-delimited list of groups that would be Max Read Groups
write_groups	A comma-delimited list of groups that would be Max Write Groups. It must be a superset of read_groups. If write_groups is NULL, then they are set to read_groups.
def_groups	Specifies the default groups. It should at least have *read_groups*, and *write_groups* should be a superset of *def_groups*. If def_groups is NULL, then they are set to the read_groups.
row_groups	Specifies the row groups. It should at least have the *def_groups* and should be a subset of max write groups. If row_groups is NULL, then they are set to the def_groups, because for inverse groups, all def_groups are also in write_ groups.

SA_USER_ADMIN.SET_USER_LABELS with Inverse Groups

The SET_USER_LABELS procedure sets the user's levels, compartments, and groups using a set of labels, instead of the individual components.

> **See Also:** Syntax for "SA_USER_ADMIN.SET_USER_LABELS" on page 8-8.

Inverse groups are handled differently than standard groups, as follows:

Table 15–10 Inverse Group Label Definitions

Name	Definition
max_read_label	Specifies the label string to be used to initialize the user's maximum authorized read label. Composed of the user's maximum level, compartments authorized for read access, and if inverse groups, minimum set of groups that can be set in any label.(Max Read Groups)
max_write_label	Specifies the label string to be used to initialize the user's maximum authorized write label. Composed of the user's maximum level, compartments authorized for write access, and if inverse groups, the maximum authorized groups that can be set in any label (Max Write Groups). All the inverse groups in this have write authorization also. It should be a superset of groups in max_read_label. If max_write_label is not specified, then it is set to max_read_label.
def_label	Specifies the label string to be used to initialize the user's session label, including level, compartments, and groups (a subset of max_read_label). If default_label is not specified, then it is set to max_read_label. For inverse groups, component it should at least have the groups in max_read_label, and groups in max_write_label should be a superset of the groups in the def_label.
row_label	Specifies the label string to be used to initialize the program's row label. Includes levels, compartments, and groups: subsets of max_write_label and def_label. If row_label is not specified, then it is set to def_label, with only the compartments and groups authorized for write access. The inverse groups component is set to the same as that in def_label if the row_ label is not specified. The inverse groups in row label should at least be those in default label and should be a subset of Max Write Groups.

SA_USER_ADMIN.SET_DEFAULT_LABEL with Inverse Groups

The SET_DEFAULT_LABEL procedure sets the user's initial session label to the one specified.

All the rules mentioned for setting inverse groups component of session label mentioned in "Session Labels and Inverse Groups" are applicable here.

> **See Also:** Syntax for "SA_USER_ADMIN.SET_DEFAULT_ LABEL" on page 8-9.

SA_USER_ADMIN.SET_ROW_LABEL with Inverse Groups

Use the SET_ROW_LABEL procedure to set the user's initial row label to the one specified.

> **See Also:** Syntax for "SA_USER_ADMIN.SET_ROW_LABEL" on page 8-9.

When specifying the row_label, the inverse groups component must contain at least all the inverse groups in def_label and should be a subset of Max Write Groups.

> **See Also:** "Setting Initial Session/Row Labels for Standard or Inverse Groups" on page 15-9

SA_COMPONENTS.CREATE_GROUP with Inverse Groups

Use the CREATE_GROUP procedure to create a group and specify its short name and long name, and optionally a parent group.

> **See Also:** Syntax for "Creating a Group with SA_COMPONENTS.CREATE_GROUP" on page 7-17.

With inverse groups, the parent_name field should always be NULL. If the user specifies a value for this field, then an error message is displayed, indicating that the group hierarchy is disabled.

SA_COMPONENTS.ALTER_GROUP_PARENT with Inverse Groups

This function is disabled for policies with the inverse group option. An error message is displayed if the user calls this function.

> **See Also:** Syntax for "Modifying a Group with SA_COMPONENTS.ALTER_GROUP" on page 7-18.

SA_SESSION.SET_LABEL with Inverse Groups

Use the SET_LABEL procedure to set the label of the current database session.

> **See Also:** Syntax for "Changing the Session Label with SA_SESSION.SET_LABEL" on page 5-14.

For the current user, this procedure follows the same rules for setting the session label as does the sa_user_admin.set_user_label function.

> **See Also:** "Setting Current Session/Row Labels for Standard or Inverse Groups" on page 15-10

SA_SESSION.SET_ROW_LABEL with Inverse Groups

Use the SET_ROW_LABEL procedure to set the default row label value for the current database session.

> **See Also:** Syntax for "Changing the Row Label with SA_SESSION.SET_ROW_LABEL" on page 5-15.

For the current user, this procedure follows the same rules for setting the row label as does the sa_user_admin.set_row_label function.

> **See Also:** "Setting Initial Session/Row Labels for Standard or Inverse Groups" on page 15-9

LEAST_UBOUND with Inverse Groups

The LEAST_UBOUND (LUBD) function returns a character string label that is the least upper bound of label1 and label2 that is, the one label that dominates both.

With *standard* groups, the least upper bound is the highest level, the union of the compartments in the labels, and t*he union of the groups* in the labels.

With *inverse* groups, the least upper bound is the highest level, the union of the compartments in the labels, and *the intersection of the inverse groups* in the labels.

For example, with inverse groups, the least upper bound of HIGHLY_SENSITIVE:ALPHA:G1,G2 and SENSITIVE:BETA:G1 is HIGHLY_SENSITIVE:ALPHA,BETA:G1

GREATEST_LBOUND with Inverse Groups

The GREATEST_LBOUND (GLBD) function can be used to determine the lowest label of the data that can be involved in an operation, given two different labels. It returns a character string label that is the greatest lower bound of label1 and label2.

With *standard* groups, the greatest lower bound is the lowest level, and the *intersection of the compartments in the labels and the groups* in the labels.

With *inverse* groups, the greatest lower bound is the lowest level, and the *intersection of the compartments in the labels and the union of inverse groups* in the labels.

For example, with inverse groups the greatest lower bound of HIGHLY_SENSITIVE:ALPHA:G1,G3 and SENSITIVE::G1 is SENSITIVE:G1,G3

See: "Determining Upper and Lower Bounds of Labels" on page 5-9

Dominance Rules for Labels with Inverse Groups

Dominance rules for Oracle Label Security with standard groups can be summarized as follows:

A user label dominates a data label if:

- User level is greater than or equal to the data level
- User compartments are a superset of the data compartments
- User groups intersects (have at least one group from) the data groups

Dominance rules for Oracle Label Security with inverse groups can be summarized as follows:

A user label dominates a data label if:

- User level is greater than or equal to the data level
- User compartments are a superset of the data compartments
- Data groups are a superset of user groups

See Also: "Dominant and Dominated Labels" on page A-1

Part IV

Appendixes

This part contains the following chapter:

- Appendix A, "Advanced Topics in Oracle Label Security"
- Appendix B, "Command-line Tools for Label Security Using Oracle Internet Directory"
- Appendix C, "Oracle Label Security in an RAC Environment"
- Appendix D, "Frequently Asked Questions on Oracle Label Security"
- Appendix E, "Reference"

A

Advanced Topics in Oracle Label Security

This appendix covers topics of interest to advanced users of Oracle Label Security. It contains these sections:

- Analyzing the Relationships Between Labels
- OCI Interface for Setting Session Labels

Analyzing the Relationships Between Labels

This section describes relationships between labels. It contains these topics:

- Dominant and Dominated Labels
- Non-Comparable Labels
- Using Dominance Functions

Dominant and Dominated Labels

The relationship between two labels can be described in terms of *dominance.* A user's ability to access an object depends on whether the user's label dominates the label of the object. If a user's label does not dominate the object's label, then the user is not allowed to access the object.

Label dominance is analyzed in terms of all its components: levels, compartments, and groups.

Table A–1 Dominance in the Comparison of Labels

Factor	Criteria for Dominance
Level	For *label1* to dominate *label2*, the level of *label1* must be greater than or equal to that of *label2*.
Compartment	For *label1* to dominate *label2*, the compartments of *label1* must contain *all* the compartments of *label2*.
Group	For *label1* to dominate *label2*, *label1* must contain *at least one* of the groups of *label2*.

One label *dominates* another label if all of its components dominate the components of the other label. For example, the label HIGHLY_SENSITIVE:FINANCE,OPERATIONS dominates the label HIGHLY_SENSITIVE:FINANCE. Similarly, the label HIGHLY_SENSITIVE::WR_AP dominates the label HIGHLY_SENSITIVE::WR_AP, WR_AR.

> **See Also:** "Dominance Rules for Labels with Inverse Groups" on page 15-17

Non-Comparable Labels

The relationship between two labels cannot always be defined by dominance. Two labels are *non-comparable* if neither label dominates the other. If any compartments differ between the two labels (as with HS:A and HS:B), then they are non-comparable. Similarly, the labels HS:A and S:B are non-comparable.

Using Dominance Functions

You can use dominance functions to specify ranges in queries. The following functions enable you to indicate dominance relationships between specified labels.

Table A–2 Functions to Determine Dominance

Function	Meaning
STRICTLY_DOMINATES	The value of *label1* dominates that of *label2*, and is not equal to it.
DOMINATES	The value of *label1* dominates, or is equal to, that of *label2*.
DOMINATED_BY	The value of *label1* is dominated by that of *label2*.
STRICTLY_DOMINATED_BY	The value of *label1* is dominated by that of *label2*, and is not equal to it.

Note that there are two types of dominance function. While the SA_UTL dominance functions return BOOLEAN values, the standalone dominance functions return integers.

- The DOMINATES Standalone Function
- The STRICTLY_DOMINATES Standalone Function
- The DOMINATED_BY Standalone Function
- The STRICTLY_DOMINATED_BY Standalone Function
- SA_UTL.DOMINATES
- SA_UTL.STRICTLY_DOMINATES
- SA_UTL.DOMINATED_BY
- SA_UTL.STRICTLY_DOMINATED_BY

See Also: "Ordering Labeled Data Rows" on page 5-8

The DOMINATES Standalone Function

The DOMINATES (DOM) function returns 1 (TRUE) if *label1* dominates *label2*, or 0 (FALSE) if it does not.

Syntax:

```
FUNCTION DOMINATES (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN INTEGER;
```

The STRICTLY_DOMINATES Standalone Function

The STRICTLY_DOMINATES (SDOM) function returns 1 (TRUE) if *label1* dominates *label2* and is not equal to it.

Syntax:

```
FUNCTION STRICTLY_DOMINATES (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN INTEGER;
```

The DOMINATED_BY Standalone Function

The DOMINATED_BY (DOM_BY) function returns 1 (TRUE) if *label1* is dominated by *label2*.

Syntax:

```
FUNCTION DOMINATED_BY (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN INTEGER;
```

The STRICTLY_DOMINATED_BY Standalone Function

The STRICTLY_DOMINATED_BY (SDOM_BY) function returns 1 (TRUE) if *label1* is dominated by *label2* and is not equal to it.

Syntax:

```
FUNCTION STRICTLY_DOMINATED_BY (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN INTEGER;
```

SA_UTL.DOMINATES

The SA_UTL.DOMINATES function returns TRUE if *label1* dominates *label2*.

Syntax:

```
FUNCTION DOMINATES (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN BOOLEAN;
```

SA_UTL.STRICTLY_DOMINATES

The SA_UTL.STRICTLY_DOMINATES function returns TRUE if *label1* dominates *label2* and is not equal to it.

Syntax:

```
FUNCTION STRICTLY_DOMINATES (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN BOOLEAN;
```

SA_UTL.DOMINATED_BY

The SA_UTL.DOMINATED_BY function returns TRUE if *label1* is dominated by *label2*.

Syntax:

```
FUNCTION DOMINATED_BY (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN BOOLEAN;
```

SA_UTL.STRICTLY_DOMINATED_BY

The SA_UTL.STRICTLY_DOMINATED_BY function returns TRUE if *label1* is dominated by *label2* and is not equal to it.

Syntax:

```
FUNCTION STRICTLY_DOMINATED_BY (
  label1          IN NUMBER,
  label2          IN NUMBER)
RETURN BOOLEAN;
```

See Also:: "Determining Upper and Lower Bounds of Labels" on page 5-9.

OCI Interface for Setting Session Labels

When using Oracle Call Interface (OCI) to connect, the policy's SYS_CONTEXT variables can be used to initialize the session label and the row label. The variables are set using the OCIAttrSet function to initialize *externally initialized* SYS_CONTEXT variables. These are available in Release 8.1.7 only when Oracle Label Security is installed.

Each policy has a SYS_CONTEXT named SA$*policy_name*_X. There are two variables that can be set, INITIAL_LABEL and INITIAL_ROW_LABEL.

When set to valid labels within the user's authorizations, the new values will be used instead of the default values stored for the user. This is the same mechanism used for remote connections

See Also: Chapter 13, "Using Oracle Label Security with a Distributed Database"

OCIAttrSet

Additional attributes are defined for OCIAttrSet to insert context. Use OCI_ATTR_APPCTX_SIZE to initialize the context array size with the desired number of context attributes:

```
OCIAttrSet(session, OCI_HTYPE_SESSION,
                (dvoid *)&size, (ub4)0, OCI_ATTR_APPCTX_SIZE, error_handle);
```

Note that size is ub4 type.

OCIAttrGet

Then call OCIAttrGet with OCI_ATTR_APPCTX_LIST to get a handle on the application context list descriptor for the session:

```
OCIAttrGet(session, OCI_HTYPE_SESSION,
```

```
                    (dvoid *)&ctxl_desc, (ub4)0, OCI_ATTR_APPCTX_LIST, error_handle);
```

Note that ctxl_desc is (OCIParam *) type.

OCIParamGet

Then use the application context list descriptor to obtain an individual descriptor for the i-th application context:

```
OCIParamGet(ctxl_desc, OCI_DTYPE_PARAM, error_handle,(dvoid **)&ctx_desc, i);
```

Note that ctx_desc is (OCIParam *) type.

OCIAttrSet

Set the proper values in the application context by using the three new attributes OCI_ATTR_APPCTX_NAME, OCI_ATTR_APPCTX_ATTR, and OCI_ATTR_APPCTX_VALUE:

```
OCIAttrSet(ctx_desc, OCI_DTYPE_PARAM,
                 (dvoid *)ctx_name, sizeof(ctx_name), OCI_ATTR_APPCTX_NAME,
                 error_handle);

OCIAttrSet(ctx_desc, OCI_DTYPE_PARAM,
                 (dvoid *)attr_name, sizeof(attr_name), OCI_ATTR_APPCTX_ATTR,
                 error_handle);

OCIAttrSet(ctx_desc, OCI_DTYPE_PARAM,
                 (dvoid *)value, sizeof(value), OCI_ATTR_APPCTX_VALUE,
                 error_handle);
```

Note that only character type is supported, because application context operations are based on the VARCHAR2 type.

OCI Example

The following example shows how to use externalized SYS_CONTEXT with Oracle Label Security.

```
#ifdef RCSID
static char *RCSid =
   "$Header: ext_mls.c 09-may-00.10:07:08 jdoe Exp $ ";
#endif /* RCSID */

/* Copyright (c) Oracle Corporation 1999, 2000. All Rights Reserved. */

/*

   NAME
ext_mls.c - externalized SYS_CONTEXT with Label Security

   DESCRIPTION
Run olsdemo.sql script before executing this example.
Usage: <executable obtained with .c file> <user_name> <password>
<session-initial-label
Example: avg_sal sa_demo sa_demo L3:M,E:D10

   PUBLIC FUNCTION(S)
<list of external functions declared/defined - with one-line descriptions>
```

```
   PRIVATE FUNCTION(S)
<list of static functions defined in .c file - with one-line descriptions>

   RETURNS
The average salary in the EMP table of the SA_DEMO schema querying as the
specified user with the specified session label.

   NOTES
<other useful comments, qualifications, and so on>

   MODIFIED   (MM/DD/YY)
jlev      09/18/03 - cleanup
jdoe      05/09/00 - cleanup
   jdoe      10/13/99 - standalone OCI program to test MLS SYS_CONTEXT
   jdoe      10/13/99 - Creation

*/
#include <stdio.h>
#include <stdlib.h>
#include <string.h>
#include <oci.h>

static OCIEnv *envhp;
static OCIError *errhp;

int main(/*_ int argc, char *argv[] _*/);

/* get and print error */
static void checkerr(/*_OCIError *errhp, sword status _*/);
/* print error */
static void printerr(char *call);
static sword status;

/* return the average of employees' salary */
static CONST text *const selectstmt = (text *)
     "select avg(sal) from sa_demo.emp";

int main(argc, argv)
int argc;
char *argv[];
{
  OCISession *authp = (OCISession *) 0;
  OCIServer *srvhp;
  OCISvcCtx *svchp;
  OCIDefine *defnp = (OCIDefine *) 0;
  dvoid *parmdp;
  ub4 ctxsize;
  OCIParam *ctxldesc;
  OCIParam *ctxedesc;
  OCIStmt *stmtp = (OCIStmt *) 0;
  ub4 avg_sal = 0;
  sword status;

  if (OCIInitialize((ub4) OCI_DEFAULT, (dvoid *) 0,
                    (dvoid * (*)(dvoid *, size_t)) 0,
                    (dvoid * (*)(dvoid *, dvoid *, size_t)) 0,
                    (void (*)(dvoid *, dvoid *)) 0))
    printerr("OCIInitialize");

  if (OCIEnvInit((OCIEnv **) &envhp, OCI_DEFAULT, (size_t) 0, (dvoid **) 0))
```

```
  printerr("OCIEnvInit");

if (OCIHandleAlloc((dvoid *) envhp, (dvoid **) &errhp, OCI_HTYPE_ERROR,
                   (size_t) 0, (dvoid **) 0))
  printerr("OCIHandleAlloc:OCI_HTYPE_ERROR");

if (OCIHandleAlloc((dvoid *) envhp, (dvoid **) &srvhp, OCI_HTYPE_SERVER,
                   (size_t) 0, (dvoid **) 0))
  printerr("OCIHandleAlloc:OCI_HTYPE_SERVER");

if (OCIHandleAlloc((dvoid *) envhp, (dvoid **) &svchp, OCI_HTYPE_SVCCTX,
                   (size_t) 0, (dvoid **) 0))
  printerr("OCIHandleAlloc:OCI_HTYPE_SVCCTX");

if (OCIServerAttach(srvhp, errhp, (text *) "", strlen(""), 0))
  printerr("OCIServerAttach");

/* set attribute server context in the service context */
if (OCIAttrSet((dvoid *) svchp, OCI_HTYPE_SVCCTX, (dvoid *) srvhp,
               (ub4) 0, OCI_ATTR_SERVER, (OCIError *) errhp))
  printerr("OCIAttrSet:OCI_HTYPE_SVCCTX");

if (OCIHandleAlloc((dvoid *) envhp, (dvoid **) &authp,
                   (ub4) OCI_HTYPE_SESSION, (size_t) 0, (dvoid **) 0))
  printerr("OCIHandleAlloc:OCI_HTYPE_SESSION");

/* set application context to 1 */
ctxsize = 1;

/* set up app ctx buffer */
if (OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION, (dvoid *) &ctxsize,
               (ub4) 0, (ub4) OCI_ATTR_APPCTX_SIZE, errhp))
  printerr("OCIAttrSet:OCI_ATTR_APPCTX_SIZE");

/* retrieve the list descriptor */
if (OCIAttrGet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION,
               (dvoid *) &ctxldesc, 0, OCI_ATTR_APPCTX_LIST, errhp))
  printerr("OCIAttrGet:OCI_ATTR_APPCTX_LIST");

if (status = OCIParamGet(ctxldesc, OCI_DTYPE_PARAM, errhp,
                         (dvoid **) &ctxedesc, 1))
  {
    if (status == OCI_NO_DATA)
      {
        printf("No Data found!\n");
        exit(1);
      }
  }

/* set context namespace to SA$<pol_name>_X */
if (OCIAttrSet((dvoid *) ctxedesc, (ub4) OCI_DTYPE_PARAM,
               (dvoid *) "SA$HUMAN_RESOURCES_X",
               (ub4) strlen((char *) "SA$HUMAN_RESOURCES_X"),
               (ub4) OCI_ATTR_APPCTX_NAME, errhp))
  printerr("OCIAttrSet:OCI_ATTR_APPCTX_NAME:SA$HUMAN_RESOURCES_X");

/* set context attribute to INITIAL_LABEL */
if (OCIAttrSet((dvoid *) ctxedesc, (ub4) OCI_DTYPE_PARAM,
               (dvoid *) "INITIAL_LABEL",
               (ub4) strlen((char *) "INITIAL_LABEL"),
```

```
                (ub4) OCI_ATTR_APPCTX_ATTR, errhp))
    printerr("OCIAttrSet:OCI_DTYPE_PARAM:INITIAL_LABEL");

  /* set context value to argv[3] - initial label */
  if (OCIAttrSet((dvoid *) ctxedesc, (ub4) OCI_DTYPE_PARAM,
                (dvoid *) argv[3],
                (ub4) strlen((char *) argv[3]),
                (ub4) OCI_ATTR_APPCTX_VALUE, errhp))
    printerr("OCIAttrSet:argv[3]");

  /* username first command line argument */
  if (OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION, (dvoid *) argv[1],
                (ub4) strlen((char *) argv[1]), (ub4) OCI_ATTR_USERNAME,
                errhp))
    printerr("OCIAttrSet:username");

  /* password second command line argument */
  if (OCIAttrSet((dvoid *) authp, (ub4) OCI_HTYPE_SESSION, (dvoid *) argv[2],
                (ub4) strlen((char *) argv[2]), (ub4) OCI_ATTR_PASSWORD,
                errhp))
    printerr("OCIAttrSet:password");

  if (OCISessionBegin(svchp, errhp, authp, OCI_CRED_RDBMS, (ub4) OCI_DEFAULT))
    printerr("OCISessionBegin");

  if (OCIAttrSet((dvoid *) svchp, (ub4) OCI_HTYPE_SVCCTX, (dvoid *) authp,
                (ub4) 0, (ub4) OCI_ATTR_SESSION, errhp))
    printerr("OCIAttrSet:OCI_ATTR_SESSION");

  if (OCIHandleAlloc((dvoid *) envhp, (dvoid **) &stmtp, OCI_HTYPE_STMT,
                    0, 0))
    printerr("OCIHandleAlloc:OCI_HTYPE_STMT");

  if (OCIStmtPrepare(stmtp, errhp, (CONST OraText *) selectstmt,
                    (ub4) strlen((const char *) selectstmt),
                    (ub4) OCI_NTV_SYNTAX, (ub4) OCI_DEFAULT))
    printerr("OCIStmtPrepare");

  if (OCIDefineByPos(stmtp, &defnp, errhp, (ub4) 1, (dvoid *) &avg_sal,
                    (sb4) sizeof(avg_sal), SQLT_INT, 0, 0, 0, OCI_DEFAULT))
    printerr("OCIDefineByPos");

  if (status = OCIStmtExecute(svchp, stmtp, errhp, 1, 0, NULL, NULL,
                              OCI_DEFAULT))
    {
      if (status == OCI_NO_DATA)
        {
          printf("No Data found!\n");
          exit(1);
        }
    }

  if (OCISessionEnd(svchp, errhp, authp, OCI_DEFAULT))
    printerr("OCISessionEnd");

  printf("average salary is: %d\n", avg_sal);
}

void checkerr(errhp, status)
     OCIError *errhp;
```

```
     sword status;
{
  text errbuf[512];
  sb4 errcode = 0;

  switch (status)
    {
    case OCI_ERROR:
      (void) OCIErrorGet((dvoid *) errhp, 1, NULL, &errcode, errbuf,
                         (ub4) sizeof(errbuf), OCI_HTYPE_ERROR);
      printf("Error - %.*s\n", 512, errbuf);
      break;
    default:
      break;
    }
}

void printerr(call)
     char *call;
{
  printf("Error: %s\n", call);
}
/* end of file ext_mls.c */
```

B

Command-line Tools for Label Security Using Oracle Internet Directory

When Oracle Label Security is used with Oracle Internet Directory, security administrators can use certain commands to create and alter label security attributes stored in the directory.

> **Note:** Starting this release, you can also use the graphical user interface provided by Oracle Enterprise Manager Database Control or Grid Control to manage Oracle Label Security. Detailed documentation can be found in Oracle Enterprise Manager help.

This Appendix describes these commands and the parameters they require. They perform updates, inserts and deletes of entries in the directory and are implemented through a script named *olsadmintool*, which you call from `$ORACLE_HOME/bin/olsadmintool`. This Appendix contains the following sections and tables:

- Table B–1, " Oracle Label Security Commands in Categories" lists all the commands, in categories, with links to their explanations. Some of these commands replace PL/SQL procedures (indicated in Table B–2, " olsadmintool Commands Linked to Their Explanations") that are used for the indicated purposes when Oracle Label Security is used without Oracle Internet Directory. Sites already using Oracle Label Security that add Oracle Internet Directory must replace the use of those PL/SQL procedures by switching to use these new commands instead.
- Table B–2, " olsadmintool Commands Linked to Their Explanations" then lists the commands alphabetically, with links to their explanations.
- Command Explanations, after Table B–2, " olsadmintool Commands Linked to Their Explanations", provides the individual explanations and examples of the commands and their parameters, in alphabetic order.
- Relating Parameters to Commands for olsadmintool follows Table B–2, " olsadmintool Commands Linked to Their Explanations" with Summaries in Table B–3, " Summary: olsadmintool Command Parameters" and Table B–4, " Summary of Profile and Default Command Parameters". These tables present summaries of the commands' use of parameters by listing the commands and their parameters in tabular format, enabling you to see patterns of parameter usage.
- Table B–3, " Summary: olsadmintool Command Parameters" gives a detailed explanation for each parameter, in alphabetic order and lists the commands in which it is used.

- Examples of Using olsadmintool shows typical uses of the tool and the results of the specific examples shown.

Table B–1 Oracle Label Security Commands in Categories

Command Category	Purpose of Command	Command	Replaces PL/SQL Statement
Policies	Create Policy	olsadmintool createpolicy	SA_SYSDBA.CREATE_POLICY
	Alter a Level	olsadmintool alterpolicy	SA_SYSDBA.ALTER_POLICY
	Drop a Policy	olsadmintool droppolicy	SA_SYSDBA.DROP_POLICY
	Add Policy Creator	olsadmintool addpolcreator	None; new
	Drop Policy Creator	olsadmintool droppolcreator	None; new
Levels in a Policy	Create a Level	olsadmintool createlevel	SA_COMPONENTS.CREATE_LEVEL
	Alter a Level	olsadmintool alterlevel	SA_COMPONENTS.ALTER_LEVEL
	Drop a Level	olsadmintool droplevel	SA_COMPONENTS.DROP_LEVEL
Groups in a Policy	Create a Group	olsadmintool creategroup	SA_COMPONENTS.CREATE_GROUP
	Alter a Group	olsadmintool altergroup	SA_COMPONENTS.ALTER_GROUP
	(also a group parent)	olsadmintool altergroupparent	SA_COMPONENTS.ALTER_GROUP_PARENT
	Drop a Group	olsadmintool dropgroup	SA_COMPONENTS.DROP_GROUP
Compartments in a Policy	Create a Compartment	olsadmintool createcompartment	SA_COMPONENTS.CREATE_COMPARTMENT
	Alter a Compartment	olsadmintool altercompartment	SA_COMPONENTS.ALTER_COMPARTMENT
	Drop a Compartment	olsadmintool dropcompartment	SA_COMPONENTS.DROP_COMPARTMENT
Data Labels	Create a Label	olsadmintool createlabel	SA_LABEL_ADMIN.CREATE_LABEL
	Alter a Label	olsadmintool alterlabel	SA_LABEL_ADMIN.ALTER_LABEL
	Drop a Label	olsadmintool droplabel	SA_LABEL_ADMIN.DROP_LABEL
Users	Add a User to a Profile	olsadmintool adduser	None; new
	Drop a User	olsadmintool dropuser	SA_USER_ADMIN.DROP_USER_ACCESS
Profiles	Create a Profile	olsadmintool createprofile	Replaces the use of several methods. [1]
	List Profiles	olsadmintool listprofile	None; new
	Describe a Profile	olsadmintool describeprofile	None; new
	Drop a Profile	olsadmintool dropprofile	None; new

Table B–1 (Cont.) Oracle Label Security Commands in Categories

Command Category	Purpose of Command	Command	Replaces PL/SQL Statement
Policy Administrators	Drop Policy Administrator	olsadmintool addadmin	None; new
	Drop Policy Administrator	olsadmintool dropadmin	None; new
Policy Access			
	Set Audit Options	olsadmintool addpolaccess	None; new
	Relating Parameters to Commands for olsadmintool	olsadmintool droppolaccess	None; new
Auditing			
	Set Audit Options	olsadmintool audit	SA_AUDIT_ADMIN.AUDIT
		olsadmintool noaudit	SA_AUDIT_ADMIN.NOAUDIT
Help			
	Get Help for olsadmintool	olsadmintool command --help	None; new

[1] Replaces several methods in SA_USER_ADMIN: SET_LEVELS, SET_USER_PRIVILEGES, and SET_DEFAULT_LABEL

Table B–2 olsadmintool Commands Linked to Their Explanations

Purpose of Command (Links in Alphabetical Order)	Command
Add a User to a Profile	olsadmintool adduser
Add Policy Administrators	olsadmintool addadmin
Add Policy Creator	olsadmintool addpolcreator
Alter a Compartment	olsadmintool altercompartment
Alter a Group	olsadmintool altergroup
Alter a Group's Parent	olsadmintool altergroupparent
Alter a Label	olsadmintool alterlabel
Alter a Level	olsadmintool alterlevel
Alter a Level	olsadmintool alterpolicy
Cancel Audit Options	olsadmintool noaudit
Create a Compartment	olsadmintool createcompartment
Create a Group	olsadmintool creategroup
Create a Label	olsadmintool createlabel
Create a Level	olsadmintool createlevel
Create a Profile	olsadmintool createprofile
Create Policy	olsadmintool createpolicy
Describe a Profile	olsadmintool describeprofile

Table B–2 (Cont.) olsadmintool Commands Linked to Their Explanations

Purpose of Command (Links in Alphabetical Order)	Command
Drop a Compartment	olsadmintool dropcompartment
Drop a Group	olsadmintool dropgroup
Drop a Label	olsadmintool droplabel
Drop a Level	olsadmintool droplevel
Drop a Policy	olsadmintool droppolicy
Drop a Profile	olsadmintool dropprofile
Drop a User	olsadmintool dropuser
Drop Policy Administrator	olsadmintool dropadmin
Drop Policy Creator	olsadmintool droppolcreator
Get Help for an olsadmintool Command	olsadmintool <*command name*> --help
List Profiles	olsadmintool listprofile
Set Audit Options	olsadmintool audit

Command Explanations

In the command explanations that follow, some parameters are optional, which is indicated by enclosing such a parameter within brackets. The two most common examples are `[ -b <admin context> ]` and `[-p <port>]`, indicating that it is optional to specify either the administrative context for the command or the port through which to connect to Oracle Internet Directory. (Default port is 389.)

The use of two dashes (`--`, no space) is required for all parameters other than `b`, `h`, `p`, `D`, and `w`, which are preceded by a single dash. The double dash indicates the need to specify the full or long version of the name or parameter being used. If any such name or parameter contains spaces, it must be enclosed by double quotation marks, for example, "this is an extremely long name or parameter."

Each command appears in this listing on multiple lines for readability, but in reality, would be given out as a single long string on the command line.

Add a User to a Profile

```
olsadmintool adduser --polname <policy name> --profname <profilename> --userdn
<enterprise user DN>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `adduser` command Use the `adduser` command to add an enterprise user to a profile within a policy. Provide the profile and policy names and the user DN.[1] Enterprise users are normal Oracle Internet Directory users with the additional capability of connecting to the database. Users added to a profile must be enterprise users.

[1] Command Footnote

Every command must include the directory host name, the bind DN, and the bind password. Any command may, as needed, also supply the subscriber administrative context (optional), the directory port number (also optional), or both. See also Table B–3, " Summary: olsadmintool Command Parameters" for additional details on these parameters.

Example of the adduser command

```
olsadmintool adduser --polname tradesecret --profname topsales --userdn "cn=perot"
-b "cn=EDS" -h ford -p 1890 -D cn=lbacsys -w lbacsyspwrd
```

> **See Also:** Rxefer to the *Oracle Database Advanced Security Administrator's Guide*, Chapter 13, Administering Enterprise User Security, for further concepts, tools, steps, and procedures.

Add Policy Administrators

```
olsadmintool addadmin --polname <policy name> --admindn <admin DN>
[ -b <admin context>] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the addadmin command

Use the `addadmin` command to add an enterprise user to the administrative group for a policy, so that the user is able to create, modify, or delete the specified policy's metadata. Provide the policy name and the new administrator's DN. This group should contain only enterprise users.

Example of the addadmin command

```
olsadmintool addadmin --polname defense --admindn "cn=scott,c=us"
-h yippee -D cn=lbacsys -w lbacsys
```

Add Policy Creator

```
olsadmintool addpolcreator --userdn <user DN>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the addpolcreator command

Use the `addpolcreator` command to enable the specified user to create policies. Provide the DN for the user.

Example of the addpolcreator command

```
olsadmintool addpolcreator --userdn "cn=scott" -h yippee -D cn=lbacsys -w lbacsys
```

Alter a Compartment

```
olsadmintool altercompartment --polname <policy name> --shortname <short
compartment name> --longname <new long compartment name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `altercompartment` command Use the `altercompartment` command to change the long name of a compartment. Provide the name of the policy, the short name of the compartment, and the new long name of the compartment.

Example of the `altercompartment` command

```
olsadmintool altercompartment --polname defense --shortname A --longname "Allied
Forces" -h yippee -D cn=defense_admin -w welcome1
```

Alter a Group

```
olsadmintool altergroup --polname <policy name> --shortname <short group name>
--longname <"new long group name">
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `altergroup` command Use the `altergroup` command to change the long name for a group component or parent group. Provide the name of the policy, the short name of the group, and the long name of the group.

Example of the `altergroup` command

```
olsadmintool altergroup --polname defense --shortname US --longname "United States
of America"  -h yippee -D cn=defense_admin -w welcome1
```

Alter a Group's Parent

```
olsadmintool altergroupparent --polname <policy name> --shortname <short group
name> [--parentname <new parent group name> ] [--clearparent]
--longname <"new long group name"> [--parentname <new short group name> ]
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `altergroupparent` command Use the `altergroupparent` command to change or remove the parent group of a group. Provide the name of the policy, the short name of the group, and either the short name of the parent group or the clearparent flag, but not both.

Examples of the `altergroupparent` command

```
olsadmintool altergroupparent --polname defense --shortname US --parentname
"Earth" -h yippee -p 5678 -D cn=defense_admin -w welcome1
or
olsadmintool altergroupparent --polname defense --shortname US --clearparent
-h yippee -p 5678 -D cn=defense_admin -w welcome1
```

Alter a Label

```
olsadmintool alterlabel --polname <policy name> --tag <tag number> --value <new
label value>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `alterlabel` command Use the `alterlabel` command to change the character string defining the label associated with a label tag. Provide the policy name, the numeric tag of the label, and the new character string representing the label.

Example of the `alterlabel` command

```
olsadmintool alterlabel --polname defense --tag 100 --value "TS:A:US" -h yippee -D
cn=defense_admin -w welcome1
```

Alter a Level

```
olsadmintool alterlevel --polname <policy name> --shortname <short level name>
--longname <"new long level name">
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `alterlevel` command Use the `alterlevel` command to change the long name of a level. Provide the name of the policy, the short name of the level, and the new long name of the level.

Example of the `alterlevel` command

```
olsadmintool alterlevel --polname defense --shortname TS
--longname "VERY TOP SECRET" -h yippee -D cn=defense_admin -w welcome1
```

Alter Policy

```
olsadmintool alterpolicy --name <policy name> --options <new options>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>

where <new options> can be any combination of the following entries:
INVERSE_GROUP, HIDE, LABEL_DEFAULT, LABEL_UPDATE, CHECK_CONTROL, READ_
CONTROL,WRITE_CONTROL,INSERT_CONTROL, DELETE_CONTROL, UPDATE_CONTROL, ALL_CONTROL,
or NO_CONTROL
```

Description of the `alterpolicy` command Use the `alterpolicy` command to alter the options of a policy. Provide the name of the policy and the new options.

Example of the `alterpolicy` command

```
olsadmintool alterpolicy --name defense --options "READ_CONTROL,INSERT_CONTROL" -h
yippee -D cn=defense_admin -w welcome1
```

Cancel Audit Options

```
olsadmintool noaudit --polname <policy name> --options <audit option name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>

where <audit option name> can be any combination of APPLY, REMOVE, SET, PRIVILEGE
```

Description of the `noaudit` command Use the `noaudit` command to cancel the audit options for a policy. Provide the policy name and the options that are no longer to be audited.

Example of the `noaudit` command

```
olsadmintool noaudit --polname defense --options "APPLY,PRIVILEGES" -h yippee -D
cn=defense_admin -w welcome1
```

Create a Compartment

```
olsadmintool createcompartment --polname <policy name> --tag <tag number>
--shortname <short compartment name> --longname <"long compartment name">
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `createcompartment` command Use the `createcompartment` command to create a new compartment component. Provide the name of the policy, the tag numeric value of the compartment, the short name of the compartment, and the long name of the compartment.

Example of the `createcompartment` command

```
olsadmintool createcompartment --polname defense --tag 100 --shortname A
--longname Alpha -h yippee -D cn=defense_admin -w welcome1
```

Create a Group

```
olsadmintool creategroup --polname <policy name> --tag <tag number> --shortname
<short group name> --longname <"long group name">
[--parentname <parent group name>]
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `creategroup` command Use the `creategroup` command to create a new group component. Provide the name of the policy, the tag numeric value

of the group, the short name of the group, the long name of the group, and the parent group name (optional).

Example of the `creategroup` command

```
olsadmintool creategroup --polname defense --tag 55 --shortname US
--longname "United States" -h yippee -D cn=defense_admin -w welcome1
```

Create a Label

```
olsadmintool createlabel --polname <policy name> --tag <tag number> --value <label
value>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `createlabel` command Use the `createlabel` command to create a valid data label. Provide the policy name, the numeric tag of the label to be created, and the character string representation of the label.

Example of the `createlabel` command

```
olsadmintool createlabel --polname defense --tag 100 --value "TS:A,B:US,CA"
-h yippee -D cn=defense_admin -w welcome1
```

Create a Level

```
olsadmintool createlevel --polname <policy name> --tag <tag number> --shortname
<short level name> --longname <"long level name">
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `createlevel` command Use the `createlevel` command to create a new level component. Provide the name of the policy, the tag numeric value, the short name of the level, and the long name of the level.

Example of the `createlevel` command

```
olsadmintool createlevel --polname defense --tag 100 --shortname TS
--longname "TOP SECRET" -h yippee -D cn=defense_admin -w welcome1
```

Create a Profile

```
olsadmintool createprofile --polname <policy name> --profname <profile name>
--maxreadlabel <max read label> --maxwritelabel <max write label> --minwritelabel
<min write label> --defreadlabel <default read label> --defrowlabel <default row
label> --privileges <privileges separated by comma>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `createprofile` command Use the `createprofile` command to create a new profile. Provide the policy name, the profile name, and either privileges, labels, or both privileges and labels. (A user profile can have either null label information or null privilege information, but not both null at the same time.) For labels, specify the maximum label users in this profile can use to read data, the maximum label users in this profile can use to write data, the minimum label users in this profile can use to write data, the default label for reading, the default row label for writing. For privileges, enclose in quotation marksthe list of privileges, separated by commas, for members of this profile.

Example of the `createprofile` command

```
olsadmintool createprofile --polname topsecret --profname topsales --maxreadlabel
"TS:A,B:US,CA" --maxwritelabel "TS:A,B:US,CA" --minwritelabel "C:A,B:US,CA"
```

```
--defreadlabel "TS:A,B:US,CA" --defrowlabel "C:A,B:US,CA"
--privileges "READ,COMPACCESS,WRITEACROSS"
-b EDS -h ford -p 1890 -D cn=lbacsys -w lbacsyspwrd
```

Create Policy

```
olsadmintool createpolicy --name <policy name> --colname <column name> --options
<options separated by commas>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>

where <new options> can be any combination of the following entries:
INVERSE_GROUP, HIDE, LABEL_DEFAULT, LABEL_UPDATE, CHECK_CONTROL, READ_CONTROL,
WRITE_CONTROL,INSERT_CONTROL, DELETE_CONTROL, UPDATE_CONTROL, ALL_CONTROL, or
NO_CONTROL
```

Description of the `createpolicy` command Use the `createpolicy` command to create a policy. Provide the name of the policy, the name of its label column, and the options.

Example of the `createpolicy` command

```
olsadmintool createpolicy --name defense --colname defense_col --options "READ_
CONTROL,UPDATE_CONTROL" -h yippee -p 389 -D cn=defense_admin -w welcome1
```

Describe a Profile

```
olsadmintool describeprofile --polname <policy name> --profname <profile name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `describeprofile` command Use the `describeprofile` command to see the contents of the specified profile in the specified policy. Provide the policy name and the name of the profile.

Example of the `describeprofile` command

```
olsadmintool describeprofile --polname defense --profname contractors
-h yippee -D cn=defense_admin -w welcome1
```

Drop a Compartment

```
olsadmintool dropcompartment --polname <policy name> --shortname <short
compartment name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `dropcompartment` command Use the `dropcompartment` command to remove a compartment component. Provide the name of the policy and the short name of the compartment.

Example of the `dropcompartment` command

```
olsadmintool dropcompartment --polname defense --shortname A
-h yippee -D cn=defense_admin -w welcome1
```

Drop a Group

```
olsadmintool dropgroup --polname <policy name> --shortname <short group name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `dropgroup` command Use the `dropgroup` command to remove a group component. Provide the policy name and the short group name.

Example of the `dropgroup` command

```
olsadmintool dropgroup --polname defense --shortname US
-h yippee -D cn=defense_admin -w welcome1
```

Drop a Label

```
olsadmintool droplabel --polname <policy name> --value <label value>
-h yippee [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `droplabel` command Use the `droplabel` command to drop a label from the policy. Provide the policy name and the string representation of the label.

Example of the `droplabel` command

```
olsadmintool droplabel --polname defense --value "TS:A:US"
h yippee -D cn=defense_admin -w welcome1
```

Drop a Level

```
olsadmintool droplevel --polname <policy name> --shortname <short level name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `droplevel` command Use the `droplevel` command to remove a level component from a specified policy. Provide the name of the policy and the short name of the level.

Example of the `droplevel` command

```
olsadmintool droplevel --polname defense --shortname TS
-h yippee -D cn=defense_admin -w welcome1
```

Drop a Policy

```
olsadmintool droppolicy --name <policy name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `droppolicy` command Use the `droppolicy` command to drop a policy. Provide the name of the policy to be dropped. For directory-enabled installations of Oracle Label Security, refer to "Subscribing Policies in Directory-Enabled Label Security" on page 10-1.

Example of the `droppolicy` command

```
olsadmintool droppolicy --name defense -h yippee -D cn=defense_admin -w welcome1
```

Drop a Profile

```
olsadmintool dropprofile --polname <policy name> --profname <profile name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `dropprofile` command Use the `dropprofile` command to remove the specified profile. Provide the policy name and the name of the profile to be dropped.

Note: Dropping a profile removes the authorization on that policy for all the users in the dropped profile. The users will be unable to see data protected by that policy.

Example of the `dropprofile` command

```
olsadmintool dropprofile --name defense --profname employees
-h yippee -D cn=defense_admin -w welcome1
```

Drop a User

```
olsadmintool dropuser --polname <policy name> --profname <profilename>
--userdn <enterprise user DN>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `dropuser` command Use the `dropuser` command to drop a user from the specified profile in the specified policy. Provide the policy name, the name of the profile, and the DN of the user.

Example of the `dropuser` command

```
olsadmintool dropuser --polname defense --profname contractors --userdn
"cn=hanssen,c=us" -h yippee -D cn=defense_admin -w welcome1
```

Drop Policy Administrator

```
olsadmintool dropadmin --polname <policy name> --admindn <admin DN>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `dropadmin` command Use the `dropadmin` command to remove an enterprise user from the administrative group of a policy, so that the user is no longer able to create, modify, or delete the specified policy's metadata. Provide the policy name and the DN of the administrator to be removed from the administrative group.

Example of the `dropadmin` command

```
olsadmintool dropadmin --polname defense --admindn "cn=scott,c=us"
-h yippee -D cn=lbacsys -w lbacsys
```

Drop Policy Creator

```
olsadmintool droppolcreator --userdn <user DN>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `droppolcreator` command Use the `droppolcreator` command to cancel the ability of the specified user to create policies. Provide the user's DN.

Example of the `droppolcreator` command

```
olsadmintool droppolcreator --userdn "cn-scott,c=us"
-b UA -h yippee -p 1890 -D <bind DN> -w <bind password>
```

Get Help for an olsadmintool Command

`olsadmintool` <*command name*> --help

List Profiles

```
olsadmintool listprofile --polname <policy name>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Description of the `listprofile` command Use the `listprofile` command to see a list of all profiles in a given policy. Provide the policy name.

Example of the `listprofile` command

```
olsadmintool listprofile --polname defense -b CIA
-h yippee -D cn=defense_admin -w welcome1
```

Set Audit Options

```
olsadmintool audit --polname <policy name> --options <audit option name> --type
<audit option type> --success <audit success type>
[ -b <admin context> ] -h <OID host> [-p <port>] -D <bind DN> -w <bind password>

where <audit option name> can be any combination of APPLY, REMOVE, SET, PRIVILEGE,
type can be "session" or "access", and success can be "successful", "not
successful" or "both".
```

Description of the `audit` command Use the `audit` command to set the audit options for a policy. Provide the policy name, the options to be audited, the type of audit, and the type of success to be audited.

Example of the `audit` command

```
olsadmintool audit --polname defense --options "APPLY,PRIVILEGE" --type session
--success success -h yippee -D cn=defense_admin -w welcome1
```

Relating Parameters to Commands for olsadmintool

All olsadmintool commands must specify connection parameters: the OID host, the bind DN, the bind password, and optionally, the port through which the connection to Oracle Internet Directory is to be made. (The default port is 389.)

All olsadmintool commands may specify, as needed, the subscriber/administrative-context using the `-b` flag.

The fact that specifying a parameter is optional, such as a port or an administrative context, is shown by enclosing the parameter within brackets. The two most common examples are `[ -b <admin context> ]` and `[-p <port>]`.

Since every command must specify a host, bind DN, and password, and may, if needed, also specify an administrative context, Table B–3, " Summary: olsadmintool Command Parameters" uses the abbreviation **CON** to represent all of these connection parameters as a group:

```
[ -b <admin context> ] h <OID host> [-p <port>] -D <bind DN> -w <bind password>
```

Summaries

Table B–3, " Summary: olsadmintool Command Parameters" summarizes the commands in the following categories:

- **Policies:** creating, altering, or dropping policies or their components, that is, levels, groups, and compartments
- **Data labels:** creating, altering, or dropping them

- **Administrators and policy creators:** adding or dropping them
- **Users:** adding or dropping users from a profile
- **Auditing options:** setting the options for what to audit for a policy
- **Profiles:** creating, listing, describing, or dropping them
- **Default read or row labels:** setting them

In Table B–3, " Summary: olsadmintool Command Parameters" and Table B–4, " Summary of Profile and Default Command Parameters", the column headings show only the parameters, not the keywords that must precede them. For example, Table B–3, " Summary: olsadmintool Command Parameters" shows *policyname* and *column-name* as parameters for the `createpolicy` command, without showing the keywords that must precede them (--name and --colname). These keywords are shown as required in each of the individual command descriptions, such as at Create Policy.

Table B–3, " Summary: olsadmintool Command Parameters" explains the individual parameters that are used as column headings in the summaries of Table B–3, " Summary: olsadmintool Command Parameters" and Table B–4, " Summary of Profile and Default Command Parameters".

In all these tables:

- X means required, and O means unused or omitted.
- OptionsP means policy enforcement options, that is, any combination of the following entries, separated by a comma:
  - `INVERSE_GROUP`
  - `HIDE`
  - `LABEL_DEFAULT`
  - `LABEL_UPDATE`
  - `CHECK_CONTROL`
  - `READ_CONTROL`
  - `WRITE_CONTROL`
  - `INSERT_CONTROL`
  - `DELETE_CONTROL`
  - `UPDATE_CONTROL`
  - `ALL_CONTROL`
  - `NO_CONTROL`
- OptionsA means audit options, that is, any comma-separated combination of the following entries: `SET`, `APPLY`, `REMOVE`, or `PRIVILEGE`.

Table B–3 Summary: olsadmintool Command Parameters

Command Category	**Commands & Parameters**						
Policies	**Command**	**policy name**	**column-name**	**optionsP**	**CON**		
	olsadmintool createpolicy	X	X	X	X		
	olsadmintool alterpolicy	X	O	X	X		
	olsadmintool droppolicy	X	O	O	X		
Within a Policy, Create:	**Command**	**policy name**	**tag**	**short name**	**long name**	**CON**	**parent name**
a level	olsadmintool createlevel	X	X	X	X	X	O
a group	olsadmintool creategroup	X	X	X	X	X	**[X]**
a compartment	olsadmintool createcompartment	X	X	X	X	X	O
Within a Policy, Alter:							
a level	olsadmintool alterlevel	X	O	u	u	u	O
a group or group parent	olsadmintool altergroup	X	O	X	X	X	O
	olsadmintool altergroupparent	X	O	X	O	X	[X]
	Command	**policy name**	**tag**	**short name**	**long name**	**CON**	**parent name**
a compartment	olsadmintool altercompartment	X	O	X	X	X	O
Within a Policy, Drop:							
level	olsadmintool droplevel	X	O	X	O	X	O
group	olsadmintool dropgroup	X	O	X	O	X	O
compartment	olsadmintool dropcompartment	X	O	X	O	X	O

Data Labels	**Command**	**policy name**	**tag**	**value**	**CON**
Create label	olsadmintool createlabel	X	X	X	X
Alter data label	olsadmintool alterlabel	X	X	X	X
Drop data label	olsadmintool droplabel	X	O	X	X
Policy Administrators	**Command**	**policy name**	**userDN**	**CON**	
Add an Admin	olsadmintool addadmin	X	X	X	

Table B–3 (Cont.) Summary: olsadmintool Command Parameters

Command Category	Commands & Parameters					
Drop an Admin	olsadmintool dropadmin	X	X	X		
Policy Creation	olsadmintool addpolcreator	O	X	X		
	`olsadmintool droppolcreator`	O	X	X		
Users	**Command**	**policy name**	**profile name**	**userDN**	**CON**	
Add a User	olsadmintool adduser	X	X	X	X	
Drop a User	`olsadmintool dropuser`	X	X	X	X	
Auditing	olsadmintool audit	X	**optionsA**	**type**	**success**	**CON**
	olsadmintool noaudit	X	X	X	X	X
Help on olsadmintool	olsadmintool <commandname> -- help	O	O	O	O	O

Table B–4 Summary of Profile and Default Command Parameters

Profile Action	Profile Command	Policy Name	Profile Name	Max Read Label	Max Write Label	Min Write Label	Def Read Label	Def Row Label	Priv's	CON
Create a Profile[1]	olsadmin tool create profile	X	X	X	X	X	X	X	X	X
List Profiles	olsadmin tool list profile	X	O	O	O	O	O	O	O	X
Describe a Profile	olsadmin tool describe profile	X	X	O	O	O	O	O	O	X
Drop a Profile	olsadmin tool drop profile	X	X	O	O	O	O	O	O	X

[1] In createprofile, specifying both privileges and labels is not required: a profile can specify labels, privileges, or both.

Examples of Using olsadmintool

The subsections that follow illustrate using the olsadmintool commands in typical tasks needed to set up Oracle Label Security in an Oracle Internet Directory environment. Each command appears in this listing on multiple lines for readability, but in reality, would be given out as a single long string on the command line. The summarized results of carrying out all these commands appear in Results of These Examples, which follows the last example.

Make Other Users Policy Creators

```
ORACLE_HOME/bin/olsadmintool addpolcreator --userdn "cn=snamudur,c=us"
 -b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=lbacsys,c=us" -w lbacsys
```

Create Policies with Valid Options

```
ORACLE_HOME/bin/olsadmintool createpolicy --name Policy1 --colname pol1
--options READ_CONTROL,WRITE_CONTROL -b "ou=Americas,o=Oracle,c=US"
-h yippee -p 389 -D "cn=snamudur,c=us" -w snamudur

ORACLE_HOME/bin/olsadmintool createpolicy --name Policy2 --colname pol2
--options READ_CONTROL -b "ou=Americas,o=Oracle,c=US"
-h yippee -p 389 -D "cn=lbacsys,c=us" -w lbacsys
```

Create Policy Administrators

```
ORACLE_HOME/bin/olsadmintool addadmin --polname Policy1
--admindn "cn=shwong,c=us" -b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D
"cn=snamudur,c=us" -w snamudur

ORACLE_HOME/bin/olsadmintool addadmin --polname Policy2
--admindn "cn=shwong,c=us" -b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D
"cn=lbacsys,c=us" -w lbacsys
```

Create Some Levels

```
ORACLE_HOME/bin/olsadmintool createlevel --polname Policy1 --tag 100
--shortname TS --longname "TOP SECRET" -b "ou=Americas,o=Oracle, c=US"
-h yippee -p 389 -D "cn=shwong,c=us" -w shwong

ORACLE_HOME/bin/olsadmintool createlevel --polname Policy1 --tag 99
--shortname S --longname SECRET -b "ou=Americas,o=Oracle,c=US"
-h yippee -p 389 -D "cn=shwong,c=us" -w shwong

ORACLE_HOME/bin/olsadmintool createlevel --polname Policy1 --tag 98
--shortname U --longname UNCLASSIFIED -b "ou=Americas,o=Oracle,c=US"
-h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Create Some Compartments

```
ORACLE_HOME/bin/olsadmintool createcompartment --polname Policy1 --tag 100
```

```
--shortname A --longname ALPHA -b "ou=Americas,o=Oracle,c=US"
-h yippee -p 389 D "cn=shwong,c=us" -w shwong

ORACLE_HOME/bin/olsadmintool createcompartment --polname Policy1 --tag 99
--shortname B --longname BETA -b "ou=Americas,o=Oracle,c=US"
-h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Create Some Groups

```
ORACLE_HOME/bin/olsadmintool creategroup --polname Policy1 --tag 100
--shortname G1 --longname GROUP1
-b "ou=Americas,o=Oracle,c=US"  -h yippee -p 389 -D "cn=shwong,c=us" -w shwong

ORACLE_HOME/bin/olsadmintool creategroup --polname Policy1 --tag 99
--shortname G2 --longname GROUP2
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong

ORACLE_HOME/bin/olsadmintool creategroup --polname Policy1 --tag 98
--shortname G3 --longname GROUP3
-b "ou=Americas,o=Oracle,c=US"  -h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Create Some Labels

```
ORACLE_HOME/bin/olsadmintool createlabel --polname Policy1 --tag 100
--value TS:A:G1
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong

ORACLE_HOME/bin/olsadmintool createlabel --polname Policy1 --tag 101
--value TS:A,B:G2
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Create a Profile

```
ORACLE_HOME/bin/olsadmintool createprofile --polname Policy1 --profname Profile1
--maxreadlabel TS:A:G1 --maxwritelabel TS:A:G1 --minwritelabel U::
--defreadlabel U:A:G1 --defrowlabel U:A:G1 --privileges WRITEUP,READ
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Add a User to the Profile

```
ORACLE_HOME/bin/olsadmintool adduser --polname Policy1 --profname Profile1
--userdn cn=nina,ou=Asia,o=microsoft,l=seattle,st=WA,c=US
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Add Another User to the Profile

```
ORACLE_HOME/bin/olsadmintool adduser --polname Policy1 --profname Profile1
--userdn cn=daniel,ou=France,o=oracle,l=madison,st=WI,c=US
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Set Some Audit Options

```
ORACLE_HOME/bin/olsadmintool audit --polname Policy1 --option "SET,APPLY"
--type SESSION --success BOTH
-b "ou=Americas,o=Oracle,c=US" -h yippee -p 389 -D "cn=shwong,c=us" -w shwong
```

Results of These Examples

As a result of running the sets of olsadmintool commands outlined, this sample Oracle Label Security site has the following structure:

- **Policy creators:** User snamudur
- **Policies:** Policy1 and Policy2.
- **Policy Administrators:** User shwong
- **Levels, Compartments, and Groups:** Refer to Table B–5, " Label Component Definitions from Using olsadmintool Commands".

Table B–5 Label Component Definitions from Using olsadmintool Commands

Label Component	Tag	Short Name	Long Name
Level	100	TS	TOP SECRET
	99	S	SECRET
	98	U	UNCLASSIFIED
Compartment	100	A	ALPHA
	99	B	BETA
Group	100	G1	GROUP1
	99	G2	GROUP2
	98	G3	GROUP3

- **Data labels:** Tag 100 for TS:A:G1 and tag 101 for TS:A,B:G2
- **Users:** Nina, from the Asia group of Microsoft, based in Seattle, Washington, managed under the Americas organization of the US Oracle organization, and Daniel, from the France group of Oracle in Madison, Wisconsin, managed under the same organization.
- **Profiles:** Refer to Table B–6, " Contents of Profile1 from Using olsadmintool Commands".

Table B–6 Contents of Profile1 from Using olsadmintool Commands

Profile Element	Contents	Long-name Expansion or Meaning
MaxReadLabel	TS:A:G1	TOP SECRET:ALPHA:GROUP1
MaxWriteLabel	TS:A:G1	TOP SECRET:ALPHA:GROUP1
MinWriteLabel	U::	UNCLASSIFIED (not restricted to any compartments or groups)
DefReadLabel	U:A:G1	UNCLASSIFIED:ALPHA:GROUP1
DefRowLabel	U:A:G1	UNCLASSIFIED:ALPHA:GROUP1
Privileges	WRITE_UP, READ	User can read any row and raise the level of rows the user writes.

- **Auditing options:** SET, APPLY, SESSION, and BOTH

C

Oracle Label Security in an RAC Environment

This appendix discusses using Oracle Label Security in a Real Application Clusters (RAC) environment. It includes the following sections:

- Using Oracle Label Security Policy Functions in an RAC Environment
- Using Transparent Application Failover in Oracle Label Security

Using Oracle Label Security Policy Functions in an RAC Environment

Policy changes made on one instance are available to other instances in the RAC immediately. It is not necessary to restart the other instances to pick up the changes.

Important changes made on one database instance are automatically propagated to the other instances. One example would be creating a new policy. Another would be altering the policy options.

Propagating such changes ensures two valuable protections:

- That all users of the table are subject to the same policy
- That if any instance fails, continuation of its work by other instances will use the same policies and parameters that were in force immediately prior to that failure. So, if a policy had been enabled or disabled, it would be seen as such in all instances.

If an administrator changes policy information in one instance by using the policy functions listed in Table C–1, Oracle Label Security stores the relevant information about whatever that function call changed. The new information is immediately available to the other active instances in the RAC, enabling uniformity among users of the affected policies.

Table C–1 Policy Functions Preserving Status in an RAC Environment

Policy Functions	Comments
sa_sysdba.create_policy()	Creates a new policy
sa_sysdba.drop_policy()	Drops an existing policy
sa_sysdba.enable_policy()	Enables an existing policy
sa_sysdba.disable_policy()	Disables an existing policy
sa_sysdba.alter_policy()	Alters an existing policy

Using Transparent Application Failover in Oracle Label Security

Session information is preserved on Transparent Application Failover. Any changes to the session's information by way of session functions listed in Table C–2 are preserved on Transparent Application Failover.

For example, suppose a user `Scott` is logged on with default label `Top Secret`. If he calls sa_session.set_label() to change his session label to `Secret`, and a failover to another instance occurs, he will see no change but his session label remains `Secret`.

Preserving current user session information means that the access permissions and restrictions on what data that user can see or affect remain as they were. Despite the failover, the user can see and affect only the tables and rows accessible before the failover. If preservation were not the case, failing over to another instance could cause or enable the user to see a different set of data.

Whenever one of the session functions listed in Table C–2 is used, Oracle Label Security stores the relevant information about whatever was changed by that function call.

Table C–2 Session Functions Preserving Status in an RAC Environment

Session Functions	Comments
sa_session.set_label()	Lets the user set a new level and new compartments and groups to which he or she has read access
sa_session.set_row_label()	Lets the user set the default row label that will be applied to new rows
sa_session.save_default_labels()	Lets the user store the current session label and row label as the default for future sessions
sa_session.restore_default_labels()	Lets the user reset the current session label and row label to the stored default settings
sa_session.set_access_profile()	Sets the Oracle Label Security authorizations and privileges of the database session to those of the specified user

D

Frequently Asked Questions on Oracle Label Security

This appendix attempts to answer some of the most common and frequently asked questions on Oracle Label Security.

Who uses Oracle Label Security?

Sensitivity labels are used to categorize data in virtually every industry. These industries include health care, law enforcement, energy, retail, national security, and defense industries. The following list gives some examples of sensitivity labels:

- Internal
- Confidential
- Physician Only
- Highly Sensitive
- Widget Corporation
- Confidential : Chicago Operation
- Sensitive : Finance : Europe
- Top Secret
- Unclassified

How can Oracle Label Security address my security needs?

Oracle Label Security can be used to label data and restrict access with a high degree of granularity. This is especially useful when multiple organizations or companies share a single application. Sensitivity labels can be used to restrict application users to an organization or to a subset of data within an organization.

Data privacy is important to consumers and regulatory measures continue to be announced. Oracle Label Security can be used to implement privacy policies on data, restricting access to only those who have a need-to-know.

Should I use Oracle Label Security to protect all my tables?

No. The traditional Oracle discretionary access control (DAC) object privileges such as `SELECT`, `INSERT`, `UPDATE`, and `DELETE` combined with database roles and stored procedures are sufficient in most cases.

What is the difference between Oracle Virtual Private Database and Oracle Label Security?

Oracle Virtual Private Database (VPD) is provided at no additional cost with the Enterprise Edition of Oracle Database. Oracle Label Security is an add-on security option for the Oracle Database Enterprise Edition.

Oracle VPD is a term used for several powerful security features like, fine grained access control (FGAC), application context and global application context. VPD policies are written using `PL/SQL`, and can be assigned to an individual table or view. An information request, that accesses a table or view protected by VPD, is modified according to the policy assigned to the table or view.

VPD policies can be as simple as enforcing access during business hours. VPD policies can restrict access by comparing the value of an attribute in an individual row with an application context value. Global application context allows an application context to be accessed across multiple database sessions, reducing or eliminating the need to create a separate application context for each user session.

Oracle Label Security is an out-of-the-box solution for row level security. No coding or software development is required, allowing the administrator to focus completely on the policy. Oracle Label Security provides an interface for creating policies, specifying enforcement options, defining data sensitivity labels, establishing user label authorizations, and protecting individual tables or schemes.

Data sensitivity labels provide a powerful and flexible method of restricting access to data. For example, data belonging to different organizations or companies can be separated using data sensitivity labels and selectively shared between companies by changing the data sensitivity label.

Depending on the complexity of the security policy, Oracle Virtual Private Database may be the preferred method for implementing your security policy. Oracle Label Security is best suited for situations where access control decisions need to be based on the sensitivity of the information.

Can I combine Virtual Private Database and Oracle Label Security?

Yes. The following scenarios are possible:

- A `WHERE` clause can be appended to an OLS policy, which provides one more level of granularity. An example would be that users, regardless of their label authorizations, are only allowed to connect from a specific IP address or subnet, and during business hours only.
- A VPD policy, whether column sensitive or not, can evaluate user labels and determine access to columns and rows without the need to apply data labels.

Can I use Oracle Label Security with the Oracle E-Business Suite?

Oracle Applications are using Oracle VPD to provide new functionality and security protections.

The following Best Practices document can be found on the Oracle Technology Network Web site:

`http://otn.oracle.com/deploy/security/database-security/pdf/WhitePaper1169_rraj_srtata.pdf`

Can I use Oracle Label Security with Oracle Database Vault?

You can protect Oracle Database Vault tables using Oracle Label Security just as you would do for an Oracle Database table.

In addition, Oracle Label Security can be used together with Database Vault features. You can assign Oracle Label Security labels to Database Vault Factors. These labels are then merged with the user clearance labels, following the algorithms documented in chapter 4.4.5 of the OLS admin guide: "Merging Labels with the MERGE_LABEL Function", before access control decisions are being made by comparing the merged user labels with the row labels.

The following example on the Oracle Technology Network Web site also discusses using Oracle Label security alongwith Oracle Database Vault features:

http://www.oracle.com/technology/deploy/security/database-security/howtos/ols_dbv-how-to.html

Does Oracle Label Security provide column-level access control?

No, Oracle Label Security is not column aware. This behavior is available with Virtual Private Database (VPD). A VPD policy can be written so that it only becomes active when a certain column is part of a SQL statement against a protected table. If the *column sensitivity* switch is on, then VPD either returns only those rows for which the sensitive column values are accessible to the user, or it returns all rows with all cells in the sensitive column being empty, except those values that the user is allowed to see.

The following link on the Oracle Technology Network Web site contains an example:

http://www.oracle.com/technology/deploy/security/database-security/virtual-private-database/index.html

A column-sensitive VPD policy can determine access to a specific column by evaluating OLS user labels.

Can I base Secure Application Roles on Oracle Label Security?

Yes. The procedure that determines if the `SET ROLE` command is executed can evaluate OLS user labels. In this case, the OLS policy does not need to be applied to a table, since row labels are not part of this solution.

What are Trusted Stored Program Units?

Stored procedures, functions and packages execute with the system and object privileges (DAC) of the definer. If the invoker is a user with Oracle Label Security user clearances (labels), the procedure executes with a combination of the definer's DAC privileges and the invoker's security clearances.

Trusted stored procedures are procedures that are either granted the OLS privilege `FULL` or `READ`. When a trusted stored program unit is run, the policy privileges in force are a combination of the invoking user's privileges and the program unit's privileges.

Does VPD or OLS add an additional column to the protected table?

When you apply an Oracle Label Security (OLS) policy to a table, the policy adds an additional column to the table. The name of this column needs to be specified when the policy is initially created.

An existing column can be used to store the OLS row labels. This column must have the `NUMBER(10)` datatype.

Virtual Private Database (VPD) does not add an additional column to the protected table.

Why should the additional OLS row label column be hidden?

Most applications were not designed with access control mechanisms in mind, so Oracle Label Security (OLS) needs to do this transparently.

When an application queries a table with a `SELECT FROM` *`tablename`* statement, it returns all columns, including the unhidden label column. Existing applications may not be designed to display an additional column, and malfunction. However, if the label column is hidden, then it is displayed only when it's name is included in the SQL statement. A `SELECT FROM` *`tablename`* would return all columns as expected by the application, excluding the hidden OLS column.

Where can I find Oracle Label Security?

Oracle Label Security ships with the Oracle Database Enterprise Edition CD. Oracle Label Security is not installed as part of the default Oracle installation. You need to perform a custom installation to add Oracle Label Security to the database.

E

Reference

This appendix provides the following reference information:

Oracle Label Security Data Dictionary Tables and Views

Oracle Database Data Dictionary Tables

Oracle Label Security does not in any way label the Oracle data dictionary tables. Access is controlled by standard *Oracle Database* system and object privileges. For a description of all data dictionary tables and views, refe to the *Oracle Database Reference*.

Oracle Label Security Data Dictionary Views

Oracle Label Security maintains an independent set of data dictionary tables. These tables are exempt from any policy enforcement. This section lists the views that can display information related to Oracle Label Security.

Note that access to the DBA views is granted by default to the SELECT_CATALOG_ROLE, a standard *Oracle Database* role that lets you examine the *Oracle Database* data dictionary.

ALL_SA_AUDIT_OPTIONS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
APY		VARCHAR2(3)
REM		VARCHAR2(3)
SET_		VARCHAR2(3)
PRV		VARCHAR2(3)

ALL_SA_COMPARTMENTS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
COMP_NUM	NOT NULL	NUMBER(4)
SHORT_NAME	NOT NULL	VARCHAR2(30)
LONG_NAME	NOT NULL	VARCHAR2(80)

ALL_SA_DATA_LABELS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
LABEL		VARCHAR2(4000)
LABEL_TAG		NUMBER

ALL_SA_GROUPS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
GROUP_NUM	NOT NULL	NUMBER(4)
SHORT_NAME	NOT NULL	VARCHAR2(30)
LONG_NAME	NOT NULL	VARCHAR2(80)
PARENT_NUM		NUMBER(4)
PARENT_NAME		VARCHAR2(30)

ALL_SA_LABELS

Access to ALL_SA_LABELS is PUBLIC. However only the labels authorized for read access by the session are visible.

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
LABEL		VARCHAR2(4000)
LABEL_TAG		NUMBER
LABEL_TYPE		VARCHAR2(15)

ALL_SA_LEVELS

Name	Null?	Type
POLICY_NAME		VARCHAR2(30)
LEVEL_NUM		NUMBER(4)
SHORT_NAME		VARCHAR2(30)
LONG_NAME		VARCHAR2(80)

ALL_SA_POLICIES

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME	NOT NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)
POLICY_OPTIONS		VARCHAR2(4000)

ALL_SA_PROG_PRIVS

Name	Null?	Type
SCHEMA_NAME	NOT NULL	VARCHAR2(30)
PROGRAM_NAME	NOT NULL	VARCHAR(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
PROGRAM_PRIVILEGES		VARCHAR2(4000)

ALL_SA_SCHEMA_POLICIES

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
SCHEMA_NAME	NOT NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)
SCHEMA_OPTIONS		VARCHAR2(4000)

ALL_SA_TABLE_POLICIES

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
SCHEMA_NAME	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)
TABLE_OPTIONS		VARCHAR2(4000)
FUNCTION		VARCHAR2(1024)
PREDICATE		VARCHAR2(256)

ALL_SA_USERS

Name	Null?	Type
USER_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_PRIVILEGES		VARCHAR2(4000)
MAX_READ_LABEL		VARCHAR2(4000)
MAX_WRITE_LABEL		VARCHAR2(4000)
MIN_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_READ_LABEL		VARCHAR2(4000)
DEFAULT_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_ROW_LABEL		VARCHAR2(4000)
USER_LABELS		VARCHAR2(4000)

ALL_SA_USER_LABELS

Name	Null?	Type
USER_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
MAX_READ_LABEL	NOT NULL	VARCHAR2(4000)
MAX_WRITE_LABEL		VARCHAR2(4000)
MIN_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_READ_LABEL		VARCHAR2(4000)
DEFAULT_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_ROW_LABEL		VARCHAR2(4000)
LABELS		VARCHAR2(4000)

Note: The field USER_LABELS in ALL_SA_USERS and the field LABELS in ALL_SA_USER_LABELS are retained solely for backward compatibility and will be removed in the next release.

ALL_SA_USER_LEVELS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
MAX_LEVEL	NOT NULL	VARCHAR2(30)
MIN_LEVEL	NOT NULL	VARCHAR2(30)
DEF_LEVEL	NOT NULL	VARCHAR2(30)
ROW_LEVEL	NOT NULL	VARCHAR2(30)

ALL_SA_USER_PRIVS

Name	Null?	Type
USER_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_PRIVILEGES		VARCHAR2(4000)

DBA_SA_AUDIT_OPTIONS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
APY		VARCHAR2(3)
REM		VARCHAR2(3)
SET_		VARCHAR2(3)
PRV		VARCHAR2(3)

DBA_SA_COMPARTMENTS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
COMP_NUM	NOT NULL	NUMBER(4)
SHORT_NAME	NOT NULL	VARCHAR2(30)
LONG_NAME	NOT NULL	VARCHAR2(80)

DBA_SA_DATA_LABELS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
LABEL		VARCHAR2(4000)
LABEL_TAG		NUMBER

DBA_SA_GROUPS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
GROUP_NUM	NOT NULL	NUMBER(4)
SHORT_NAME	NOT NULL	VARCHAR2(30)
LONG_NAME	NOT NULL	VARCHAR2(80)
PARENT_NUM		NUMBER(4)
PARENT_NAME		VARCHAR2(30)

DBA_SA_GROUP_HIERARCHY

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
HIERARCHY_LEVEL		NUMBER
GROUP_NAME		VARCHAR2(4000)

DBA_SA_LABELS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
LABEL		VARCHAR2(4000)
LABEL_TAG		NUMBER
LABEL_TYPE		VARCHAR2(15)

DBA_SA_LEVELS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
LEVEL_NUM	NOT NULL	NUMBER(4)
SHORT_NAME	NOT NULL	VARCHAR2(30)
LONG_NAME	NOT NULL	VARCHAR2(80)

DBA_SA_POLICIES

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
COLUMN_NAME	NOT NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)
POLICY_OPTIONS		VARCHAR2(4000)

DBA_SA_PROG_PRIVS

Name	Null?	Type
SCHEMA_NAME	NOT NULL	VARCHAR2(30)
PROGRAM_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
PROGRAM_PRIVILEGES		VARCHAR2(4000)

DBA_SA_SCHEMA_POLICIES

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
SCHEMA_NAME	NOT NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)
SCHEMA_OPTIONS		VARCHAR2(4000)

DBA_SA_TABLE_POLICIES

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
SCHEMA_NAME	NOT NULL	VARCHAR2(30)
TABLE_NAME	NOT NULL	VARCHAR2(30)
STATUS		VARCHAR2(8)
TABLE_OPTIONS		VARCHAR2(4000)
FUNCTION		VARCHAR2(1024)
PREDICATE		VARCHAR2(256)

DBA_SA_USERS

Name	Null?	Type
USER_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_PRIVILEGES		VARCHAR2(4000)

Name	Null?	Type
MAX_READ_LABEL		VARCHAR2(4000)
MAX_WRITE_LABEL		VARCHAR2(4000)
MIN_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_READ_LABEL		VARCHAR2(4000)
DEFAULT_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_ROW_LABEL		VARCHAR2(4000)
USER_LABELS		VARCHAR2(4000)

Note: The field USER_LABELS in DBA_SA_USERS is retained solely for backward compatibility and will be removed in the next release.

DBA_SA_USER_COMPARTMENTS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
COMP	NOT NULL	VARCHAR2(30)
RW_ACCESS		VARCHAR2(5)
DEF_COMP	NOT NULL	VARCHAR2(1)
ROW_COMP	NOT NULL	VARCHAR2(1)

DBA_SA_USER_GROUPS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
GRP	NOT NULL	VARCHAR2(30)
RW_ACCESS		VARCHAR2(5)
DEF_GROUP	NOT NULL	VARCHAR2(1)
ROW_GROUP	NOT NULL	VARCHAR2(1)

DBA_SA_USER_LABELS

Name	Null?	Type
USER_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
MAX_READ_LABEL	NOT NULL	VARCHAR2(4000)

Name	Null?	Type
MAX_WRITE_LABEL		VARCHAR2(4000)
MIN_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_READ_LABEL		VARCHAR2(4000)
DEFAULT_WRITE_LABEL		VARCHAR2(4000)
DEFAULT_ROW_LABEL		VARCHAR2(4000)
LABELS		VARCHAR2(4000)

Note: The field LABELS in DBA_SA_USER_LABELS is retained solely for backward compatibility and will be removed in the next release.

DBA_SA_USER_LEVELS

Name	Null?	Type
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_NAME	NOT NULL	VARCHAR2(30)
MAX_LEVEL	NOT NULL	VARCHAR2(30)
MIN_LEVEL	NOT NULL	VARCHAR2(30)
DEF_LEVEL	NOT NULL	VARCHAR2(30)
ROW_LEVEL	NOT NULL	VARCHAR2(30)

DBA_SA_USER_PRIVS

Name	Null?	Type
USER_NAME	NOT NULL	VARCHAR2(30)
POLICY_NAME	NOT NULL	VARCHAR2(30)
USER_PRIVILEGES		VARCHAR2(4000)

Oracle Label Security Auditing Views

Using the SA_AUDIT_ADMIN.CREATE_VIEW procedure, you can create an audit trail view for a specific policy. By default, this view is named DBA_*policyname*_AUDIT_TRAIL.

The DBA_SA_AUDIT_OPTIONS view contains the columns POLICY_NAME, USER_NAME, APY, SET_, and PRV.

See Also: "Creating and Dropping an Audit Trail View for Oracle Label Security" on page 12-6

Restrictions in Oracle Label Security

The following restrictions exist in this Oracle Label Security release:

CREATE TABLE AS SELECT Restriction in Oracle Label Security

If you attempt to perform CREATE TABLE AS SELECT in a schema that is protected by an Oracle Label Security policy, then the statement will fail.

Label Tag Restriction

Label tags must be unique across the policies in the database. When you use multiple policies in a database, you cannot use the same numeric label tag in different policies.

Export Restriction in Oracle Label Security

The LBACSYS schema cannot be exported due to the use of opaque types in Oracle Label Security. An export of the entire database (parameter FULL=Y) with Oracle Label Security installed can be done, except that the LBACSYS schema would not be exported.

Oracle Label Security Removal Restriction

Do not perform a DROP USER CASCADE on the LBACSYS account.

Connect to the database as user SYS, using the AS SYSDBA syntax, and run the file `$ORACLE_HOME/rdbms/admin/catnools.sql` to remove Oracle Label Security.

> **See Also:** Your platform-specific Oracle installation documentation

Shared Schema Support

User accounts defined in the Oracle Internet Directory cannot be given individual Oracle Label Security authorizations. However, authorizations can be given to the shared schema to which the directory users are mapped.

The Oracle Label Security function SET_ACCESS_PROFILE can be used programmatically to set the label authorization profile to use after a user has been authenticated and mapped to a shared schema. Oracle Label Security does not enforce a mapping between users who are given label authorizations in Oracle Label Security and actual database users.

Hidden Columns Restriction

PL/SQL does not recognize references to hidden columns in tables. A compiler error will be generated.

Installing Oracle Label Security

The person intending to install Oracle Label Security first selects the Custom installation choice. Oracle Label Security is listed as one of the options in the custom installation screen. After copying the Oracle Label Security files and relinking Oracle, the installer software automatically launches the Database Configuration Assistant (DBCA) during the database registration process, to configure options for the database to be created.

In DBCA, if Oracle Internet Directory is to be enabled for Oracle Label Security use, an additional option enables the installer users to configure the password for the Oracle Directory Integration and Provisioning (DIP) user. A DIP user with default password DIP has been created by catproc.sql. If the password is set during this configuration step, then the DIP provisioning profile will be created with the new DIP password.

Behind the scenes, DBCA does the following:

- Runs catolsd.sql (as supposed to running catols.sql for a standalone Oracle Label Security configuration)
- Creates the DIP provisioning profile with the given database DN for this database
- Runs the bootstrap utility to refresh the database with policy information from Oracle Internet Directory
- adds database DN to the cn=DBServers group

Note: If this password is ever changed, Oracle Internet Directory must be updated with this information, using the provisioning tool oidprovtool.

Oracle Label Security and the SYS.AUD$ Table

Installing Oracle Label Security automatically moves the AUD$ table out of SYS and into SYSTEM, and into a different tablespace.

Having the AUD$ table in the SYSTEM schema is supported when Oracle Label Security is being used.

When Oracle Label Security is not installed, moving the SYS.AUD$ table out of the SYSTEM tablespace is not supported because the Oracle code makes implicit assumptions about the data dictionary tables, such as SYS.AUD$, in support of upgrades and backup/recovery scenarios. Moving SYS.AUD$ is not supported unless done by Oracle when Oracle Label Security is installed.

Removing Oracle Label Security

Perform the following steps to remove Oracle Label Security. Do not perform a DROP USER CASCADE on the LBACSYS account to remove Oracle Label Security.

1. Connect AS SYSDBA.
2. Run the $ORACLE_HOME/rdbms/admin/catnools.sql script to delete the LBACSYS account.
3. Use the Oracle Universal Installer to remove Oracle Label Security.

See Also: Your platform-specific Oracle installation documentation

Index

A

B

C

D

E

F

G

H

I

L

M

N

O

P

R

S

T

U

V

W