Oracle® Advanced Security

Administrator's Guide

Release 9.0.1

June 2001

Part No.   A90150-01

ORACLE®

Oracle Advanced Security Administrator's Guide, Release 9.0.1

Part No. A90150-01

Author: Mike Cowan

Contributors: Kristy Browder, Sudha Iyer, Nina Lewis, Michael Hwa, Adam Lindsey Jacobs, Lakshmi Kethana, Andrew Koyfman, Van Le, Andy Philips, Ramana Turlapati, Philip Thornton, Gary Gilchrist, Min-Hank Ho, Torrance Brooksfuller, Cynthia Kibbe.

Graphic Artist: Valarie Moore

Contents

Part I   Introduction

1   Introduction to Oracle Advanced Security

Part II   Encryption, Integrity, and JDBC

2   Configuring Data Encryption and Integrity

3   Thin JDBC Support

Part III   Configuring Authentication Methods

4 Configuring RADIUS Authentication

5 Configuring CyberSafe Authentication

8   Configuring Entrust-Enabled SSL Authentication

9   Configuring Multiple Authentication Methods

Part IV   Oracle DCE Integration

10   Overview of Oracle DCE Integration

11   Configuring DCE for Oracle DCE Integration

12   Configuring Oracle9*i* for Oracle DCE Integration

13 Connecting to an Oracle Database in DCE

14 DCE and Non-DCE Interoperability

Part V   Oracle9*i* Enterprise User Security

15 Managing Enterprise User Security

16 Using Oracle Wallet Manager

17 Using Oracle Enterprise Login Assistant

18 Using Oracle Enterprise Security Manager

Part VI   Appendixes

A   Data Encryption and Integrity Parameters

B   Authentication Parameters

C   Integrating Authentication Devices Using RADIUS

D   Oracle Advanced Security FIPS 140-1 Settings

E Oracle Implementation of Java SSL

F Abbreviations and Acronyms

Glossary

Index

List of Figures

List of Tables

Send Us Your Comments

Oracle Advanced Security Administrator's Guide, Release 9.0.1

Part No. A90150-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this document. Your input is an important part of the information used for revision.

- Did you find any errors?
- Is the information clearly presented?
- Do you need more information? If so, where?
- Are the examples correct? Do you need more examples?
- What features did you like most?

If you find any errors or have any other suggestions for improvement, please indicate the document title and part number, and the chapter, section, and page number (if available). You can send comments to us in the following ways:

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227   Attn: Server Technologies Documentation Manager
- Postal service:
  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

If you have problems with the software, please contact your local Oracle Support Services.

xx

Preface

Welcome to the Oracle Advanced Security Administrator's Guide for Release 9.0.1 of Oracle Advanced Security.

Oracle Advanced Security contains a comprehensive suite of security features that protect enterprise networks and securely extend them to the Internet. It provides a single source of integration with multiple network encryption and authentication solutions, single sign-on services, and security protocols.

The Oracle Advanced Security Administrator's Guide describes how to implement, configure and administer Oracle Advanced Security.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

The Oracle Advanced Security Administrator's Guide is intended for users and systems professionals involved with the implementation, configuration, and administration of Oracle Advanced Security including:

- Implementation consultants
- System administrators
- Security administrators

Organization

This document contains:

Part I: Introduction

Chapter 1, Introduction to Oracle Advanced Security

This chapter provides an overview of Oracle Advanced Security features provided with this release.

Part II: Encryption, Integrity, and JDBC

Chapter 2, Configuring Data Encryption and Integrity

This chapter describes how to configure data encryption and integrity within an existing Oracle Net Release 9.0.1 network.

Chapter 3, Thin JDBC Support

This chapter provides an overview of the Java implementation of Oracle Advanced Security, which lets Thin Java Database Connectivity (JDBC) clients securely connect to Oracle9*i* databases.

Part III: Configuring Authentication Methods

Chapter 4, Configuring RADIUS Authentication

This chapter describes how to configure Oracle for use with RADIUS (Remote Authentication Dial-In User Service). It provides an overview of how RADIUS works within an Oracle environment, and describes how to enable RADIUS authentication and accounting. It also introduces the challenge-response user

interface that third party vendors can customize to integrate with third party authentication devices.

Chapter 5, Configuring CyberSafe Authentication

This chapter describes how to configure Oracle for use with CyberSafe, and provides a brief overview of steps to configure CyberSafe to authenticate Oracle users.

Chapter 6, Configuring Kerberos Authentication

This chapter describes how to configure Oracle for use with MIT Kerberos and provides a brief overview of steps to configure Kerberos to authenticate Oracle users.

Chapter 7, Configuring Secure Sockets Layer Authentication

This chapter describes the SSL feature of Oracle Advanced Security and explains how to configure SSL.

Chapter 8, Configuring Entrust-Enabled SSL Authentication

This chapter describes how to configure and use Entrust-enabled Oracle Advanced Security for Secure Socket Layer (SSL) authentication.

Chapter 9, Configuring Multiple Authentication Methods

This chapter describes the authentication methods that can be used with Oracle Advanced Security, and how to user conventional user name and password authentication. It also describes how to configure the network so that Oracle clients can user a specific authentication method, and Oracle servers can accept any method specified.

Part IV: Oracle DCE Integration

Chapter 10, Overview of Oracle DCE Integration

This chapter provides a brief discussion of Open Software Foundation (OSF) DCE and Oracle DCE Integration.

Chapter 11, Configuring DCE for Oracle DCE Integration

This chapter describes what you need to do to configure DCE to use Oracle DCE Integration. It also describes how to configure the DCE CDS naming adapter.

Chapter 12, Configuring Oracle9i for Oracle DCE Integration

This chapter describes the DCE parameters that you need to add to the configuration files to enable clients and servers to access Oracle servers in the DCE environment. It also describes some Oracle Server configuration that you need to perform, such as setting up DCE groups to map to external roles. Additionally, it describes how to configure clients to use the DCE CDS naming adapter.

Chapter 13, Connecting to an Oracle Database in DCE

This chapter describes how to connect to an Oracle database in a DCE environment.

Chapter 14, DCE and Non-DCE Interoperability

This chapter describes how clients outside of DCE can access Oracle databases using another protocol such as TCP/IP.

Part V: Oracle9i Enterprise User Security

Chapter 15, Managing Enterprise User Security

This chapter describes Oracle directory and security integration. It describes its components and provides an overview of the interaction between the components.

Chapter 16, Using Oracle Wallet Manager

This chapter describes how to configure and use the Oracle Wallet Manager.

Chapter 17, Using Oracle Enterprise Login Assistant

This chapter describes how to configure and use the Oracle Enterprise Login Assistant.

Chapter 18, Using Oracle Enterprise Security Manager

This chapter describes how an Enterprise DBA uses Oracle Enterprise Security Manager to administer database security in an enterprise domain of Oracle9*i* databases.

Part VI: Appendixes

Appendix A, Data Encryption and Integrity Parameters

This appendix describes Oracle Advanced Security data encryption and integrity configuration parameters.

Appendix B, Authentication Parameters

This appendix describes Oracle Advanced Security authentication configuration file parameters.

Appendix C, Integrating Authentication Devices Using RADIUS

This appendix explains how third party authentication device vendors can integrate their devices and customize the graphical user interface used in RADIUS challenge-response authentication.

Appendix D, Oracle Advanced Security FIPS 140-1 Settings

This appendix describes the *Sqlnet.ora* configuration parameters required to comply with the FIPS 140-1 Level 2 evaluated configuration.

Appendix E, Oracle Implementation of Java SSL

This appendix provides an overview of components and usage of the Oracle implementation of Java SSL.

Appendix F, Abbreviations and Acronyms

This appendix defines abbreviations and acronyms used in this document.

Related Documentation

For more information, see these Oracle resources:

- *ACE/Server Administration Manual, from Security Dynamics*
- *ACE/Server Client for UNIX, from Security Dynamics*
- *ACE/Server Installation Manual, from Security Dynamics*
- *Oracle Net Services Administrator's Guide*
- *Oracle9i Heterogeneous Connectivity Administrator's Guide*
- *Oracle9i Enterprise JavaBeans Developer's Guide and Reference*
- *Oracle9i JDBC Developer's Guide and Reference*
- *Oracle Internet Directory Administrator's Guide*
- *Oracle9i Database Administrator's Guide*

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle9i Sample Schemas* for information on how these schemas were created and how you can use them yourself.

In North America, printed documentation is available for sale in the Oracle Store at

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

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

```
http://www.oraclebookshop.com/
```

Other customers can contact their Oracle representative to purchase printed documentation.

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://technet.oracle.com/membership/index.htm
```

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

```
http://technet.oracle.com/docs/index.htm
```

For information from third-party vendors, see:

- *RADIUS Administrator's Guide*

- *CyberSafe TrustBroker Release Notes*

- *CyberSafe TrustBroker Administrator's Guide*

- *CyberSafe TrustBroker Navigator Administrator's Guide*

- *CyberSafe TrustBroker UNIX User's Guide, Release*

- *CyberSafe TrustBroker Windows and Windows NT User's Guide*

- *CyberSafe TrustBroker Client*

- *CyberSafe TrustBroker Server*

- *CyberSafe Trust Broker* documentation

- Notes about building and installing Kerberos from Kerberos V5 source distribution

- *Entrust/PKI for Oracle*

- *Administering Entrust/PKI on UNIX*

- *Transarc DCE User's Guide and Reference*

- *Transarc DCE Application Development Guide*

- *Transarc DCE Application Development Reference*

- *Transarc DCE Administration Guide*

- *Transarc DCE Administration Reference*

- *Transarc DCE Porting and Testing Guide*

- *Application Environment Specification/Distributed Computing*

- *Transarc DCE Technical Supplement*

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.	*Oracle9i 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, RMAN 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.

Convention	Meaning	Example
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as 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 open 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 monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the `parallel_clause`. Run `Uold_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 normal 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
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (digits [ , precision ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE	DISABLE}`
|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE	DISABLE}` `[COMPRESS	NOCOMPRESS]`
...	Horizontal ellipsis points indicate either:	
	■ That we have omitted parts of the code that are not directly related to the example	`CREATE TABLE ... AS subquery;`
	■ That you can repeat a portion of the code	`SELECT col1, col2, ... , coln FROM employees;`
. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	
Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points 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`

Convention	Meaning	Example

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. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		
		`SELECT * FROM USER_TABLES;`
		
		`DROP TABLE hr.employees;`

lowercase	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates 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 >	How to start a program. For example, to start Oracle Database Configuration Assistant, you must click the Start button on the taskbar and then choose Programs > Oracle - *HOME_NAME* > Database Administration > Database Configuration Assistant.	Choose Start > Programs > Oracle - *HOME_NAME* > Database Administration > Database Configuration Assistant
`C:\>`	Represents the Windows command prompt of the current hard disk drive. Your prompt reflects the subdirectory in which you are working. Referred to as the command prompt in this guide.	`C:\oracle\oradata>`
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 8.1, when you installed Oracle components, all subdirectories were located under a top level *ORACLE_HOME* directory that by default was: ■ `C:\orant` for Windows NT ■ `C:\orawin95` for Windows 95 ■ `C:\orawin98` for Windows 98 or whatever you called your Oracle home. In this Optimal Flexible Architecture (OFA)-compliant release, 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`. If you install release 8.1.7 on a computer with no other Oracle software installed, the default setting for the first Oracle home directory is `C:\oracle\ora81`. The Oracle home directory is located directly under *ORACLE_BASE*. All directory path examples in this guide follow OFA conventions. See *Oracle9i Database Getting Starting for Windows* for additional information on OFA compliances and for information on installing Oracle products in non-OFA compliant directories.	Go to the *ORACLE_BASE\ORACLE_HOME*\rdbms\admin directory.

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. 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. 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 additional information, visit the Oracle Accessibility Program Web site at

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

JAWS, a Windows screen reader, 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, JAWS may not always read a line of text that consists solely of a bracket or brace.

Part I

Introduction

This part introduces Oracle Advanced Security and describes its features. It contains the following chapter:

- Chapter 1, Introduction to Oracle Advanced Security

1

Introduction to Oracle Advanced Security

This chapter introduces Oracle Advanced Security and describes its features. These features are available to database and related products that interface with Oracle Net, including Oracle9*i*, Oracle Designer, and Oracle Developer.

This chapter contains the following topics:

- About Oracle Advanced Security
- Oracle Advanced Security Features
- Oracle Advanced Security Architecture
- Secure Data Transfer Across Network Protocol Boundaries
- System Requirements
- Oracle Advanced Security Restrictions

About Oracle Advanced Security

Oracle Advanced Security provides a comprehensive suite of security features to protect enterprise networks and securely extend corporate networks to the Internet. It provides a single source of integration with network encryption and authentication solutions, single sign-on services, and security protocols. By integrating industry standards, it delivers unparalleled security to the Oracle network.

This section contains the following topics:

- Security in an Intranet or Internet Environment
- Security Threats

Security in an Intranet or Internet Environment

Oracle databases power the largest and most popular web sites on the Internet. In record numbers, organizations throughout the world are deploying distributed databases and client/server applications based on Oracle9*i* and Oracle Net. This proliferation of distributed computing is matched by an increase in the amount of information that organizations place on computers. Employee and financial records, customer orders, product information, and other sensitive data have moved from filing cabinets to file structures. The volume of sensitive information on the web has thus increased the value of data that can be compromised.

Security Threats

The increased volume of data in distributed environments exposes users to a variety of security threats, including the following:

- Eavesdropping and Data Theft
- Data Tampering
- Falsifying User Identities
- Password-Related Threats

Eavesdropping and Data Theft

Over the Internet and in wide area network environments, both public carriers and private networks route portions of their network through insecure land lines, vulnerable microwave and satellite links, or a number of servers— exposing valuable data to interested third parties. In local area network environments within

a building or campus, the potential exists for insiders with access to the physical wiring to view data not intended for them, and network **sniffers** can be installed to eavesdrop on network traffic.

Data Tampering

Distributed environments bring with them the possibility that a malicious third party can compromise integrity by tampering with data as it moves between sites.

Falsifying User Identities

In a distributed environment, it is more feasible for a user to falsify an identity to gain access to sensitive information. How can you be sure that user *Pat* connecting to Server A from Client B really is user `Pat`?

Moreover, in distributed environments, malefactors can hijack connections. How can you be sure that Client B and Server A are what they claim to be? A transaction that should go from the Personnel system on Server A to the Payroll system on Server B could be intercepted in transit and re-routed to a terminal masquerading as Server B.

Password-Related Threats

In large systems, users typically must remember multiple passwords for the different applications and services that they use. For example, a developer can have access to a development application on a workstation, a PC for sending email, and several computers or intranet sites for testing, reporting bugs, and managing configurations.

Users typically respond to the problem of managing multiple passwords in several ways:

- They may select easy-to-guess passwords—such as a name, fictional character, or a word found in a dictionary. All of these passwords are vulnerable to **dictionary attacks**.

- They may also choose to standardize passwords so that they are the same on all machines or web sites. This results in a potentially large exposure in the event of a compromised password. They can also use passwords with slight variations that can be easily derived from known passwords.

- Users with complex passwords may write them down where an attacker can easily find them, or they may just forget them—requiring costly administration and support efforts.

All of these strategies compromise password secrecy and service availability. Moreover, administration of multiple user accounts and passwords is complex, time-consuming, and expensive.

Oracle Advanced Security Features

Oracle Advanced Security provides data privacy, integrity, authentication, single sign-on, and access authorization in a variety of ways.

For example, you can configure either Oracle Net native encryption or Secure Sockets Layer (SSL) for data privacy. Oracle Advanced Security also provides the choice of several strong authentication methods, including Kerberos, smart cards, and digital certificates.

Oracle Advanced Security features are described in the following sections:

- Data Privacy
- Data Integrity
- Authentication
- Single Sign-On
- Authorization

Data Privacy

Oracle Advanced Security protects the privacy of data transmissions through the following encryption methods:

- RC4 Encryption
- DES Encryption
- Triple-DES Encryption

Selection of the network encryption method is a user configuration option, providing varying levels of security and performance for different types of data transfers.

Prior versions of Oracle Advanced Security provided three editions: Domestic, Upgrade, and Export—each with different key lengths. Release 9.0.1 now contains a complete complement of the available encryption algorithms and key lengths, previously only available in the Domestic edition. Users deploying prior versions of the product can obtain the Domestic edition for a specific product release.

> **Note:** T*he U.S. government has relaxed its export guidelines for encryption products. Accordingly, Oracle can now ship Oracle Advanced Security with its strongest encryption features—to virtually all of its customers.*

RC4 Encryption

The RC4 encryption module uses the RSA Security, Inc. RC4 encryption algorithm. Using a secret, randomly-generated key unique to each session, all network traffic is fully safeguarded—including all data values, SQL statements, and stored procedure calls and results. The client, server, or both, can request or require the use of the encryption module to guarantee that data is protected. Oracle's optimized implementation provides a high degree of security for a minimal performance penalty. For the RC4 algorithm, Oracle provides encryption key lengths of 40-bits, 56-bits, 128-bits, and 256-bits.

DES Encryption

The U.S. Data Encryption Standard algorithm (DES) uses symmetric key cryptography to safeguard network communications. Oracle Advanced Security implements DES with a standard, optimized 56-bit key encryption algorithm, and also provides DES40, a 40-bit version, for backwards compatibility.

Triple-DES Encryption

Oracle Advanced Security also supports Triple-DES encryption (3DES), which encrypts message data with three passes of the DES algorithm. 3DES provides a high degree of message security, but with a performance penalty—the magnitude of which is dependant upon on the speed of the processor performing the encryption; 3DES typically takes three times as long to encrypt a data block as compared with the standard DES algorithm.

3DES is available in two-key and three-key versions, with effective key lengths of 112-bits and 168-bits, respectively. Both versions operate in outer **Cipher Block Chaining (CBC)** mode.

Federal Information Processing Standard

Oracle Advanced Security Release 8.1.6 has been validated under U.S. Federal Information Processing Standard 140-1 (FIPS) at the Level 2 security level. This provides independent confirmation that Oracle Advanced Security conforms to

federal government standards. FIPS configuration settings are described by Appendix D, Oracle Advanced Security FIPS 140-1 Settings.

> **See Also:**
>
> - Chapter 2, Configuring Data Encryption and Integrity
> - Appendix A, Data Encryption and Integrity Parameters

Data Integrity

To ensure the **integrity** of data packets during transmission, Oracle Advanced Security can generate a cryptographically secure message digest—using MD5 or SHA encryption algorithms—and include it with each message sent across a network.

Data integrity algorithms add little overhead, and protect against the following attacks:

- Data modification
- Deleted packets
- Replay attacks

> **Note:** SHA is slightly slower than MD5, but produces a larger message digest, making it more secure against brute-force collision and inversion attacks.

> **See Also:** Chapter 2, Configuring Data Encryption and Integrity, for information about MD5 and SHA.

Authentication

Authenticating user identity is imperative in distributed environments, without which there can be little confidence in network security. Passwords are the most common **authentication** method, and Oracle Advanced Security provides enhanced user authentication through several third-party authentication services, and through the use of SSL and digital certificates (See: Figure 1–1).

Many Oracle Advanced Security authentication methods use centralized authentication. This can give you high confidence in the identity of users, clients, and servers in distributed environments. Having a central facility authenticate all members of the network (clients to servers, servers to servers, users to both clients and servers) is one effective way to address the threat of nodes on a network falsifying their identities.

How Centralized Network Authentication Works

Figure 1–1 shows how a centralized network authentication service typically operates:

Figure 1–1   How a Network Authentication Service Authenticates a User
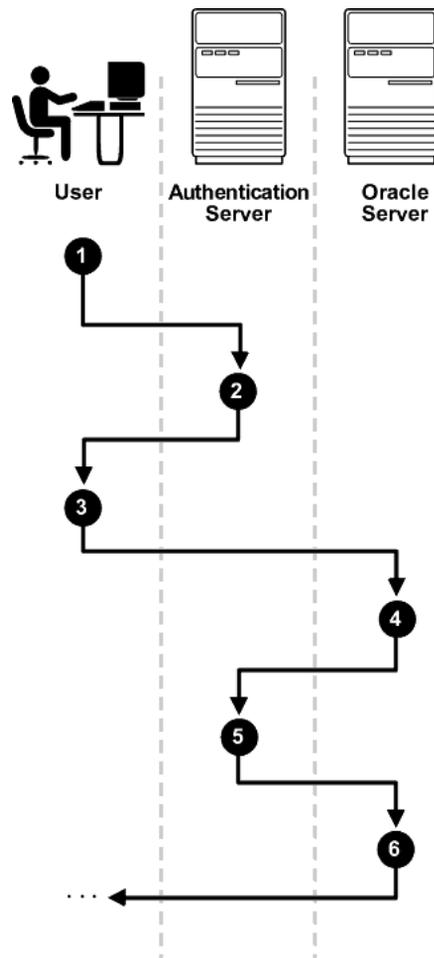
1. A user (client) requests authentication services and provides identifying information, such as a token or password.

2. The authentication server validates the user's identity and passes a ticket or credentials back to the client—which may include an expiration time.

3. The client passes these credentials to the Oracle server concurrent with a service request, such as connection to a database.

4. The server sends the credentials back to the authentication server for authentication.

5. If the authentication server accepts the credentials, it notifies the Oracle Server; the user is authenticated.

6. If the authentication server does not accept the credentials, authentication fails and the service request is denied.

Supported Authentication Methods

Oracle Advanced Security supports the following authentication methods:

- Secure Sockets Layer (with digital certificates)
- Entrust/PKI
- Remote Authentication Dial-In User Service
- Kerberos and CyberSafe
- Smart Cards
- Token Cards

Secure Sockets Layer

Secure Sockets Layer (SSL) is an industry standard protocol for securing network connections. SSL provides **authentication**, data **encryption**, and data **integrity**, and it contributes to a **public-key infrastructure** (PKI).

Oracle Advanced Security SSL can be used to secure communications between any client and any server. You can configure SSL to provide server authentication only, client authentication only, or both client and server authentication.

SSL uses digital certificates (X.509 v3), and a **public/private key pair** to authenticate users and systems.

SSL features can be used by themselves or in combination with other authentication methods supported by Oracle Advanced Security.

Entrust/PKI

Oracle Advanced Security supports the public key infrastructure (PKI) provided by the Entrust/PKI software from Entrust Technologies, Inc. Entrust-enabled Oracle Advanced Security lets Entrust users incorporate Entrust single sign-on into their Oracle applications, and it lets Oracle users incorporate Entrust-based single sign-on into Oracle applications.

Remote Authentication Dial-In User Service

Remote Authentication Dial-In User Service (RADIUS) is a client/server security protocol that is most widely known for enabling remote authentication and access. Oracle Advanced Security uses this standard in a client/server network environment to enable use of any authentication method that supports the RADIUS protocol. RADIUS can be used with a variety of authentication mechanisms, including token cards, smart cards, and Biometrics.

Kerberos and CyberSafe

Oracle Advanced Security support for Kerberos and CyberSafe provides the benefits of single sign-on and centralized authentication of Oracle users. Kerberos is a trusted third-party authentication system that relies on shared secrets. It presumes that the third party is secure, and provides single sign-on capabilities, centralized password storage, database link authentication, and enhanced PC security. It does this through a Kerberos authentication server, or through Cybersafe Active Trust, a commercial Kerberos-based authentication server.

> **Note:** Oracle authentication for Kerberos provides database link authentication (also called proxy authentication). CyberSafe does not support proxy authentication.

Smart Cards

A RADIUS-compliant smart card is a credit card-like hardware device. It has memory and a processor and is read by a smart card reader located at the client workstation.

Smart cards provide the following benefits:

Enhanced password security	Smart cards rely on two-factor authentication. The smart card can be locked, and only the user who (i) possesses the card and (ii) knows the correct personal identification number (PIN) can unlock it.
Improved performance	Some sophisticated smart cards contain hardware-based encryption chips that can provide better throughput than software-based implementations. A smart card can also store a user name.
Accessibility from any workstation	Users log in by inserting the smart card in a hardware device that reads the card and prompts the user for whatever authentication information the card requires, such as a PIN. Once the user enters the correct authentication information, the smart card generates and enters whatever other authentication information is required.
Ease of use	Users need only remember a PIN—instead of multiple passwords.

Token Cards

Token cards (SecurID or RADIUS-compliant) can improve ease of use through several different mechanisms. Some token cards dynamically display one-time passwords that are synchronized with an authentication service. The server can verify the password provided by the token card at any given time by contacting the authentication service. Other token cards have a keypad and operate on a challenge-response basis. In this case, the server offers a challenge (a number) that the user enters into a token card. The token card provides a response (another number cryptographically derived from the challenge) that the user enters and sends to the server.

Token cards provide the following benefits (Table 1–1):

Table 1–1   Token Card Benefits

Benefit	Description
Enhanced password security	To masquerade as a user, a malefactor must have the token card as well as the personal identification number (PIN) required to operate it. This is called two-factor authentication.
Ease of use	Users need only remember a PIN—instead of multiple passwords.
Enhanced accountability	Token cards provide a stronger authentication mechanism; users are thus more accountable for their actions.
Access from any workstation	Users can log on from any workstation using their PIN, which provides strong, two-factor authentication without any additional hardware devices.

You can use SecurID tokens through the RADIUS adapter.

Single Sign-On

Centralized authentication can enable a single, integrated user sign-on (**single sign-on**). This feature lets users access multiple accounts and applications with a single password, eliminates the need for multiple passwords, and simplifies management of user accounts and passwords for system administrators.

Oracle Advanced Security single sign-on authenticates the user once upon initial connection, with strong authentication occurring transparently in subsequent

connections to other databases or services. Using single sign-on, users can access multiple accounts and applications with a single password. Oracle Advanced Security supports many forms of single sign-on, including Kerberos and CyberSafe.

Oracle Advanced Security also provides SSL-based single sign-on for Oracle users by integrating with **LDAP** v3-compliant directory services. The combination of integrated directory services and Oracle's PKI implementation enable SSL-based single sign-on to Oracle9*i* databases. Single sign-on lets users be authenticated once, with subsequent connections relying on the user's digital certificate.

This enhances ease-of-use for users, and provides centralized management to security administrators.

Authorization

User authorization, a function of Oracle9*i* roles and privileges, is significantly enhanced by using the authentication methods supported by Oracle Advanced Security. For example, on certain operating systems, such as Solaris, Oracle Advanced Security supports authorization with DCE.

Authorizations are also provided by Oracle Advanced Security Enterprise User Security (See: Chapter 15, Managing Enterprise User Security). Oracle Advanced Security can integrate with LDAP version 3-compliant directories to centrally manage users and authorizations. Your Oracle Advanced Security license entitles you to deploy Oracle Internet Directory for user management as well as authorization storage and retrieval. You must license Oracle Internet Directory separately if you use it for additional purposes.

Oracle Advanced Security Architecture

Oracle Advanced Security is an add-on product that complements an Oracle server or client installation. Figure 1–2 shows the Oracle Advanced Security architecture within an Oracle networking environment.

Figure 1–2   Oracle Advanced Security in an Oracle Networking Environment
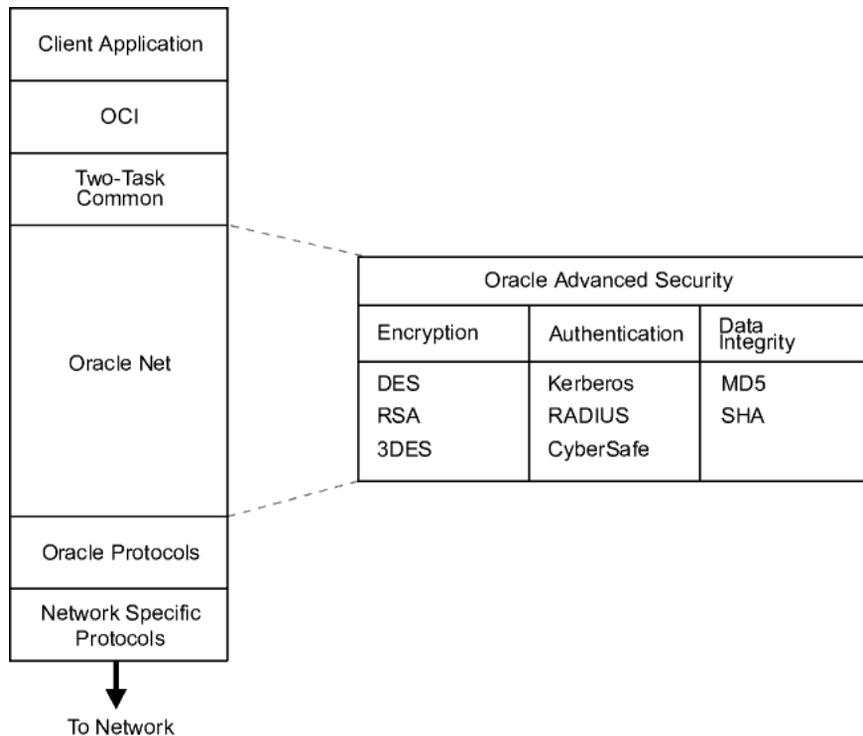
Oracle Advanced Security supports authentication through adapters that are similar to the existing Oracle protocol adapters. As shown in Figure 1–3, authentication adapters integrate below the Oracle Net interface and let existing applications take advantage of new authentication systems transparently, without any changes to the application.

Figure 1–3 Oracle Net with Authentication Adapters
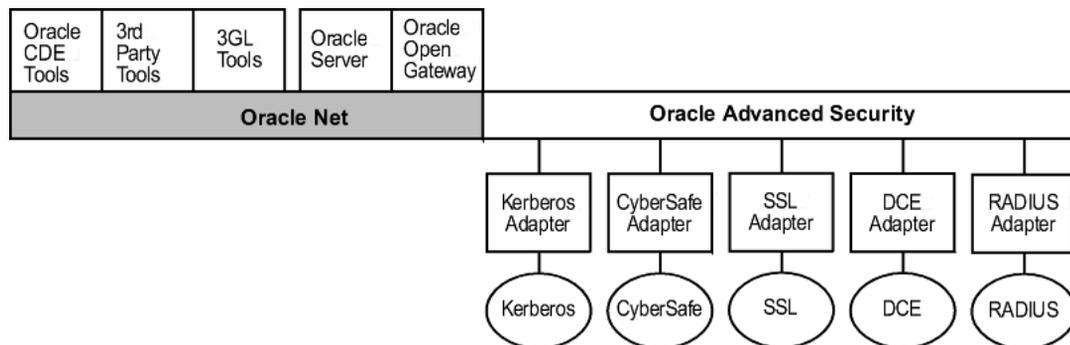
> **See Also:**   *Oracle Net Services Administrator's Guide,* for more
> information about stack communications in an Oracle networking
> environment

Secure Data Transfer Across Network Protocol Boundaries

Oracle Advanced Security is fully supported by Oracle Connection Manager, making secure data transfer a reality across network protocol boundaries. Clients using LAN protocols such as NetWare (SPX/IPX), for example, can securely share data with large servers using different network protocols such as LU6.2, TCP/IP, or DECnet. To eliminate potential weak points in the network infrastructure and to maximize performance, Connection Manager passes encrypted data from protocol to protocol without the cost and exposure of decryption and re-encryption.

System Requirements

Oracle Advanced Security is an add-on product bundled with the Oracle Net Server or Oracle Net Client. It must be purchased and installed on both the client and the server.

Oracle Advanced Security Release 9.0.1 requires Oracle Net Release 9.0.1 and supports Oracle9*i* Enterprise Edition. Table 1–2 lists additional system requirements.

> **Note:** Oracle Advanced Security is not *available* with Oracle9*i* Standard Edition.

Table 1–2   Authentication Methods and System Requirements

Authentication Method	System Requirements
Cybersafe Active Trust	■ CyberSafe GSS Runtime Library, version 1.1 or later, installed on both the machine that runs the Oracle client and on the machine that runs the Oracle server. ■ Cybersafe Active Trust, release 1.2 or later, installed on a physically secure machine that runs the authentication server. ■ Cybersafe Active Trust Client, release 1.2 or later, installed on the machine that runs the Oracle client.
Kerberos	■ MIT Kerberos Version 5, release 1.1 ■ The Kerberos authentication server must be installed on a physically secure machine.
RADIUS	■ A RADIUS server that is compliant with the standards in the Internet Engineering Task Force (IETF) RFC #2138, *Remote Authentication Dial In User Service (RADIUS)* and RFC #2139 *RADIUS Accounting*. ■ To enable challenge-response authentication, you must run RADIUS on an operating system that supports the Java Native Interface as specified in release 1.1 of the Java Development Kit from JavaSoft.
SSL	■ A wallet that is compatible with the Oracle Wallet Manager version 2.1. Wallets created in earlier releases of the Oracle Wallet Manager are not forward compatible.
Entrust/PKI	■ Entrust IPSEC Negotiator Toolkit Release 5.0.2 ■ Entrust/PKI 5.0.2

Oracle Advanced Security Restrictions

Oracle Applications support Oracle Advanced Security encryption and data integrity. However, because Oracle Advanced Security requires Oracle Net to transmit data securely, Oracle Advanced Security external authentication features are not supported by some parts of Oracle Financial, Human Resource, and Manufacturing Applications when they are running on Microsoft Windows. *The portions of these products that use Oracle Display Manager (ODM) do not take advantage of Oracle Advanced Security, since ODM does not use Oracle Net.*

Part II

Encryption, Integrity, and JDBC

This part describes how to configure data encryption and integrity for your existing Oracle network, and the Java implementation of Oracle Advanced Security. It contains the following chapters:

- Chapter 2, Configuring Data Encryption and Integrity
- Chapter 3, Thin JDBC Support

> **See Also:** Platform-specific documentation for your particular platform.

2

Configuring Data Encryption and Integrity

This chapter describes how to configure native Oracle Net data **encryption** and **integrity** for Oracle Advanced Security. It contains the following topics:

- Oracle Advanced Security Encryption
- Oracle Advanced Security Data Integrity
- Diffie-Hellman Based Key Management
- Configuring Data Encryption and Integrity

Oracle Advanced Security Encryption

This section describes data encryption algorithms available in the current release of Oracle Advanced Security:

- Overview
- DES Algorithm for Standards-Based Encryption
- Triple-DES Support
- RSA RC4 Algorithm for High Speed Encryption

> **Note:** Prior to Release 8.1.7, Oracle Advanced Security provided three editions: Domestic, Upgrade, and Export—each with different key lengths. This release now contains a complete complement of the available encryption algorithms and key lengths, previously only available in the Domestic edition. Users deploying prior versions of the product can obtain the Domestic edition for a specific product release.

Overview

The purpose of a secure cryptosystem is to convert **plaintext** data into unintelligible **ciphertext** based on a key, in such a way that it is very hard (computationally infeasible) to convert ciphertext back into its corresponding plaintext without knowledge of the correct key. In a symmetric cryptosystem, the same key is used both for encryption and decryption of the same data. Oracle Advanced Security provides the DES, 3DES, and RC4 symmetric cryptosystems for protecting the confidentiality of Oracle Net traffic.

DES Algorithm for Standards-Based Encryption

Oracle Advanced Security provides the Data Encryption Standard (DES) algorithm. DES has been a U.S. government standard for many years and is sometimes mandated in the financial services industry. Because it has been a standard for so long, DES is deployed throughout the world for use in a wide variety of applications.

Triple-DES Support

Oracle Advanced Security supports Triple-DES encryption (3DES), which encrypts message data with three passes of the DES algorithm. 3DES provides a high degree

of message security, but with a performance penalty—the magnitude of which is dependant upon on the speed of the processor performing the encryption; 3DES typically takes three times as long to encrypt a data block as compared with the standard DES algorithm.

3DES is available in two-key and three-key versions, with effective key lengths of 112-bits and 168-bits, respectively. Both versions operate in outer **Cipher Block Chaining (CBC)** mode.

DES40 Algorithm

The DES40 algorithm, available in every release of Oracle Advanced Security, Oracle Advanced Networking Option, and Secure Network Services, is a variant of DES in which the secret key is preprocessed to provide 40 effective key bits. It was designed to provide DES-based encryption to customers outside the U.S. and Canada at a time when the U.S. export laws were more restrictive. Now, in Oracle Advanced Security Release 9.0.1, DES40, DES, and 3DES are all available for export. DES40 is still supported to provide backward-compatibility for international customers.

RSA RC4 Algorithm for High Speed Encryption

The RC4 algorithm, developed by RSA Data Security Inc., has become the international standard for high-speed data encryption. RC4 is a variable key-length stream cipher that operates at several times the speed of DES, making it possible to encrypt large, bulk data transfers with minimal performance consequences.

Oracle Advanced Security Release 9.0.1 provides an RC4 implementation with 40-bit, 56-bit, 128-bit, and 256-bit key lengths. This provides backward-compatibility and strong encryption, with no material performance compromise.

See Also:

- Configuring Encryption on the Client and the Server on page 2-10.
- Table 2–2, Valid Encryption Algorithms on page 2-12.

Oracle Advanced Security Data Integrity

Encryption of network data provides data privacy, so that unauthorized parties are not able to view plaintext data as it passes over the network. Oracle Advanced Security also provides protection against two forms of active attack:

Data Modification Attack	An unauthorized party intercepts data in transit, alters it, and retransmits it. *Example: The monetary amount of $100 is changed to $10,000.*
Replay Attack	An entire set of valid data is repetitively retransmitted. *Example: A valid $100 withdrawal is resubmitted ten times.*

Data Integrity Algorithms Supported

Oracle Advanced Security lets you select a keyed, sequenced implementation of the Message Digest 5 (MD5) algorithm or the Secure Hash Algorithm (SHA-1) to protect against both of these forms of attack. Both of these hash algorithms create a checksum that changes if the data is altered in any way. This protection operates independently from the encryption process—you can enable data integrity with or without enabling encryption.

See Also:

- Configuring Integrity on the Client and the Server on page 2-12.
- Table 2–3, Valid Integrity Algorithms on page 2-14.

Diffie-Hellman Based Key Management

The secrecy of encrypted data depends upon the existence of a secret key shared between the communicating parties. A key is a secret exclusively shared by parties on both sides of a connection. Without the key, it is extremely difficult (computationally infeasible) to decrypt an encrypted message or to alter a cryptographic-checksummed message without detection. Providing and maintaining such secret keys is referred to as key management.

Secure key distribution is difficult in a multi-user environment. Oracle Advanced Security uses the well known **Diffie-Hellman key negotiation algorithm** to perform secure key distribution for both encryption and data integrity.

When encryption is used to protect the security of encrypted data, keys must be changed frequently to minimize the effects of a compromised key. Accordingly, the Oracle Advanced Security key management function changes the session key with every session.

Authentication Key Fold-in

The purpose of Authentication Key Fold-in is to defeat a possible third party attack (historically called the *man-in-the-middle attack*) on the Diffie-Hellman key negotiation. It strengthens the session key significantly by combining a shared secret, known only to the client and the server, with the original session key negotiated by Diffie-Hellman.

The client and the server begin communicating using the session key generated by Diffie-Hellman. When the client authenticates to the server, they establish a shared secret that is only known to both parties. Oracle Advanced Security combines the shared secret and the Diffie-Hellman session key to generate a stronger session key designed to defeat a man-in-the-middle attack.

> **Note:** The authentication key fold-in function is an imbedded feature of Oracle Advanced Security and requires no configuration by the system or network administrator.

Configuring Data Encryption and Integrity

This section describes how to configure Oracle Advanced Security native Oracle Net encryption and integrity, and presumes the prior installation of Oracle Net.

The network or security administrator sets up the encryption and integrity configuration parameters. The profile on client and server systems using data encryption and integrity (`sqlnet.ora file`) must contain some or all of the parameters listed in this section, under the following topics:

- Activating Encryption and Integrity

- Negotiating Encryption and Integrity

- Setting the Encryption Seed

- Configuring Encryption and Integrity Parameters Using Oracle Net Manager

> **See Also:** Chapter 7, Configuring Secure Sockets Layer Authentication, to configure the SSL feature for encryption, integrity, and authentication

Activating Encryption and Integrity

In any network connection, it is possible for both the client and server to each support more than one encryption algorithm and more than one integrity algorithm. When a connection is made, the server selects which algorithm to use, if any, from those algorithms specified in the `sqlnet.ora` files.

The server searches for a match between the algorithms available on both the client and the server, and picks the *first* algorithm in its own list that also appears in the client list. If one side of the connection does not specify an algorithm list, all the algorithms installed on that side are acceptable. The connection *fails* with error message `ORA-12650` if *either* side specifies an algorithm that is not installed.

Encryption and integrity parameters are defined by modifying a `sqlnet.ora` file on the clients and the servers on the network.

You can choose to configure any or all of the available Oracle Oracle Advanced Security encryption algorithms (Table 2–2), and either or both of the available integrity algorithms (Table 2–3). Only *one* encryption algorithm and *one* integrity algorithm are used for each connect session.

Note: Oracle Advanced Security selects the first encryption algorithm and the first integrity algorithm enabled on the client and the server. *Oracle Corporation recommends that you select algorithms and key lengths in the order in which you prefer negotiation—probably with the strongest key length first.*

See Also: Appendix A, Data Encryption and Integrity Parameters

Negotiating Encryption and Integrity

To negotiate whether to turn on encryption or integrity, you can specify four possible values for the Oracle Advanced Security encryption and integrity configuration parameters. The four values are listed in the order of increasing security. The value REJECTED provides the *minimum* amount of security between client and server communications, and the value REQUIRED provides the *maximum* amount of network security:

- REJECTED
- ACCEPTED
- REQUESTED
- REQUIRED

The default value for each of the parameters is ACCEPTED.

REJECTED

Select this value if you do not elect to enable the security service, even if required by the other side.

In this scenario, this side of the connection specifies that the security service is not permitted. If the other side is set to REQUIRED, the connection *terminates* with error message `ORA-12650`. If the other side is set to REQUESTED, ACCEPTED, or REJECTED, the connection continues without error and without the security service enabled.

ACCEPTED

Select this value to enable the security service if required or requested by the other side.

In this scenario, this side of the connection does not require the security service, but it is enabled if the other side is set to REQUIRED or REQUESTED. If the other side is set to REQUIRED or REQUESTED, and an encryption or integrity algorithm match is found, the connection continues without error and with the security service enabled. If the other side is set to REQUIRED and no algorithm match is found, the connection terminates with error message `ORA-12650`.

If the other side is set to REQUESTED and no algorithm match is found, or if the other side is set to ACCEPTED or REJECTED, the connection continues without error and without the security service enabled.

REQUESTED

Select this value to enable the security service if the other side permits it.

In this scenario, this side of the connection specifies that the security service is desired but not required. The security service is enabled if the other side specifies ACCEPTED, REQUESTED, or REQUIRED. There must be a matching algorithm available on the other side—otherwise the service is not enabled. If the other side specifies REQUIRED and there is no matching algorithm, *the connection fails.*

REQUIRED

Select this value to enable the security service or preclude the connection.

In this scenario, this side of the connection specifies that the security service *must be enabled.* The connection *fails* if the other side specifies REJECTED or if there is no compatible algorithm on the other side.

Table 2–1 shows whether the security service is enabled, based on a combination of client and server configuration parameters. If either the server or client has specified REQUIRED, the lack of a common algorithm *causes the connection to fail.* Otherwise, if the service is enabled, lack of a common service algorithm results in the service being *disabled.*

Table 2–1   Encryption and Data Integrity Negotiation

		Client			
		REJECTED	**ACCEPTED**	**REQUESTED**	**REQUIRED**
Server	**REJECTED**	OFF	OFF	OFF	Connection fails
	ACCEPTED	OFF	OFF[1]	ON	ON
	REQUESTED	OFF	ON	ON	ON
	REQUIRED	Connection fails	ON	ON	ON

[1] This value defaults to OFF. Cryptography and data integrity are not enabled until the user changes this parameter using Oracle Net Manager or by modifying the `sqlnet.ora` file.

Setting the Encryption Seed

Three seeds are used to generate a random number on the client and on the server. One of the seeds is a user-defined encryption seed (`sqlnet.crypto_seed=`) that can be 10 to 70 characters in length—and changed at any time. The Diffie-Hellman

key exchange uses the random numbers to generate unique session keys for every connect session.

Configuring Encryption and Integrity Parameters Using Oracle Net Manager

You can set up or change encryption and integrity parameter settings using Oracle Net Manager. This section describes the following topics:

- Configuring Encryption on the Client and the Server
- Configuring Integrity on the Client and the Server

    See Also:

    - Appendix A, Data Encryption and Integrity Parameters, for valid encryption algorithms
    - Oracle Net Manager online help, for more detailed configuration information

Configuring Encryption on the Client and the Server

To configure encryption on the client and on the server:

1. Start Oracle Net Manager:

    - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.
    - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right window pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears (Figure 2–1):

Figure 2–1   Oracle Advanced Security Encryption Window
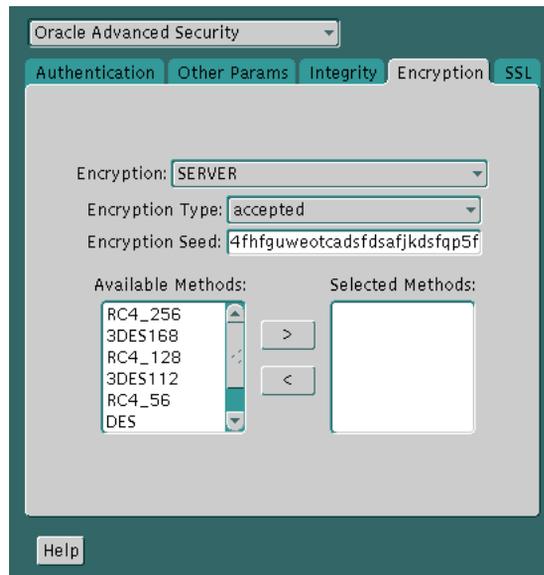
4. Choose the Encryption tab.

5. Depending upon which system you are configuring, select CLIENT or SERVER from the pull-down list.

6. From the Encryption Type list, select one of the following:

    - REQUESTED

    - REQUIRED

    - ACCEPTED

    - REJECTED

7. In the Encryption Seed field, enter between 10 and 70 random characters; the encryption seed for the client should not be the same as that for the server.

8. Select an encryption algorithm in the Available Methods list. Move it to the Selected Methods list by choosing the right arrow [>]. Repeat for each additional method you want to use.

9. Choose File > Save Network Configuration; the sqlnet.ora file is updated.

10. Repeat this procedure to configure encryption on the other system. The sqlnet.ora file on the two systems should contain the following entries:

- On the server:

```
SQLNET.ENCRYPTION_SERVER = [accepted | rejected | requested | required]
SQLNET.ENCRYPTION_TYPES_SERVER = (valid_encryption_algorithm [,valid_
encryption_algorithm])
SQLNET.CRYPTO_SEED = "10-70 random characters"
```

- On the client:

```
SQLNET.ENCRYPTION_CLIENT = [accepted | rejected | requested | required]
SQLNET.ENCRYPTION_TYPES_CLIENT = (valid_encryption_algorithm [,valid_
encryption_algorithm])
SQLNET.CRYPTO_SEED = "10-70 random characters"
```

Valid encryption algorithms and their associated legal values are summarized by Table 2–2:

Table 2–2   Valid Encryption Algorithms

Algorithm Name	Legal Value
RC4 256-bit key	RC4_256
RC4 128-bit key	RC4_128
RC4 56-bit key	RC4_56
RC4 40-bit key	RC4_40
3-key 3DES	3DES168
2-key 3DES	3DES112
DES 56-bit key	DES
DES 40-bit key	DES40

Configuring Integrity on the Client and the Server

To configure data integrity on the client and on the server:

1. Start Oracle Net Manager:

    - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

    - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears (Figure 2–2):

Figure 2–2   Oracle Advanced Security Integrity Window
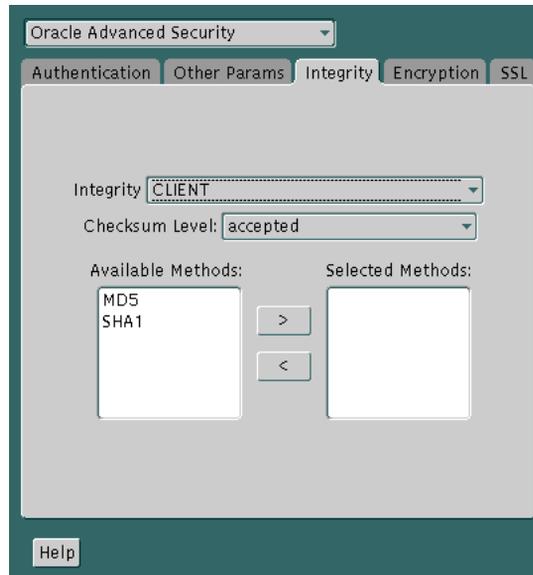
4. Choose the Integrity tab.

5. Depending upon which system you are configuring, choose the Server or Client check box.

6. From the Checksum Level list, select one of the following checksum level values:

   - REQUESTED
   - REQUIRED
   - ACCEPTED
   - REJECTED

7. Select an integrity algorithm in the Available Methods list. Move it to the Selected Methods list by choosing the right arrow [>]. Repeat for each additional method you want to use.

8. Choose File > Save Network Configuration; the `sqlnet.ora` file is updated.

9. Repeat this procedure to configure integrity on the other system. The `sqlnet.ora` file on the two systems should contain the following entries:

   - On the server:

     ```
     SQLNET.CRYPTO_CHECKSUM_SERVER = [accepted | rejected | requested |
     required]
     SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (valid_crypto_checksum_algorithm
     [,valid_crypto_checksum_algorithm])
     ```

   - On the client:

     ```
     SQLNET.CRYPTO_CHECKSUM_CLIENT = [accepted | rejected | requested |
     required]
     SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (valid_crypto_checksum_algorithm
     [,valid_crypto_checksum_algorithm])
     ```

Valid integrity algorithms and their associated legal values are displayed by Table 2–3:

Table 2–3   Valid Integrity Algorithms

Algorithm Name	Legal Values
MD5	MD5
SHA-1	SHA1

3

Thin JDBC Support

This chapter describes the Java implementation of Oracle Advanced Security, which lets Thin Java Database Connectivity (JDBC) clients securely connect to Oracle9*i* databases. This chapter contains the following topics:

- About the Java Implementation
- Configuration Parameters

> **See Also:** *Oracle9i JDBC Developer's Guide and Reference,* for information about JDBC, including examples

About the Java Implementation

The Java implementation of Oracle Advanced Security provides network encryption and integrity protection for Thin JDBC clients communicating with Oracle9*i* databases that have Oracle Advanced Security enabled.

This section contains the following topics:

- Java Database Connectivity Support
- Securing Thin JDBC
- Implementation Overview
- Obfuscation

Java Database Connectivity Support

Java Database Connectivity (JDBC), an industry-standard Java interface, is a Java standard for connecting to a relational database from a Java program. Sun Microsystems defined the JDBC standard and Oracle Corporation implements and extends the standard with its own JDBC drivers.

Oracle JDBC drivers are used to create JDBC applications to communicate with Oracle databases. Oracle implements two types of JDBC drivers: Thick JDBC drivers built on top of the C-based Oracle Net client, as well as a Thin (Pure Java) JDBC driver to support downloadable applets. Oracle extensions to JDBC include the following features:

- Data access and manipulation
- LOB access and manipulation
- Oracle object type mapping
- Object reference access and manipulation
- Array access and manipulation
- Application performance enhancement

Securing Thin JDBC

Because the Thin JDBC driver is designed to be used with downloadable applets used over the Internet, Oracle designed a 100% Java implementation of Oracle Advanced Security encryption and integrity algorithms for use with thin clients. Oracle Advanced Security provides the following features for Thin JDBC:

- Data encryption

- Data integrity checking

- Secure connections from Thin JDBC clients to the Oracle RDBMS

- Ability for developers to build applets that transmit data over a secure communication channel

- Secure connections from middle tier servers with Java Server Pages (JSP) to the Oracle RDBMS

- Secure connections from Oracle9*i* databases to older versions of Oracle databases with Oracle Advanced Security installed

The Oracle JDBC Thin driver implements the Oracle O3LOGON protocol for authentication. It does not support Oracle Advanced Security SSL implementation, nor does it support third party authentication features such as RADIUS, Kerberos, and SecurID. However, the Oracle JDBC OCI (thick) driver support is the same as thick client support, where all Oracle Advanced Security features are implemented.

Oracle Advanced Security continues to encrypt and provide integrity checking of Oracle Net traffic between Oracle Net clients and Oracle servers using algorithms written in C. The Oracle Advanced Security Java implementation provides Java versions of the following encryption algorithms:

- RC4_256

- RC4_128

- RC4_56

- RC4_40

- DES56

- DES40

> **Note:**   In Oracle Advanced Security, DES runs in Cipher Block Chaining (CBC) mode.

In addition, this implementation provides data integrity checking for Thin JDBC using Message Digest 5 (MD5), a cryptographically secure message digest.

Implementation Overview

On the server side, the negotiation of algorithms and the generation of keys function exactly the same as Oracle Advanced Security native encryption. This enables backward and forward compatibility of clients and servers.

On the client side, the algorithm negotiation and key generation occur in exactly the same manner as C-based Oracle Advanced Security encryption. The client and server negotiate encryption algorithms, generate random numbers, use Diffie-Hellman to exchange session keys, and use the Oracle Password Protocol (O3LOGON key fold-in), in the same manner as traditional Oracle Net clients. Thin JDBC contains a complete implementation of a Oracle Net client in pure Java.

Obfuscation

Java cryptography code is *obfuscated* in this release. Obfuscation protects Java classes and methods that contain encryption and decryption capabilities with obfuscation software.

Java byte code **obfuscation** is a process frequently used to protect intellectual property written in the form of Java programs. It mixes up Java symbols found in the code. The process leaves the original program structure intact, letting the program run correctly while changing the names of the classes, methods, and variables in order to hide the intended behavior. Although it is possible to decompile and read non-obfuscated Java code, obfuscated Java code is sufficiently difficult to decompile to satisfy U.S. government export controls.

Configuration Parameters

A properties class object containing several configuration parameters is passed to the Oracle Advanced Security interface. This chapter lists the configuration parameters for the following:

- Client Encryption Level
- Client Encryption Selected List
- Client Integrity Level
- Client Integrity Selected List

Client Encryption Level

Parameter Name	`oracle.net.encryption_client`
Description	Defines the level of security that the client wants to negotiate with the server
Parameter Type	String
Parameter Class	Static
Permitted Values	REJECTED; ACCEPTED; REQUESTED; REQUIRED
Default Value	ACCEPTED
Syntax	`up.put("oracle.net.encryption_client",level)`
Example	`up.put("oracle.net.encryption_client", "REQUIRED"), where up is defined as Properties up=new properties()`

Client Encryption Selected List

Parameter Name	`oracle.net.encryption_types_client`
Description	Defines the encryption algorithm to be used
Parameter Type	String
Parameter Class	Static
Permitted Values	RC4_256; RC4_128; RC4_56C; RC4_40; DES56C; DESC40C
Syntax	`up.put("oracle.net.encryption_types_` `client",alg)`
Example	`up.put("oracle.net.encryption_types_client",` `"DESC40C"), where up is defined as Properties` `up=new Properties()`

Note: In this context, "C" refers to CBC (Cipher Block Chaining) mode.

Client Integrity Level

Parameter Name	`oracle.net.crypto_checksum_client`
Description	Defines the level of security that it wants to negotiate with the server for data integrity
Parameter Type	String
Parameter Class	Static
Permitted Values	REJECTED; ACCEPTED; REQUESTED; REQUIRED
Default Value	ACCEPTED
Syntax	`up.put("oracle.net.crypto_checksum_` `client",level)`
Example	`up.put("oracle.net.crypto_checksum_client",` `"REQUIRED"), where up is defined as Properties` `up=new Properties()`

Client Integrity Selected List

Parameter Name	`oracle.net.crypto_cheksum_types_client`
Description	Defines the data integrity algorithm to be used
Parameter Type	String
Parameter Class	Static
Permitted Values	MD5; SHA
Syntax	`up.put("oracle.net.crypto_checksum_types_ client",alg)`
Example	`up.put("oracle.net.crypto_checksum_types_ client","MD5")`, where up is defined as Properties up=new Properties()

Part III

Configuring Authentication Methods

This part describes how to configure authentication methods into your existing Oracle network. It contains the following chapters, each of which describes a particular authentication method supported by Oracle Advanced Security:

- Chapter 4, Configuring RADIUS Authentication

- Chapter 5, Configuring CyberSafe Authentication

- Chapter 6, Configuring Kerberos Authentication

- Chapter 7, Configuring Secure Sockets Layer Authentication

- Chapter 8, Configuring Entrust-Enabled SSL Authentication

- Chapter 9, Configuring Multiple Authentication Methods

> **Note:** Oracle Advanced Security Release 9.0.1 supports dynamic loading of authentication methods. As a consequence, you no longer need to specify all possible authentication methods at install time; you can implement any available authentication method at any time subsequent to the initial installation of Oracle Advanced Security.

4

Configuring RADIUS Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle9*i,* or for the Oracle9*i* server, for use with RADIUS (Remote Authentication Dial-In User Service). This chapter contains the following topics:

- RADIUS Overview
- RADIUS Authentication Modes
- Enabling RADIUS Authentication and Accounting
- Using RADIUS to Log In to a Database

> **Note:** SecurID, an authentication product of Security Dynamics, Inc., though not directly supported by Oracle Advanced Security, has been certified as RADIUS-compliant. You can therefore run SecurID under RADIUS.
>
> See the Security Dynamics SecurID documentation for further information.

RADIUS Overview

RADIUS is a client/server security protocol widely used to enable remote authentication and access. Oracle Advanced Security uses this industry standard in a client/server network environment.

You can enable the network to use any authentication method that supports the RADIUS standard, including token cards and smart cards, by installing and configuring the RADIUS protocol. Moreover, when you use RADIUS, you can change the authentication method without modifying either the Oracle client or the Oracle database server.

From the user's perspective, the entire authentication process is transparent. When the user seeks access to an Oracle database server, the Oracle database server, acting as the RADIUS client, notifies the RADIUS server. The RADIUS server:

- Looks up the user's security information.

- Passes authentication and authorization information between the appropriate authentication server or servers and the Oracle database server.

- Grants the user access to the Oracle database server.

- Logs session information, including when, how often, and for how long the user was connected to the Oracle database server.

The Oracle/RADIUS environment is displayed in Figure 4–1:

Figure 4–1   RADIUS in an Oracle Environment
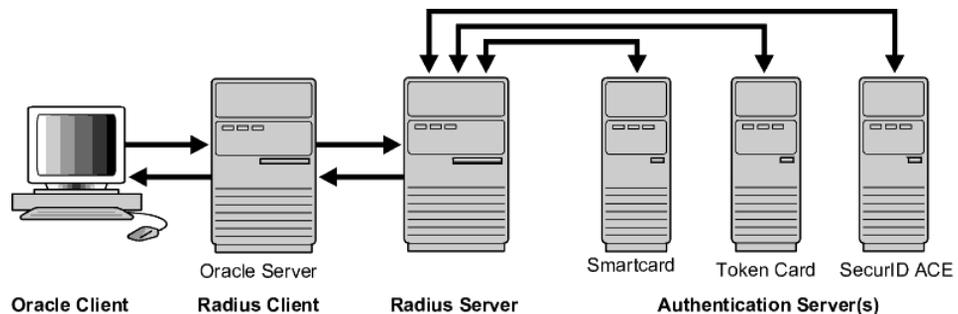
The Oracle database server acts as the RADIUS client, passing information between the Oracle client and the RADIUS server. Similarly, the RADIUS server passes

information between the Oracle database server and the appropriate authentication servers. The authentication components are listed in Table 4–1:

Table 4–1   RADIUS Authentication Components

Component	Stored Information
Oracle client	Configuration setting for communicating through RADIUS.
Oracle database server/ RADIUS client	Configuration settings for passing information between the Oracle client and the RADIUS server. The secret key file.
RADIUS server	Authentication and authorization information for all users. Each client's name or IP address. Each client's shared secret. Unlimited number of menu files enabling users already authenticated to select different login options without reconnecting.
Authentication server or servers	User authentication information such as passcodes and PINs, depending on the authentication method in use. **Note**: The RADIUS server can also be the authentication server.

A RADIUS server vendor is often the authentication server vendor as well, in which case authentication can be processed on the RADIUS server. For example, the Security Dynamics ACE/Server is both a RADIUS server and an authentication server. It thus authenticates the user's passcode.

> **See Also:**   *Oracle Net Services Administrator's Guide,* for information about the `sqlnet.ora` file

RADIUS Authentication Modes

User authentication can take place in either of two ways:

- Synchronous Authentication Mode
- Challenge-Response (Asynchronous) Authentication Mode

Synchronous Authentication Mode

In the synchronous mode, RADIUS lets you use various authentication methods, including passwords and SecurID token cards. Figure 4–2 shows the sequence in which synchronous authentication occurs:

Figure 4–2   Synchronous Authentication Sequence
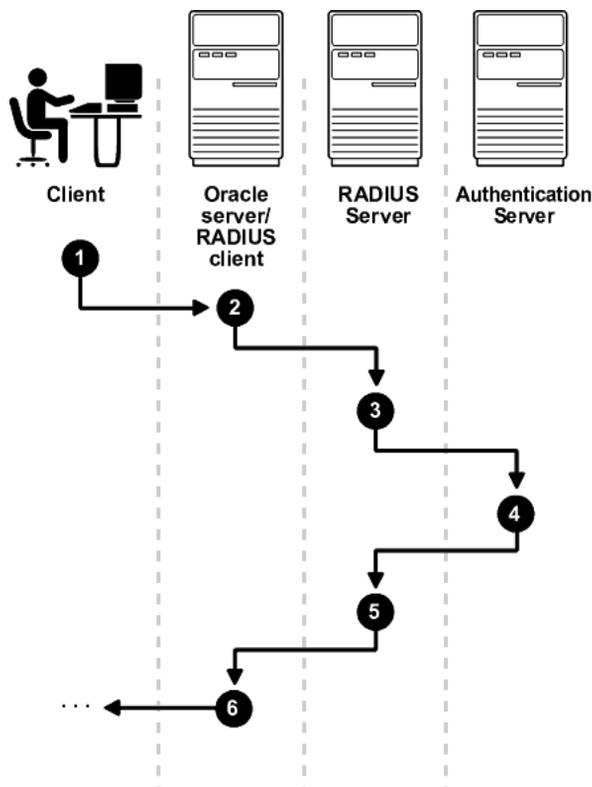
1. A user logs in by entering a connect string, passcode, or other value. The client system passes this data to the Oracle database server.

2. The Oracle database server, acting as the RADIUS client, passes the data from the Oracle client to the RADIUS server.

3. The RADIUS server passes the data to the appropriate authentication server, such as Smart Card or SecurID ACE for validation.

4. The authentication server sends either an Access Accept or an Access Reject message back to the RADIUS server.

5. The RADIUS server passes this response to the Oracle database server / RADIUS client.

6. The Oracle database server / RADIUS client passes the response back to the Oracle client.

Example: Synchronous Authentication with SecurID Token Cards

With SecurID authentication, each user has a token card that displays a dynamic number that changes every sixty seconds. To gain access to the Oracle database server/RADIUS client, the user enters a valid passcode that includes both a personal identification number (PIN) and the dynamic number currently displayed on the user's SecurID card. The Oracle database server passes this authentication information from the Oracle client to the RADIUS server, which in this case is the authentication server for validation. Once the authentication server (Security Dynamics ACE/Server) validates the user, it sends an "accept" packet to the Oracle database server, which, in turn, passes it to the Oracle client. The user is now authenticated and able to access the appropriate tables and applications.

> **See Also:**
> - Chapter 1, Introduction to Oracle Advanced Security
> - Token Cards on page 1-13
> - Documentation provided by Security Dynamics

Challenge-Response (Asynchronous) Authentication Mode

When the system uses the asynchronous mode, the user does not need to enter a user name and password at the SQL*Plus CONNECT string. Instead, a graphical user interface asks the user for this information later in the process.

Figure 4–3 shows the sequence in which challenge-response (asynchronous) authentication occurs.

> **Note:** If the RADIUS server is the authentication server, Steps 3, 4, and 5, and Steps 9, 10, and 11 in Figure 4–3 are combined.

Figure 4–3 Asynchronous Authentication Sequence
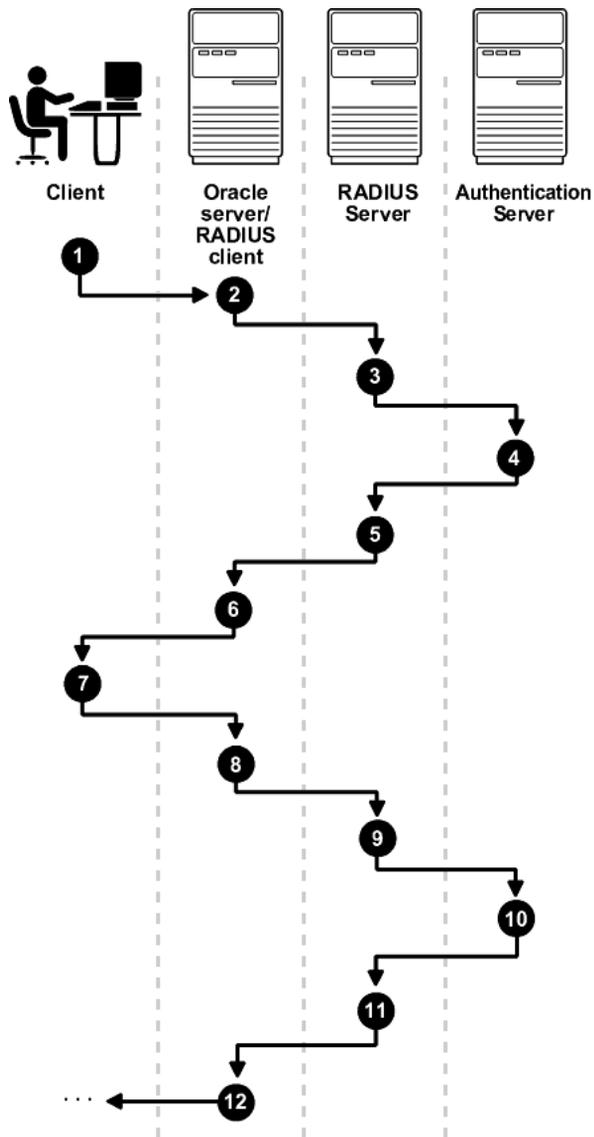
1. A user seeks a connection to an Oracle database server. The client system passes the data to the Oracle database server.

2. The Oracle database server, acting as the RADIUS client, passes the data from the Oracle client to the RADIUS server.

3. The RADIUS server passes the data to the appropriate authentication server, such as a Smart Card, SecurID ACE, or token card server.

4. The authentication server sends a challenge, such as a random number, to the RADIUS server.

5. The RADIUS server passes the challenge to the Oracle database server / RADIUS client.

6. The Oracle database server / RADIUS client, in turn, passes it to the Oracle client. A graphical user interface presents the challenge to the user.

7. The user provides a response to the challenge. To formulate a response, the user can, for example, enter the received challenge into the token card. The token card provides a dynamic password to be entered into the graphical user interface. The Oracle client passes the user's response to the Oracle database server / RADIUS client.

8. The Oracle database server / RADIUS client sends the user's response to the RADIUS server.

9. The RADIUS server passes the user's response to the appropriate authentication server for validation.

10. The authentication server sends either an Access Accept or an Access Reject message back to the RADIUS server.

11. The RADIUS server passes the response to the Oracle database server / RADIUS client.

12. The Oracle database server / RADIUS client passes the response to the Oracle client.

Example: Asynchronous Authentication with Smart Cards

With smart card authentication, the user logs in by inserting the smart card—a plastic card (like a credit card) with an embedded integrated circuit for storing information—into a hardware device which reads the card. The Oracle client sends the login information contained in the smart card to the authentication server by way of the Oracle database server/RADIUS client and the RADIUS server. The authentication server sends back a challenge to the Oracle client, by way of the RADIUS server and the Oracle database server, prompting the user for authentication information. The information could be, for example, a PIN as well as additional authentication information contained on the smart card.

The Oracle client sends the user's response to the authentication server by way of the Oracle database server and the RADIUS server. If the user has entered a valid number, the authentication server sends an "accept" packet back to the Oracle client by way of the RADIUS server and the Oracle database server. The user is now authenticated and authorized to access the appropriate tables and applications. If the user has entered incorrect information, the authentication server sends back a message rejecting the user's access.

Example: Asynchronous Authentication with ActivCard Tokens

One particular ActivCard token is a hand-held device with a keypad and which displays a dynamic password. When the user seeks access to an Oracle database server by entering a password, the information is passed to the appropriate authentication server by way of the Oracle database server/RADIUS client and the RADIUS server. The authentication server sends back a challenge to the client—by way of the RADIUS server and the Oracle database server. The user types that challenge into the token, and the token displays a number for the user to send in response.

The Oracle client then sends the user's response to the authentication server by way of the Oracle database server and the RADIUS server. If the user has typed a valid number, the authentication server sends an "accept" packet back to the Oracle client by way of the RADIUS server and the Oracle database server. The user is now authenticated and authorized to access the appropriate tables and applications. If the user has entered an incorrect response, the authentication server sends back a message rejecting the user's access.

Enabling RADIUS Authentication and Accounting

To enable RADIUS authentication and accounting, perform the following tasks:

- Task 1: Install RADIUS on the Oracle Database Server and on the Oracle Client
- Task 2: Configure RADIUS Authentication
- Task 3: Create a User and Grant Access
- Task 4: Configure RADIUS Accounting
- Task 5: Add the RADIUS Client Name to the RADIUS Server Database
- Task 6: Configure the Authentication Server for Use with RADIUS.
- Task 7: Configure the RADIUS Server for Use with the Authentication Server
- Task 8: Configure Mapping Roles

Task 1: Install RADIUS on the Oracle Database Server and on the Oracle Client

RADIUS is installed with Oracle Advanced Security during a typical installation of Oracle9*i*.

> **See:** Platform-specific installation documentation for Oracle9*i*, for information about installing Oracle Advanced Security and the RADIUS adapter

Task 2: Configure RADIUS Authentication

This task includes the following steps:

- Step 1: Configure RADIUS on the Oracle Client
- Step 2: Configure RADIUS on the Oracle Database Server
- Step 3: Configure Additional RADIUS Features

Unless otherwise indicated, perform these configuration tasks by using Oracle Net Manager or by using any text editor to modify the `sqlnet.ora` file.

Step 1: Configure RADIUS on the Oracle Client

1. To start Oracle Net Manager:
   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

- On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears (Figure 4–4):

Figure 4–4   Oracle Advanced Security Authentication Window
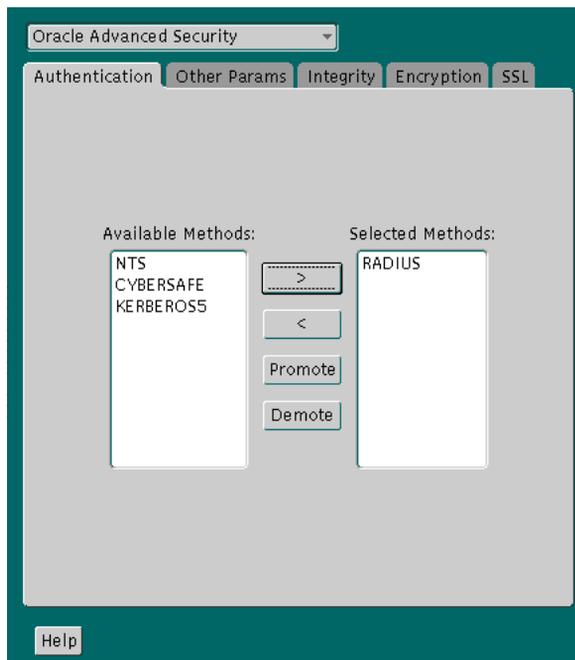
4. Choose the Authentication tab.

5. From the Available Methods list, select RADIUS.

6. Choose the right-arrow [>] to move RADIUS to the Selected Methods list. Move any other methods you want to use in the same way.

7. Arrange the selected methods in order of required usage by selecting a method in the Selected Methods list, and clicking Promote or Demote to position it in the list. For example, put RADIUS at the top of the list for it to be the first service used.

8. Choose File > Save Network Configuration.

The `sqlnet.ora` file is updated with the following entry:

```
SQLNET.AUTHENTICATION_SERVICES=(RADIUS)
```

Step 2: Configure RADIUS on the Oracle Database Server

- Create the RADIUS Secret Key File on the Oracle Database Server
- Configure RADIUS Parameters on the Server (sqlnet.ora file)
- Set Oracle Database Server Initialization Parameters

Create the RADIUS Secret Key File on the Oracle Database Server

1. Obtain the RADIUS secret key from the RADIUS server. For each RADIUS client, the administrator of the RADIUS server creates a shared secret key, which *must be longer than 16-characters.*

2. On the Oracle database server, create a directory `$ORACLE_HOME/network/security` on UNIX or `ORACLE_HOME\network\security` on Windows NT.

3. Create the file `radius.key` to hold the shared secret copied from the RADIUS server. Place the file in the directory you just created, namely, `$ORACLE_HOME/network/security` on UNIX or `ORACLE_HOME\network\security` on Windows NT.

4. Copy the shared secret key and paste it (and nothing else) into the `radius.key` file created on the Oracle database server.

5. For security purposes, change the file permission of `radius.key` to *read only,* accessible *only by the oracle owner (*Oracle relies on the file system to keep this file secret).

> **See Also:** The RADIUS server administration documentation, for information about obtaining the secret key

Configure RADIUS Parameters on the Server (`sqlnet.ora` file)

1. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears (Figure 4–4).

4. Choose the Authentication tab.

5. From the Available Methods list, select RADIUS.

6. Move RADIUS to the Selected Methods list by choosing the right-arrow [>].

7. To arrange the selected methods in order of desired use, select a method in the Selected Methods list, and choose Promote or Demote to position it in the list. For example, if you want RADIUS to be the first service used, put it at the top of the list.

8. Choose the Other Params tab; the Other Params window appears (Figure 4–5):

Figure 4–5   Oracle Advanced Security Other Params Window

9. From the Authentication Service list, select RADIUS.

10. In the Host Name field, accept the `localhost` as the default primary RADIUS server, or enter another host name.

11. Ensure that the default value of the Secret File field is valid.

12. Choose File > Save Network Configuration.

The `sqlnet.ora` file is updated with the following entries:

SQLNET.AUTHENTICATION_SERVICES=service

SQLNET.RADIUS_AUTHENTICATION=location

where *service* is RADIUS and *location* is the host name or IP address of the RADIUS server.

Set Oracle Database Server Initialization Parameters

Configure the initialization parameter file, located in `$ORACLE_BASE\admin\`*db_name*`\pfile` on UNIX and *ORACLE_BASE*`/admin/`*db_name*`/pfile` on Windows NT, with the following values:

```
REMOTE_OS_AUTHENT=FALSE
OS_AUTHENT_PREFIX=""
```

> **Caution:**   Setting REMOTE_OS_AUTHENT to TRUE can enable a security breach because it lets someone using a non-secure protocol, such as TCP, perform an operating system-authorized login (formerly called an OPS$ login).

> **See Also:**   *Oracle9i Database Reference* and the *Oracle9i Database Administrator's Guide,* for information about setting initialization parameters on the Oracle9i database server

Step 3: Configure Additional RADIUS Features

- Change Default Settings

- Configure Challenge-Response

- Set Parameters for an Alternate RADIUS Server

Change Default Settings

1. To start Oracle Net Manager:

    - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

    - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right window pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears (Figure 4–5).

4. Choose the Other Params tab.

5. From the Authentication Service list, select RADIUS.

6. Change the default setting for any of the following fields:

Field	Description
Port Number	Specifies the listening port of the primary RADIUS server. The default value is 1645.
Timeout (seconds)	Specifies the time the Oracle database server waits for a response from the primary RADIUS server. The default is 15 seconds.
Number of Retries	Specifies the number of times the Oracle database server resends messages to the primary RADIUS server. The default is three retries. For instructions on configuring RADIUS accounting, see: Task 4: Configure RADIUS Accounting on page 4-19.
Secret File	Specifies the location of the secret key on the Oracle database server. The field specifies the location of the secret key file, not the secret key itself. For information about specifying the secret key, see: Create the RADIUS Secret Key File on the Oracle Database Server on page 4-12.

7. Choose File > Save Network Configuration.

The `sqlnet.ora` file is updated with the following entries:

```
SQLNET.RADIUS_AUTHENTICATION_PORT=(PORT)
SQLNET.RADIUS_AUTHENTICATION_TIMEOUT=
(NUMBER OF SECONDS TO WAIT FOR response)
SQLNET.RADIUS_AUTHENTICATION_RETRIES=
(NUMBER OF TIMES TO RE-SEND TO RADIUS server)
SQLNET.RADIUS_SECRET=(path/radius.key)
```

Configure Challenge-Response

The challenge-response (asynchronous) mode presents the user with a graphical interface requesting first a password, then additional information—for example, a dynamic password that the user obtains from a token card. With the RADIUS adapter, this interface is Java-based to provide optimal platform independence.

> **Note:** Third party vendors of authentication devices must customize this graphical user interface to fit their particular device. For example, a smart card vendor would customize the Java interface so that the Oracle client reads data, such as a dynamic password, from the smart card. When the smart card receives a challenge, it responds by prompting the user for more information, such as a PIN.

> **See Also:** Appendix C, Integrating Authentication Devices Using RADIUS, for information about how to customize the challenge-response user interface

To configure challenge-response:

1. If you are using JDK 1.1.7 or JRE 1.1.7, set the JAVA_HOME environment variable to the JRE or JDK location on the system where the Oracle client is run:

   - On UNIX, enter this command at the prompt:

     ```
     % setenv JAVA_HOME /usr/local/packages/jre1.1.7B
     ```

   - On Windows NT, choose Start> Settings > Control Panel > System > Environment, and set the JAVA_HOME variable as follows:

     ```
     c:\java\jre1.1.7B
     ```

   > **Note:** This step is not required for any other JDK / JRE version.

2. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

3. In the Navigator window, expand Local > Profile.

4. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security Other Params window appears (Figure 4–5).

5. Choose the Other Params tab.

6. From the Authentication Service list, select RADIUS.

7. In the Challenge Response field, enter ON to enable challenge-response.

8. In the Default Keyword field, accept the default value of the challenge or enter a keyword for requesting a challenge from the RADIUS server.

> **Note:** The keyword feature is provided by Oracle and supported by some, but not all, RADIUS servers. You can use this feature only if your RADIUS server supports it.
>
> By setting a keyword, you let the user avoid using a password to verify identity. If the user does not enter a password, the keyword you set here is passed to the RADIUS server which responds with a challenge requesting, for example, a driver's license number or birth date. If the user does enter a password, the RADIUS server may or may not respond with a challenge, depending upon the configuration of the RADIUS server.

9. In the Interface Class Name field, accept the default value of `DefaultRadiusInterface` or enter the name of the class you have created to handle the challenge-response conversation. If other than the default RADIUS interface is used, you also must edit the `sqlnet.ora` file to enter `SQLNET.RADIUS_CLASSPATH=(location)`, where `location` is the complete pathname of the jar file. It defaults to `$ORACLE_HOME/network/jlib/netradius.jar: $oracle_home/JRE/lib/vt.jar`

10. Choose File > Save Network Configuration.

The `sqlnet.ora` file is updated with the following entries:

```
SQLNET.RADIUS_CHALLENGE_RESPONSE=([ON | OFF])
SQLNET.RADIUS_CHALLENGE_KEYWORD=(KEYWORD)
SQLNET.RADIUS_AUTHENTICATION_INTERFACE=(name of interface including the
package name delimited by "/" for ".")
```

Set Parameters for an Alternate RADIUS Server

If you are using an alternate RADIUS server, set these parameters in the `sqlnet.ora` file using any text editor.

SQLNET.RADIUS_ALTERNATE=(*hostname or ip address of alternate radius server*)

SQLNET.RADIUS_ALTERNATE_PORT=(*1812*)

SQLNET.RADIUS_ALTERNATE_TIMEOUT=(*number of seconds to wait for response*)

SQLNET.RADIUS_ALTERNATE_RETRIES=(*number of times to re-send to radius server*)

Task 3: Create a User and Grant Access

To grant user access:

1. Launch SQL*Plus and execute these commands to create and grant access to a user identified externally on the Oracle database server.

```
SQL> CONNECT system/manager@database_name;
SQL> CREATE USER username IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO USER username;
SQL> EXIT
```

If you are using Windows NT, you can use the Security Manager tool in the Oracle Enterprise Manager.

> **See Also:**
>
> - *Oracle9i Database Administrator's Guide*
>
> - *Oracle9i Heterogeneous Connectivity Administrator's Guide*

2. Enter the same user in the RADIUS server's users file.

> **See Also:** Administration documentation for the RADIUS server

Task 4: Configure RADIUS Accounting

RADIUS accounting logs information about access to the Oracle database server and stores it in a file on the RADIUS accounting server. Use this feature only if both the RADIUS server and authentication server support it.

Set RADIUS Accounting on the Oracle Database Server

To enable or disable RADIUS accounting:

1. Start Oracle Net Manager:

    -   On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

    -   On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* >
        Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right window pane, select Oracle Advanced Security; the
    Oracle Advanced Security Other Params window appears (Figure 4–5).

4. Choose the Other Params tab.

5. From the Authentication Service list, select RADIUS.

6. In the Send Accounting field, enter ON to enable accounting or OFF to disable
    accounting.

7. Choose File > Save Network Configuration.

    The `sqlnet.ora` file is updated with the following entry:

    ```
    SQLNET.RADIUS_SEND_ACCOUNTING= ON
    ```

Configure the RADIUS Accounting Server

RADIUS Accounting consists of an accounting server residing on either the same
host as the RADIUS authentication server or on a separate host.

> **See Also:** Administration documentation for the RADIUS server,
> for information about configuring RADIUS accounting

Task 5: Add the RADIUS Client Name to the RADIUS Server Database

You can use virtually any RADIUS server that complies with the standards in the
Internet Engineering Task Force (IETF) RFC #2138, *Remote Authentication Dial In
User Service (RADIUS)* and RFC #2139 *RADIUS Accounting*. Because RADIUS
servers vary, consult the documentation for your particular RADIUS server for any
unique interoperability requirements.

Perform the following steps to add the RADIUS client name to a Livingston
RADIUS server:

1. Open the clients file, which can be found at `/etc/raddb/clients`. The following text and table appear:

```
@ (#) clients 1.1 2/21/96 Copyright 1991 Livingston Enterprises Inc
This file contains a list of clients which are allowed to make
authentication requests and their encryption key. The first field is a valid
hostname. The second field (separated by blanks or tabs) is the encryption
key.
Client Name                     Key
```

2. In the CLIENT NAME column, enter the host name or IP address of the host on which the Oracle database server is running. In the KEY column, type the shared secret.

   The value you enter in the CLIENT NAME column, whether it is the client's name or IP address, depends on the RADIUS server.

3. Save and close the clients file.

> **See Also:** Administration documentation for the RADIUS server

Task 6: Configure the Authentication Server for Use with RADIUS

See the authentication server documentation for instructions about configuring the authentication servers. Related Documentation on page -xxvi contains a list of possible resources.

Task 7: Configure the RADIUS Server for Use with the Authentication Server

See the RADIUS server documentation.

Task 8: Configure Mapping Roles

If the RADIUS server supports vendor type attributes, you can manage roles by storing them in the RADIUS server. The Oracle database server downloads the roles when there is a CONNECT request using RADIUS.

To use this feature, configure roles on both the Oracle database server and the RADIUS server.

Perform these steps to configure roles on the Oracle database server:

1. Use a text editor to set the OS_ROLES parameter in the initialization parameters file on the Oracle database server.

2. Stop and restart the Oracle database server.

3. Create each role the RADIUS server is to manage on the Oracle database server with IDENTIFIED EXTERNALLY.

   To configure roles on the RADIUS server, refer to Table 4–1 and use the following syntax:

   ```
   ORA_DatabaseName.DatabaseDomainName_RoleName
   ```

   Example:

   ```
   ORA_USERDB.US.ORACLE.COM_MANAGER
   ```

Table 4-1 RADIUS Configuration Parameters

Parameter	Description
DatabaseName	The name of the Oracle database server for which the role is being created. This is the same as the value of the DB_NAME initialization parameter.
DatabaseDomainName	The name of the domain to which the Oracle database server belongs. The value is the same as the value of the DB_DOMAIN initialization parameter.
RoleName	The name of the role created in the Oracle database server.

4. Configure RADIUS challenge-response mode.

   To configure challenge-response mode, See:

   - Challenge-Response (Asynchronous) Authentication Mode on page 4-5
   - Configure Challenge-Response on page 4-17

Using RADIUS to Log In to a Database

If you are using the synchronous authentication mode, launch SQL*Plus and enter the following command at the prompt:

```
CONNECT username/password@database_alias
```

Note that you can log in with this command only when challenge-response is not turned to ON.

If you are using the challenge-response mode, launch SQL*Plus and, at the prompt, enter the command that follows:

```
CONNECT /@database_alias
```

Note that you can log in with this command only when challenge-response is turned to ON.

> **Note:** The challenge-response mode can be configured for all login cases.

5

Configuring CyberSafe Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle9*i*, or for the Oracle9*i* server, so that CyberSafe TrustBroker, a Kerberos-based authentication server, can be used to authenticate Oracle users. This chapter contains the following topics:

- Configuring CyberSafe Authentication
- Troubleshooting

Configuring CyberSafe Authentication

To configure CyberSafe authentication:

- Task 1: Install the CyberSafe Server
- Task 2: Install the CyberSafe TrustBroker Client
- Task 3: Install the CyberSafe Application Security Toolkit
- Task 4: Configure a Service Principal for an Oracle Database Server
- Task 5: Extract the Service Table from CyberSafe
- Task 6: Install an Oracle Database Server
- Task 7: Install Oracle Advanced Security With CyberSafe
- Task 8: Configure Oracle Net and Oracle9i
- Task 9: Configure CyberSafe Authentication
- Task 10: Create a CyberSafe User on the Authentication Server
- Task 11: Create an Externally Authenticated Oracle User on the Oracle Database Server
- Task 12: Get the Initial Ticket for the CyberSafe/Oracle User
- Task 13: Connect to an Oracle Database Server Authenticated by CyberSafe

Task 1: Install the CyberSafe Server

Perform this task on the system that functions as the authentication server.

> **See Also:** CyberSafe documentation listed under Related Documentation on page -xxvi

Task 2: Install the CyberSafe TrustBroker Client

Perform this task on the system that runs the Oracle database server and the client.

> **See Also:** CyberSafe documentation listed under Related Documentation on page -xxvi

Task 3: Install the CyberSafe Application Security Toolkit

Perform this task on both the client and server systems.

> **See Also:** CyberSafe documentation listed under Related
> Documentation on page -xxvi

Task 4: Configure a Service Principal for an Oracle Database Server

For the Oracle database server to validate the identity of clients, configure a **service principal** for an Oracle database server on the system running the CyberSafe TrustBroker Master Server. If required, also configure a realm.

The name of the principal has the following format:

```
kservice/kinstance@REALM
```

kservice	A case-sensitive string that represents the Oracle service. This might not be the same as the database service name
kinstance	Typically, this is the fully-qualified name of the system on which Oracle is running
REALM	The domain name of the server. REALM must always be uppercase, and is typically named the DNS domain name. If you do not enter a value for REALM when using `xst`, `kdb5_edit` uses the realm of the current host and displays it in the command output.

> **Note:** The utility names in this section are executable programs. However, the CyberSafe user name CYBERUSER and the realm `SOMECO.COM` are examples only.

For example, if the Oracle service is oracle, the fully-qualified name of the system on which Oracle is running is `dbserver.someco.com`, and the realm is `SOMECO.COM`, the principal name is:

```
oracle/dbserver.someco.com@SOMECO.COM
```

Run `kdb5_edit` as root to create the service principal as follows:

```
# cd /krb5/admin
# ./kdb5_edit
```

To add a principal named `oracle/dbserver.someco.com@SOMECO.COM` to the list of server principals known by CyberSafe, enter the following in `kdb5_edit`:

```
kdb5_edit:  ark oracle/dbserver.someco.com@SOMECO.COM
```

Task 5: Extract the Service Table from CyberSafe

Extract a service table from CyberSafe and copy it to both the Oracle database server and CyberSafe TrustBroker client systems.

For example, to extract a service table for `dbserver.someco.com`, perform the following steps.

1. Enter the following in `kdb5_edit`:

    ```
    kdb5_edit:  xst dbserver.someco.com oracle
    'oracle/dbserver.someco.com@SOMECO.COM' added to keytab
    'WRFILE:dbserver.someco.com-new-srvtab'

    kdb5_edit:  exit

    # /krb5/bin/klist -k -t dbserver.someco.com-new-srvtab
    ```

    If you do not enter a realm (`SOMECO.COM` in the example) when using `xst`, `kdb5_edit` uses the realm of the current host and displays it in the command output, as shown in the proceeding input example.

2. After the service table has been extracted, verify that the new entries are in the table, in addition to the old entries. If the new entries are not in the service table, or if you need to add additional new entries, use `kdb5_edit` to append them.

3. Move the CyberSafe service table to the CyberSafe TrustBroker client system. If the service table is on the same system as the CyberSafe client, move it as in the following example:

    ```
    # mv dbserver.someco.com-new-srvtab /krb5/v5srvtab
    ```

    If the service table is on a different system from the CyberSafe TrustBroker client, transfer the file with a program such as FTP. If using FTP, transfer the file in binary mode.

4. Ensure that the owner of the Oracle database server executable can read the service table (in the previous example, `/krb5/v5srvtab`). Set the file owner to

the Oracle user, or make the file readable by the group to which Oracle belongs. Do not make the file readable to all users—this can enable a security breach.

Task 6: Install an Oracle Database Server

Install an Oracle database server on the same system that is running the CyberSafe TrustBroker client.

> **See Also:** Oracle9*i* installation documentation for your platform

Task 7: Install Oracle Advanced Security With CyberSafe

Install CyberSafe, along with Oracle Advanced Security, during a custom installation of Oracle9*i*. The Oracle Universal Installer guides you through the entire installation process.

> **See Also:** Oracle9*i* installation documentation for your platform

Task 8: Configure Oracle Net and Oracle9*i*

Configure Oracle Net and Oracle9*i* on both the server and client systems.

> **See Also:** Oracle9*i* installation documentation for your platform

Task 9: Configure CyberSafe Authentication

Perform the following tasks to set parameters in the Oracle database server and client `sqlnet.ora` files to configure CyberSafe:

- Configure CyberSafe on both the Client and the Oracle Database Server
- Set REMOTE_OS_AUTHENT in the Initialization Parameter File (`init.ora`).

Configure CyberSafe on both the Client and the Oracle Database Server

To configure CyberSafe authentication service parameters on both the client and the database server:

1. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security Authentication window appears (Figure 5–1):

Figure 5–1   Oracle Advanced Security Authentication Window (Cybersafe)
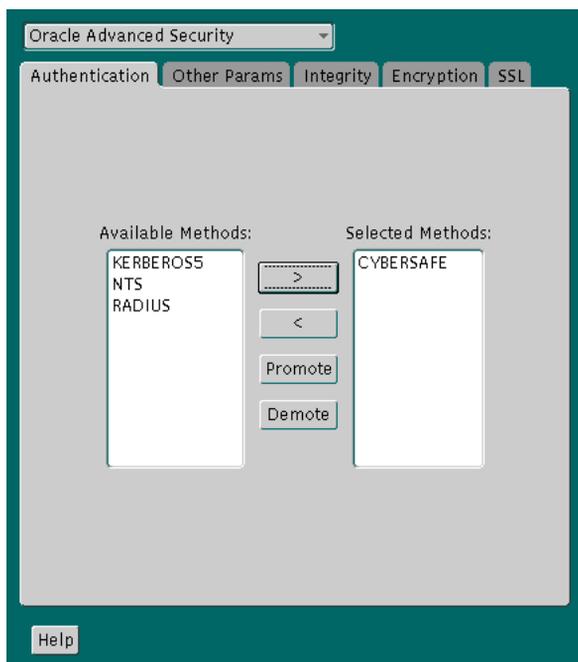
4. Choose the Authentication tab.

5. In the Available Methods list, select CYBERSAFE.

6. Move CYBERSAFE to the Selected Methods list by choosing the right-arrow [>].

7. Arrange the selected methods in order of desired use. To do this, select a method from the Selected Methods list and choose Promote or Demote to position it in the list. For example, if you want CYBERSAFE to be the first service used, put it at the top of the list.

8. Choose the Other Params tab (Figure 5–2):

Figure 5–2   Oracle Advanced Security Other Params Window (Cybersafe)
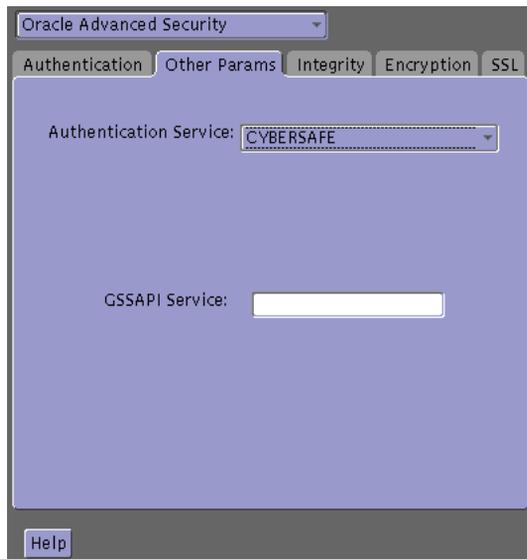
9. From the Authentication Service list, select CYBERSAFE.

10. Enter the name of the GSSAPI Service, as in the following example:

    ```
    oracle/dbserver.someco.com@SOMECO.COM
    ```

    Insert the principal name, using the format described in Task 4: Configure a Service Principal for an Oracle Database Server.

11. Choose File > Save Network Configuration.

    The sqlnet.ora file is updated with the following entries:

    ```
    SQLNET.AUTHENTICATION_SERVICES=(CYBERSAFE)
    SQLNET.AUTHENTICATION_GSSAPI_SERVICE=KSERVICE/KINSTANCE@REALM
    ```

Set REMOTE_OS_AUTHENT in the Initialization Parameter File

Add the following parameter to the Initialization Parameter File (init.ora):

```
REMOTE_OS_AUTHENT=FALSE
```

> **Note:** Setting REMOTE_OS_AUTHENT to TRUE can enable a
> security breach because it lets someone using a non-secure protocol,
> such as TCP, perform an operating system-authorized login
> (formerly called an OPS$ login).

Because CyberSafe user names can be long, and Oracle user names are limited to 30 characters, Oracle Corporation recommends using *null* for the value of OS_AUTHENT_PREFIX, as follows:

```
OS_AUTHENT_PREFIX=""
```

Restart the Oracle database server after modifying the configuration files to enable the changes.

> **See Also:** Operating system specific documentation and *Oracle9i Database Administrator's Guide* for more information about how to restart the Oracle database server

Task 10: Create a CyberSafe User on the Authentication Server

For CyberSafe to authenticate Oracle users, you must create them on the CyberSafe authentication server where the administration tools are installed. The following steps assume that the realm already exists.

> **Note:** The utility names in this section are executable programs.
> However, the CyberSafe user name CYBERUSER and realm
> SOMECO.COM are examples only.

Run /krb5/admin/kdb5_edit as root on the authentication server to create the new CyberSafe user, such as CYBERUSER.

Enter the following:

```
# kdb5_edit
kdb5_edit:
```

```
ank cyberuser
Enter password:
<password> (password does not display)
Re-enter password for verification:
<password> (password does not display)
kdb5_edit: quit
```

> **See Also:** Cybersafe documentation listed in Related Documentation on page -xxvi for information about creating the realm

Task 11: Create an Externally Authenticated Oracle User on the Oracle Database Server

Run SQL*Plus to create the Oracle user, and enter the following commands on the Oracle database server (*note that the Oracle user name must be uppercase and enclosed in double quotation marks):*

In this example, OS_AUTHENT_PREFIX is set to *null* ("").

```
SQL> CONNECT / AS SYSDBA;
SQL> CREATE USER "CYBERUSER@SOMECO.COM" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CYBERUSER@SOMECO.COM";
```

> **See Also:** *Oracle9i Database Administrator's Guide*

Task 12: Get the Initial Ticket for the CyberSafe/Oracle User

Before users can connect to the database, they must run `kinit` on the clients for an **initial ticket**:

1. Enter the following:

   ```
   % kinit cyberuser
   ```

2. Enter the password *(password does not display).*

3. To list currently owned tickets, run `klist` on the clients. Enter the following at the system command prompt:

   ```
   % klist
   ```

   The system displays the following information:

Creation Date	Expiration Date	Service
11-Aug-99 16:29:51	12-Aug-99 00:29:21	krbtgt/SCMECO.COM@SOMECO.COM
11-Aug-99 16:29:51	12-Aug-99 00:29:21	oracle/dbserver.someco.com@SOMECO.COM

Task 13: Connect to an Oracle Database Server Authenticated by CyberSafe

After running `kinit` to get an initial ticket, users can connect to an Oracle database server without using a user name or password. Enter a command similar to the following:

```
% sqlplus /@net_service_name
```

where `net_service_name` is a Oracle Net service name.

For example:

```
% sqlplus /@npddoc_db
```

> **See Also:**   Chapter 1, Introduction to Oracle Advanced Security, and *Oracle9i Heterogeneous Connectivity Administrator's Guide*

Troubleshooting

This section describes some common configuration problems and explains how to resolve them:

If you cannot get your ticket-granting ticket using kinit:

- Ensure that the default realm is correct by looking at `krb.conf`.

- Ensure that the TrustBroker Master Server is running on the host specified for the realm.

- Ensure that the Master Server has an entry for the user principal and that the passwords match.

- Ensure that the `krb.conf` and `krb.realms` files are readable by Oracle.

If you have an initial ticket, but still cannot connect:

- After trying to connect, check for a service ticket.

- Check that the `sqlnet.ora` file on the database server side has a service name that corresponds to a service known to the CyberSafe Master Server.

- Check that the clocks on all the involved systems are within a few minutes of each other.

If you have a service ticket, and you still cannot connect:

- Check the clocks on the client and database server.

- Check that the `v5srvtab` file exists in the correct location and is readable by Oracle.

- Check that the `v5srvtab` file has been generated for the service named in the profile (`sqlnet.ora`) on the database server side.

If everything seems to work fine, but then you issue another query and it fails:

- Check that the initial ticket is forwardable. You must have obtained the initial ticket by running `kinit -f`.

- Check the expiration date on the credentials.

- If the credentials have expired, close the connection and run `kinit` to get a new initial ticket.

6

Configuring Kerberos Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle9*i*, or for the Oracle9*i* server, for use with Kerberos authentication—and how to configure Kerberos to authenticate Oracle database users. This chapter contains the following topics:

- Enabling Kerberos Authentication
- Utilities for the Kerberos Authentication Adapter
- Troubleshooting

Enabling Kerberos Authentication

To enable Kerberos authentication:

- Task 1: Install Kerberos
- Task 2: Configure a Service Principal for an Oracle Database Server
- Task 3: Extract a Service Table from Kerberos
- Task 4: Install an Oracle Database Server and an Oracle Client
- Task 5: Install Oracle Net and Oracle Advanced Security
- Task 6: Configure Oracle Net and Oracle9i
- Task 7: Configure Kerberos Authentication
- Task 8: Create a Kerberos User
- Task 9: Create an Externally-authenticated Oracle User
- Task 10: Get an Initial Ticket for the Kerberos/Oracle User

Task 1: Install Kerberos

Install Kerberos on the system that functions as the authentication server

> **See Also:** Related Documentation on page -xxvi, for information about how to install Kerberos

Task 2: Configure a Service Principal for an Oracle Database Server

To enable the Oracle database server to validate the identity of clients that authenticate themselves using Kerberos, you must create a **service principal** for Oracle9*i*.

The name of the principal should have the following format:

```
kservice/kinstance@REALM
```

kservice	A case-sensitive string that represents the Oracle service; this can be the same as the database service name.
kinstance	This is typically the fully-qualified name of the system on which Oracle9*i* is running.
REALM	The domain name of the database server. REALM must always be uppercase, and is typically the DNS domain name.

Note: The utility names in this section are executable programs. However, the Kerberos user name `krbuser` and the realm `SOMECO.COM` are examples only.

For example, if `kservice` is *oracle*, the fully-qualified name of the system on which Oracle9*i* is running is `dbserver.someco.com`, and the realm is `SOMECO.COM`; the principal name is:

```
oracle/dbserver.someco.com@SOMECO.COM
```

It is a convention to use the DNS domain name as the name of the realm. To create the **service principal**, run `kadmin.local`. The following example is UNIX-specific (*enter as root user*):

```
# cd /kerberos-install-directory/sbin
# ./kadmin.local
```

To add a **principal** named `oracle/dbserver.someco.com@SOMECO.COM` to the list of server principals known by Kerberos, enter the following:

```
kadmin.local:addprinc -randkey
oracle/dbserver.someco.com@SOMECO.COM
```

Task 3: Extract a Service Table from Kerberos

Extract the **service table** from Kerberos and copy it to the Oracle database server/Kerberos client system.

For example, to extract a service table for *dbserver.someco.com*:

1. Enter the following:

    ```
    kadmin.local:  ktadd -k /tmp/keytab
    oracle/dbserver.someco.com

    Entry for principal oracle/dbserver.someco.com with kvno 2,
    encryption DES-CBC-CRC added to the keytab WRFILE:
    'WRFILE:/tmp/keytab

    kadmin.local:  exit

    oklist -k -t /tmp/keytab
    ```

2. After the service table has been extracted, verify that the new entries are in the table in addition to the old ones. If they are not, or you need to add more, use `kadmin.local` to append the them.

    If you do not enter a realm when using `ktadd`, it uses the realm of the current host and displays it in the command output, as shown above.

3. If the Kerberos service table is on the same system as the Kerberos client, you can move it. If the service table is on a different system from the Kerberos client, you must transfer the file with a program such as FTP. If using FTP, transfer the file in binary mode.

    The following example is UNIX-specific.

    ```
    # mv /tmp/keytab /etc/v5srvtab
    ```

    The default name of the service file is `/etc/v5srvtab`.

4. Verify that the owner of the Oracle database server executable can read the service table (`/etc/v5srvtab` in the previous example). To do so, set the file owner to the Oracle user, or make the file readable by the group to which Oracle belongs.

    > **Caution:**   Do not make the file readable to all users; this can enable a security breach.

Task 4: Install an Oracle Database Server and an Oracle Client

Install the Oracle database server and client software.

> **See Also:** Oracle9*i* operating system-specific documentation

Task 5: Install Oracle Net and Oracle Advanced Security

Install Oracle Net and Oracle Advanced Security on the Oracle database server and Oracle client systems.

> **See Also:** Oracle9*i* operating system-specific installation documentation

Task 6: Configure Oracle Net and Oracle9*i*

Configure Oracle Net on the Oracle database server and client.

> **See Also:**
>
> - Oracle9*i* operating system specific installation documentation
> - *Oracle Net Services Administrator's Guide.*

Task 7: Configure Kerberos Authentication

Perform these tasks to set certain parameters in the Oracle database server and client `sqlnet.ora` files:

- Step 1: Configure Kerberos on the Client and on the Database Server
- Step 2: Set the Initialization Parameters
- Step 3: Set sqlnet.ora Parameters (optional)

Step 1: Configure Kerberos on the Client and on the Database Server

Perform the following steps to configure Kerberos authentication service parameters on the client and on the database server:

1. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears (Figure 6–1):

Figure 6–1   Oracle Advanced Security Authentication Window (Kerberos)
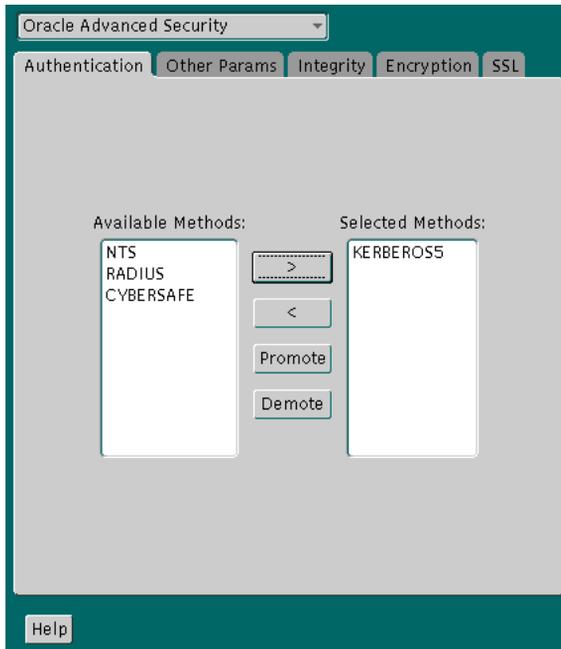
4. Choose the Authentication tab.

5. From the Available Methods list, select KERBEROS5.

6. Move KERBEROS5 to the Selected Methods list by clicking the right-arrow [>].

7. Arrange the selected methods in order of use. To do this, select a method in the Selected Methods list, then click Promote or Demote to position it in the list. For example, if you want KERBEROS5 to be the first service used, move it to the top of the list.

8. Choose the Other Params tab (Figure 6–2):

Figure 6–2   Oracle Advanced Security Other Params Window (Kerberos)
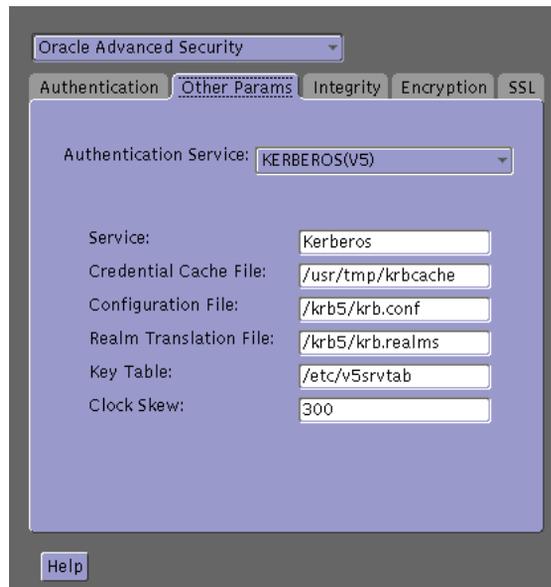
9. From the Authentication Service list, select KERBEROS(V5).

10. The Service field defines the name of the service Oracle9*i* uses to obtain a Kerberos **service ticket**; enter Kerberos. When you provide the value for this field, the other fields are enabled.

11. Optionally enter values for the following fields:

    ■   Credential Cache File

    ■   Configuration File

    ■   Realm Translation File

    ■   Key Table

    ■   Clock Skew

        See Also: Oracle Net Manager online help, and Step 3: Set sqlnet.ora Parameters (optional) on page 6-9, for more information about the fields and the parameters they configure

12. Choose File > Save Network Configuration.

The `sqlnet.ora` file is updated with the following entries:

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)

SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=kservice
```

Step 2: Set the Initialization Parameters

To set parameters in the initialization parameter file:

1. Add the following parameter to the initialization parameter file:

```
REMOTE_OS_AUTHENT=FALSE
```

> **Attention:** Setting REMOTE_OS_AUTHENT to TRUE can enable a security breach, because it lets someone using a non-secure protocol, such as TCP, perform an operating system-authorized login (formerly called an OPS$ login).

2. Because Kerberos user names can be long, and Oracle user names are limited to 30 characters, Oracle Corporation strongly recommends that you set the value of OS_AUTHENT_PREFIX to *null* as follows:

```
OS_AUTHENT_PREFIX=""
```

Setting this parameter to *null* overrides the default value of OPS$.

Step 3: Set sqlnet.ora Parameters (optional)

In addition to the required parameters, you can optionally set the following `sqlnet.ora` parameters on the client and the Oracle database server:

Parameter:	SQLNET.KERBEROS5_CC_NAME=*pathname_to_ credentials_cache_file*
Description:	Specifies the complete pathname to the Kerberos credentials cache (CC) file. The default value is operating system-dependent. For UNIX, it is `/tmp/krb5cc_user id`. You can also set this parameter by using the KRB5CCNAME environment variable, but the value set in the `sqlnet.ora` file takes precedence over the value set in KRB5CCNAME.
Example:	SQLNET.KERBEROS5_CC_NAME=/usr/tmp/krbcache
Parameter:	SQLNET.KERBEROS5_CLOCKSKEW=*number_of_ seconds_accepted_as_network_delay*
Description:	This parameter specifies how many seconds can pass before a Kerberos credential is considered out-of-date. It is used when a credential is actually received by either a client or a database server. An Oracle database server also uses it to decide if a credential needs to be stored to protect against a replay attack. The default is 300 seconds.
Example:	SQLNET.KERBEROS5_CLOCKSKEW=1200
Parameter:	SQLNET.KERBEROS5_CONF=*pathname_to_Kerberos_ configuration_file*
Description:	This parameter specifies the complete pathname to the Kerberos configuration file. The configuration file contains the realm for the default KDC (key distribution center) and maps realms to KDC hosts. The default is operating system-dependent. For UNIX, it is `/krb5/krb.conf`.
Example:	SQLNET.KERBEROS5_CONF=/krb/krb.conf
Parameter:	SQLNET.KERBEROS5_CONF_MIT=[TRUE|FALSE]

Description:	This parameter specifies whether the new MIT Kerberos configuration format will be used. If the value is set to TRUE, it will parse the file according to the new configuration format rules. When the value is set to False, the default (non-MIT) configuration is used. The default is False.
Example:	`SQLNET.KERBEROS5_CONF_MIT=False`
Parameter:	SQLNET.KERBEROS5_KEYTAB= *pathname_to_Kerberos_principal/key_table*
Description:	This parameter specifies the complete pathname to the Kerberos principal/secret key mapping file. It is used by the Oracle database server to extract its key and decrypt the incoming authentication information from the client. The default is operating system-dependent. For UNIX, it is `/etc/v5srvtab`.
Example:	`SQLNET.KERBEROS5_KEYTAB=/etc/v5srvtab`
Parameter:	SQLNET.KERBEROS5_REALMS= *pathname_to_Kerberos_realm_translation_file*
Description:	This parameter specifies the complete pathname to the Kerberos realm translation file. The translation file provides a mapping from a host name or domain name to a realm. The default is operating system-dependent. For UNIX, it is `/etc/krb.realms`.
Example:	`SQLNET.KERBEROS5_REALMS=/krb5/krb.realms`

Task 8: Create a Kerberos User

To create Oracle users that Kerberos can authenticate, perform this task on the Kerberos authentication server where the administration tools are installed. The realm must already exist.

> **Note:** The utility names in this section are executable programs. However, the Kerberos user name `krbuser` and realm `SOMECO.COM` are examples only; they can vary among systems.

Run `/krb5/admin/kadmin.local` as root to create a new Kerberos user, such as krbuser.

The following example is UNIX specific:

```
# ./kadmin.local
kadmin.local: addprinc krbuser
Enter password for principal: "krbuser@SOMECO.COM": (password does not display)
Re-enter password for principal: "krbuser@SOMECO.COM": (password does not
display)
kadmin.local: exit
```

Task 9: Create an Externally-authenticated Oracle User

Run SQL*Plus on the Oracle database server to create the Oracle user that corresponds to the Kerberos user. In the following example, OS_AUTHENT_ PREFIX is set to *null* (""). The Oracle user name is in uppercase enclosed in double quotation marks.

```
SQL> CONNECT / AS SYSDBA;
SQL> CREATE USER "KRBUSER@SOMECO.COM" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "KRBUSER@SOMECO.COM";
```

Task 10: Get an Initial Ticket for the Kerberos/Oracle User

Before you can connect to the database, you must ask the Key Distribution Center (KDC) for an **initial ticket**. To do so, run the following on the client:

```
% okinit user_name
```

If, when making a database connection, a reference such as the following follows a database link, you must use the forwardable flag (-f) option:

```
sqlplus /@oracle
```

Executing `okinit -f` enables credentials that can be used across database links. Run the following commands on the Oracle client:

```
% okinit -f
```

```
Password for krbuser@SOMECO.COM:password
```

Utilities for the Kerberos Authentication Adapter

Three utilities are shipped with the Oracle Kerberos authentication adapter. These utilities are intended for use on an Oracle client with Oracle Kerberos authentication support installed.

- Use `okinit` to obtain an initial ticket.

- Use `oklist` to display credentials

- Use `okdstry` to remove credentials from the credentials cache.

> **Note:** Solaris is shipped with Kerberos *version 4.* Ensure that the Kerberos *version 5* utilities are in the path so that the version 4 utilities are not used inadvertently.

Use okinit to Obtain the Initial Ticket

The `okinit` utility obtains and caches Kerberos tickets. This utility is typically used to obtain the ticket-granting ticket, using a password entered by the user to decrypt the credential from the key distribution center (KDC). The ticket-granting ticket is then stored in the user's credential cache.

The options available with okinit are listed in Table 6–1:

Table 6–1   Options for the okinit Utility

Option	Description
-f	Ask for a forwardable ticket-granting ticket. This option is necessary to follow database links.
-l	Specify the lifetime of the ticket-granting ticket and all subsequent tickets. By default, the ticket-granting ticket is good for eight (8) hours, but shorter or longer-lived credentials may be desired. Note that the KDC can ignore this option or put site-configured limits on what can be specified. The lifetime value is a string that consists of a number qualified by w (weeks), d (days), h (hours), m (months), or s (seconds), as in the following example: `okinit -l 2w1d6h20m30s` The example requests a ticket-granting ticket that has a life time of 2 weeks, 1 day, 6 hours, 20 minutes, and 30 seconds.
-c	Specify an alternative credential cache. For UNIX, the default is `/tmp/krb5cc_uid`. You can also specify the alternate credential cache by using the SQLNET.KERBEROS5_CC_NAME parameter in the `sqlnet.ora` file.
-?	List command line options.

Use OKLIST to Display Credentials

Run the oklist utility to display the list of tickets held; available oklist options are listed in Table 6–2:

Table 6–2   Options for the oklist Utility

Option	Description
-f	Show flags with credentials. Relevant flags are I, credential is a ticket-granting ticket, F, credential is forwardable, and f, credential is forwarded.
-c	Specify an alternative credential cache. In UNIX, the default is `/tmp/krb5cc_uid`. The alternate credential cache can also be specified by using the SQLNET.KERBEROS5_CC_NAME parameter in the `sqlnet.ora` file.
-k	List the entries in the service table (default `/etc/v5srvtab`) on UNIX. The alternate service table can also be specified by using the SQLNET.KERBEROS5_KEYTAB parameter in the `sqlnet.ora` file.

The *show flag* option (-f) displays additional information, as shown in the following example:

```
% oklist -f
27-Jul-1999 21:57:51    28-Jul-1999 05:58:14
krbtgt/SOMECO.COM@SOMECO.COM
Flags: FI
```

Use OKDSTRY to Remove Credentials from the Cache File

Use the okdstry utility to remove credentials from the credentials cache file:

```
$ okdstry -f
```

where the -f command option lets you specify an alternative credential cache. For UNIX, the default is /tmp/krb5cc_uid. You can also specify the alternate credential cache by using the SQLNET.KRB5_CC_NAME parameter in the sqlnet.ora file.

Connecting to an Oracle Database Server Authenticated by Kerberos

You can now connect to an Oracle database server without using a user name or password. Enter a command similar to the following:

```
$ sqlplus /@net_service_name
```

where net_service_name is an Oracle Net service name. For example:

```
$ sqlplus /@oracle_dbname
```

> **See Also:** Chapter 1, Introduction to Oracle Advanced Security, for information about external authentication and *Oracle9i Heterogeneous Connectivity Administrator's Guide*

Troubleshooting

This section lists some common configuration problems and explains how to resolve them.

If you cannot get your ticket-granting ticket using OKINIT:

- Ensure that the default realm is correct by examining the `krb.conf` file.

- Ensure that the KDC is running on the host specified for the realm.

- Ensure that the KDC has an entry for the user principal and that the passwords match.

- Ensure that the `krb.conf` and `krb.realms` files are readable by Oracle.

If you have an initial ticket, but still cannot connect:

- After trying to connect, check for a service ticket.

- Check that the `sqlnet.ora` file on the database server side has a service name that corresponds to a service known by Kerberos.

- Check that the clocks on all systems involved are within a few minutes of each other (or change the SQLNET.KERBEROS5_CLOCKSKEW parameter in the `sqlnet.ora` file).

If you have a service ticket and you still cannot connect:

- Check the clocks on the client and database server.

- Check that the `v5srvtab` file exists in the correct location and is readable by Oracle (remember to see the `sqlnet.ora` parameters).

- Check that the `v5srvtab` file has been generated for the service named in the `sqlnet.ora` file on the database server side.

If everything seems to work fine, but then you issue another query and it fails:

- Check that the initial ticket is forwardable. (You must have obtained the initial ticket by running the `okinit` utility.)

- Check the expiration date on the credentials.

- If the credentials have expired, close the connection and run `okinit` to get a new initial ticket.

7

Configuring Secure Sockets Layer Authentication

This chapter describes how to use the Secure Sockets Layer (SSL) protocol in Oracle Advanced Security. It contains the following topics:

- SSL in an Oracle Environment
- SSL Beyond an Oracle Environment
- SSL Combined with Other Authentication Methods
- SSL and Firewalls
- SSL Usage Issues
- Enabling SSL

SSL in an Oracle Environment

Secure Sockets Layer (SSL) is an industry standard protocol designed by Netscape Communications Corporation for securing network connections. SSL uses RSA public key cryptography to provide authentication, encryption, and data integrity in a **public-key infrastructure** (PKI).

This section discusses the following topics:

- What You Can Do with SSL

- Architecture of SSL in an Oracle Environment

- Components of SSL in an Oracle Environment

- How SSL Works in an Oracle Environment: The SSL Handshake

What You Can Do with SSL

By supporting SSL, Oracle Advanced Security expands its support encryption and data integrity, and provides public key authentication based on the SSL standard.

You can use Oracle Advanced Security SSL functionality to secure communications between clients and servers. You can authenticate:

- Any client or server to one or more Oracle database servers

- An Oracle database server to any client

You can use SSL features by themselves or in combination with other authentication methods supported by Oracle Advanced Security. For example, you can use the encryption provided by SSL in combination with the authentication provided by Kerberos. SSL supports any of the following authentication modes:

- Only the server authenticates itself to the client

- Both client and server authenticate themselves to each other

- Neither the client nor the server authenticates itself to the other, thus using the SSL encryption feature by itself

See Also:

- *The SSL Protocol*, Version 3.0, published by the Internet engineering Task Force, for a more detailed discussion of SSL

- Chapter 1, Introduction to Oracle Advanced Security, for more information about authentication methods

Architecture of SSL in an Oracle Environment

In an Oracle environment, SSL operates at the Oracle Protocols layer using TCP/IP, as illustrated by Figure 7–1:

Figure 7–1 SSL Architecture in an Oracle Environment
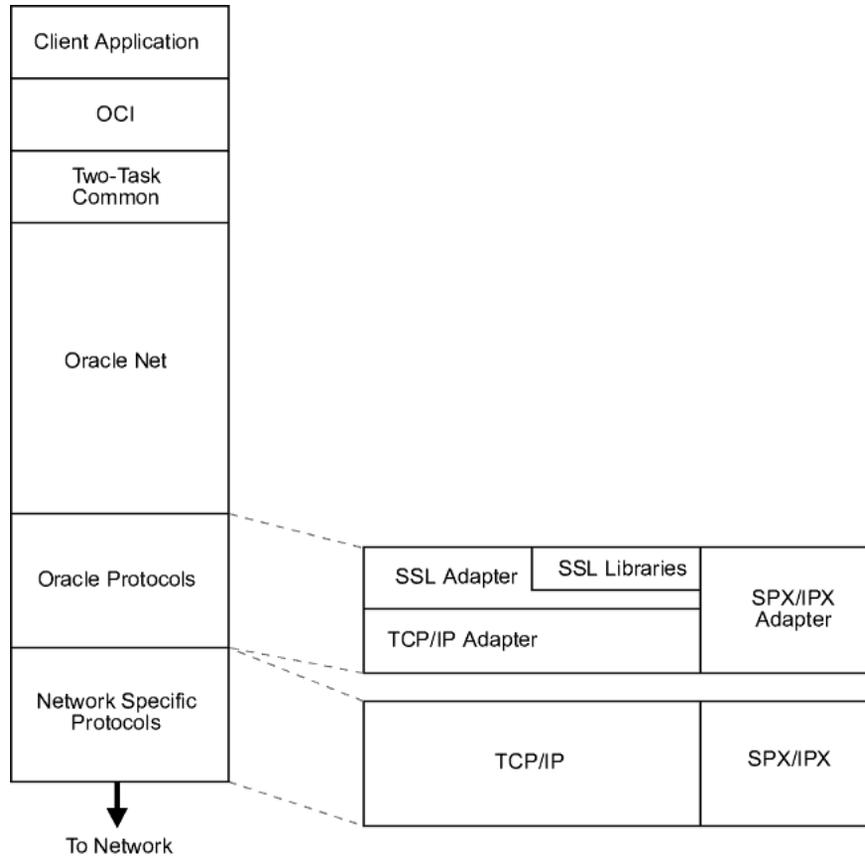
Components of SSL in an Oracle Environment

The components of SSL in an Oracle environment include the following:

- Certificate Authority
- Certificate
- Wallet

Certificate Authority

A certificate authority (CA) is a trusted third party that certifies the identity of third parties and other entities, such as users, databases, administrators, clients, and servers. The certificate authority verifies the party identity and grants a certificate, signing it with the its private key.

Different CAs may have different identification requirements when issuing certificates. One may require the presentation of a user's driver's license, while others may require notarization of the certificate request form, or fingerprints of the requesting party.

The CA publishes its own certificate, which includes its public key. Each network entity has a list of certificates of the CAs it trusts. Before communicating with another entity, a given entity uses this list to verify that the signature on the other entity's certificate is from a known, trusted CA.

Network entities can obtain their certificates from the same or different CAs. By default, Oracle Advanced Security automatically installs trusted certificates from VeriSign, RSA, Entrust, and GTE CyberTrust when you install a new wallet (See: Wallet on page 7-5).

Certificate

A certificate is created when a party's public key is signed by a trusted certificate authority (CA). A certificate ensures that a party's identification information is correct, and that the public key actually belongs to that party.

A certificate contains the party's name, public key, and an expiration date—as well as a serial number and certificate chain information. It can also contain information about the privileges associated with the certificate.

When a network entity receives a certificate, it verifies that it is a **trusted certificate**—one issued and signed by a **trusted certificate authority**. A certificate remains valid until it expires or is sooner terminated.

Wallet

A wallet is a transparent database used to manage authentication data such as keys, certificates, and trusted certificates needed by SSL. In an Oracle environment, each system using SSL has a wallet with an X.509 version 3 certificate, private key, and list of trusted certificates.

Security administrators use the Oracle Wallet Manager to manage security credentials on the server. Wallet owners use it to manage security credentials on clients. Specifically, the Oracle Wallet Manager is used to do the following:

- Generate a public-private key pair and create a certificate request for submission to a certificate authority.
- Install a certificate for the identity.
- Configure trusted certificates for the identity.

> **Note:** Installation of Oracle Advanced Security Release 9.0.1 also installs Oracle Wallet Manager release 2.0 and Oracle Enterprise Login Assistant release 1.0.

See Also:

- Chapter 16, Using Oracle Wallet Manager
- Creating a New Wallet on page 16-12.
- Managing Trusted Certificates on page 16-24.

How SSL Works in an Oracle Environment: The SSL Handshake

At the commencement of a network connection under SSL, the client and server perform a SSL handshake that includes the following principal tasks:

- The client and server establish which **cipher suites** to use.

- The server sends its certificate to the client, and the client verifies that the server's certificate was signed by a trusted CA.

- Similarly, if client authentication is required, the client sends its own certificate to the server, and the server verifies that the client's certificate was signed by a trusted CA.

- The client and server exchange key information using public key cryptography; based on this information, each generates a **session key**. All subsequent communications between the client and the server is encrypted and decrypted by using this set of session keys and the negotiated cipher suite.

In an Oracle environment, the authentication process consists of the following basic steps:

1. The user initiates a Oracle Net connection to the server by using SSL.

2. SSL performs the handshake between the client and the server.

3. If the handshake is successful, the server verifies that the user has the appropriate **authorization** to access the database.

SSL Beyond an Oracle Environment

You can use the Oracle Advanced Security SSL feature to secure connections between non-Oracle clients and Oracle database servers. For example, SSL can grant secure access to a client outside an Oracle network to authorized data within the Oracle network.

Figure 7–2 shows how SSL is used to secure connections between Oracle and non-Oracle entities over the Internet. In this example, a Web server runs as an Oracle9*i* Java client. It receives messages over **HTTPS** (HTTP secured by SSL), and sends **CORBA** requests to the Oracle database server over **IIOP**/SSL (IIOP secured by SSL). In this example, the Web server *passes its own certificate to the Oracle server*, rather than the certificate of the Web client.

Figure 7–2   Connecting to an Oracle Server over the Internet
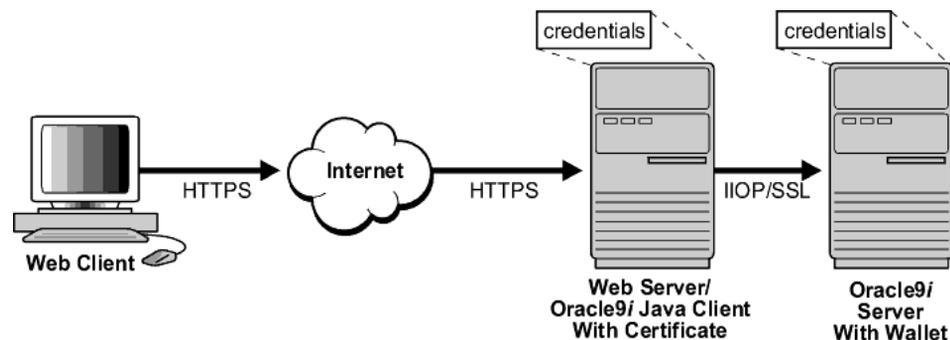
> **See Also:**   *Oracle9i Enterprise JavaBeans Developer's Guide and Reference*, for information about using and configuring IIOP/SSL

SSL Combined with Other Authentication Methods

Because of its implementation architecture, you can configure Oracle Advanced Security to use SSL concurrently with other supported authentication methods, such as Kerberos, RADIUS, or CyberSafe—as discussed in the next sections:

- Architecture: Oracle Advanced Security and SSL
- Using SSL with Other Authentication Methods

Architecture: Oracle Advanced Security and SSL

Figure 7–3 displays the Oracle Advanced Security implementation architecture, which shows that (i) Oracle Advanced Security operates at the **session layer** on top of SSL, which (ii) uses TCP/IP at the **transport layer**. This separation of functionality lets you employ SSL concurrently with other supported protocols.

Figure 7–3   SSL in Relation to Oracle Advanced Security
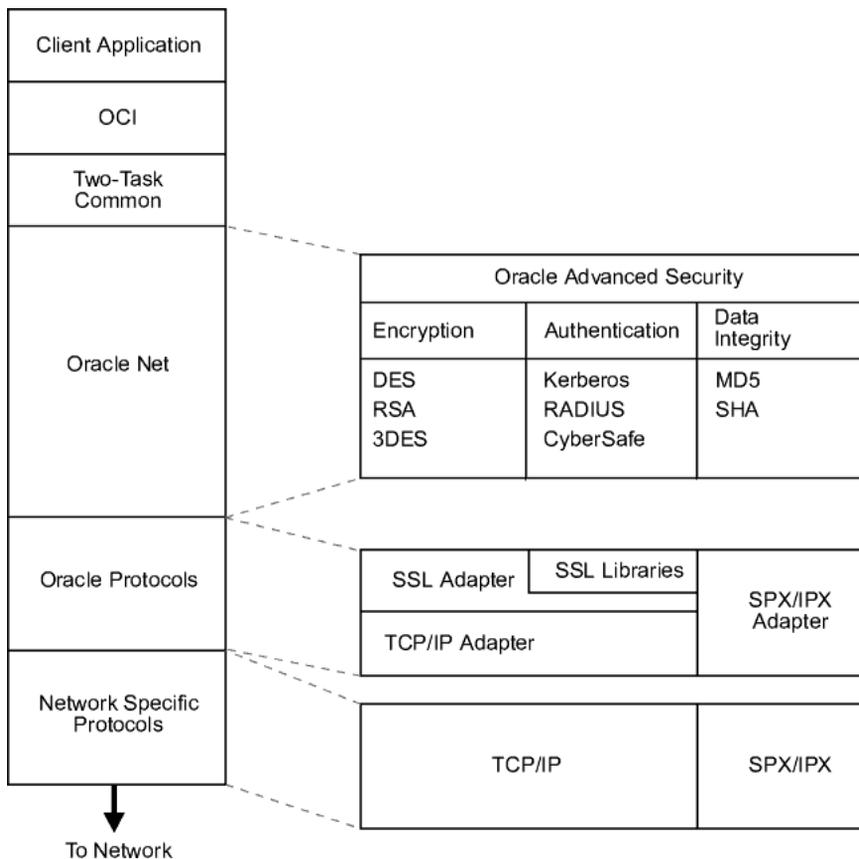
See Also: *Oracle Net Services Administrator's Guide,* for information about stack communications in an Oracle networking environment

Using SSL with Other Authentication Methods

Figure 7–4 illustrates a configuration in which SSL is used in combination with another authentication method supported by Oracle Advanced Security. In this example, SSL is used to establish the initial handshake (server authentication), and an alternative authentication method is used to authenticate the client.

Figure 7–4   SSL in Relation to Other Authentication Methods
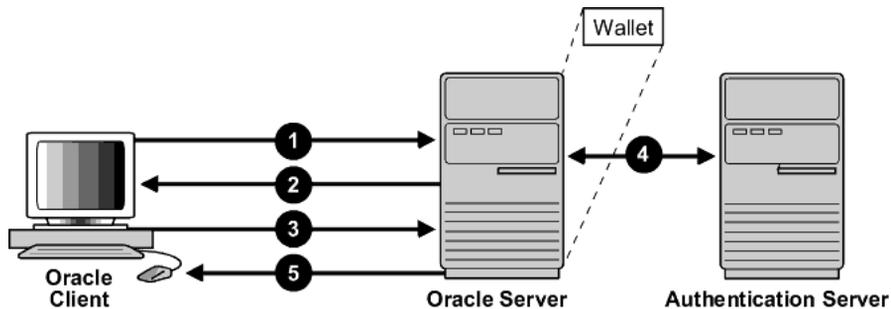
1. The client seeks to connect to the Oracle database server.

2. SSL performs a handshake during which the server authenticates itself to the client and both the client and server establish which cipher suite to use. See How SSL Works in an Oracle Environment: The SSL Handshake on page 7-6.

3. Once the SSL handshake is successfully completed, the user seeks access to the database.

4. The Oracle database server exchanges the user's authentication information with the authentication server—using a non-SSL authentication method (e.g., Kerberos, CyberSafe, RADIUS).

5. Upon validation by the authentication server, the Oracle database server grants access and authorization to the user.

6. The user accesses the Oracle database securely using SSL.

SSL and Firewalls

Oracle Advanced Security supports two types of firewalls:

- Application proxy-based firewalls, such as Network Associates Gauntlet, or Axent Raptor.

- Stateful packet inspection firewalls, such as Check Point Firewall-1, or Cisco PIX Firewall.

When you enable SSL, the stateful inspection firewalls behave like application proxy firewalls because they do not decrypt encrypted packets.

Firewalls do not inspect encrypted traffic. When a firewall encounters data addressed to an SSL port on an intranet server, it simply checks the target IP address against its access rules—letting the SSL packet pass through to permitted SSL ports, and rejecting all others.

With the availability of the Oracle Net Firewall Proxy kit, firewall applications can now provide specific support for database network traffic. If the proxy kit is implemented in the firewall, the following processing take place:

- The Net Proxy (a component of the Oracle Net Firewall Proxy kit) must know where to route its traffic.

- The database listener requires access to a **certificate** in order to participate in the SSL handshake. The listener inspects the SSL packet and identifies the target database, returning the port on which the target database listens to the client. This port must be designated as an SSL port.

- The client communicates on this server-designated port in all subsequent connections.

- The number of ports that are open in the firewall increase as a function of the number of database connections requested. This approach prohibits the database server from using randomly chosen SSL ports, because the SSL ports on the firewall must match those chosen by the database. You can avoid this condition by deploying Oracle Connection Manager, an application included with Oracle Advanced Security Enterprise Edition.

Oracle Connection Manager lets you route client connections over multiple Net Manager protocols. Each client connection request establishes an SSL connection between the client and Oracle Connection Manager, which in turn establishes a TCP/IP connection with the target database. Multiple clients can thus connect to multiple databases behind the firewall, using a single SSL port through the firewall.

> **Note:**   Although Oracle Connection Manager can be used to avoid opening up multiple SSL ports through the firewall, consider the following:
>
> - The internal connection, between Oracle Connection Manager and the database, *is not an SSL connection.* You should encrypt such connections, using Oracle Advanced Security native encryption.
>
> - Because such connections do not use SSL, clients cannot use certificate-based authentication.

SSL Usage Issues

Consider the following issues when using SSL:

- SSL use enables authorization retrieval from an LDAP-based directory service. Client-side SSL authentication is required in order to manage enterprise users and their privileges in a directory.

- Because SSL supports both authentication and encryption, the client database server connection is *somewhat slower* than the standard Oracle Net TCP/IP transport (using native encryption).

- Oracle Advanced Security SSL requires Oracle9*i*; it does not work with earlier database releases.

- Each SSL authentication mode requires unique configuration settings. See: Enabling SSL on page 7-14.

    > **Note:**
    >
    > - *U.S. government regulations prohibit double encryption. Accordingly, if you configure Oracle Advanced Security to use SSL encryption and another encryption method concurrently, the connection fails (you also cannot configure SSL authentication concurrently with non-SSL authentication).*
    >
    > - If you configure SSL encryption, you must disable non-SSL encryption. To disable such encryption, see: Disabling Oracle Advanced Security Authentication on page 9-3.

Enabling SSL

To enable SSL:

- Task 1: Install Oracle Advanced Security and Related Products

- Task 2: Configure SSL on the Client

- Task 3: Configure SSL on the Server

- Task 4: Log on to the Database

Task 1: Install Oracle Advanced Security and Related Products

Install Oracle Advanced Security on both the client and server. When you do this, the Oracle Universal Installer automatically installs SSL, Oracle Wallet Manager, and Oracle Enterprise Login Assistant on your system.

> **See Also:** The Oracle9*i* installation documentation for your platform.

Task 2: Configure SSL on the Client

To configure SSL on the client:

- Step 1: Confirm Wallet Creation

- Step 2: Configure Service Name

- Step 3: Specify Required Client Configuration (Wallet Location)

- Step 4: Set the SSL Cipher Suites on the Client (Optional)

- Step 5: Set the Required SSL Version (Optional)

- Step 6: Set SSL as an Authentication Service (Optional)

- Step 7: Create a Net Service Name that Uses TCP/IP with SSL in the Connect Descriptor

> **See Also:** Appendix B, Authentication Parameters, for the dynamic parameter names.

Step 1: Confirm Wallet Creation

Before proceeding with the next step, you must confirm that a wallet has been created.

See Also:

- Chapter 16, Using Oracle Wallet Manager, for general information about wallets

- Opening an Existing Wallet on page 16-13, for information about opening an existing wallet

- Creating a New Wallet on page 16-12, for information about creating a new wallet

Step 2: Configure Service Name

Oracle Advanced Security Release 9.0.1 matches the server's global database name against the **distinguished name (DN)** from the server certificate. This protects against the threat of connections to a server potentially faking its identity, where the server has a valid X.509 v3 certificate, but not the proper certificate for the respective database.

You can control the system's behavior when there is a mismatch between the **service name** and the DN, by defining the *Match server X.509* name in the Oracle Advanced Security SSL Window (Figure 7–5), as described by Step 3, Item 7 on page 7-18. This step defines the SSL_SERVER_DN_MATCH parameter, stored in the sqlnet.ora file.

However, before proceeding with Step 3, you must manually edit the tnsnames.ora file to configure the service name, by defining the TNS_SERVER_ DN parameter—to include the server DNs to which the client expects possible connections.

Example:

```
dbalias = (description = address_list = (address = (protocol = tcps)
(host = hostname) (port = portnum)))
(connect_date = (service_name = Finance))
(security=(SSL_SERVER_DN="CN=Finance,CN=OracleContext,C=US,O=Acme"))
```

The tnsnames.ora file can be located on the client or in the LDAP directory.

Alternatively, the administrator can ensure that DNs in the certificates from a trusted certificate authority have a common name (CN) that matches the service name.

Oracle recommends that you use Oracle Wallet Manager to remove the **trusted certificate** in your Oracle wallet associated with each **certificate authority** that you do not use.

See Also:

- Table B–22, SSL X.509 Server Match Parameters, for information about the server match parameters

- Chapter 16, Using Oracle Wallet Manager, for information about using Oracle Wallet Manager

Step 3: Specify Required Client Configuration (Wallet Location)

To specify required configuration parameters for the client:

1. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–5):

Figure 7–5   Oracle Advanced Security SSL Window (Client)
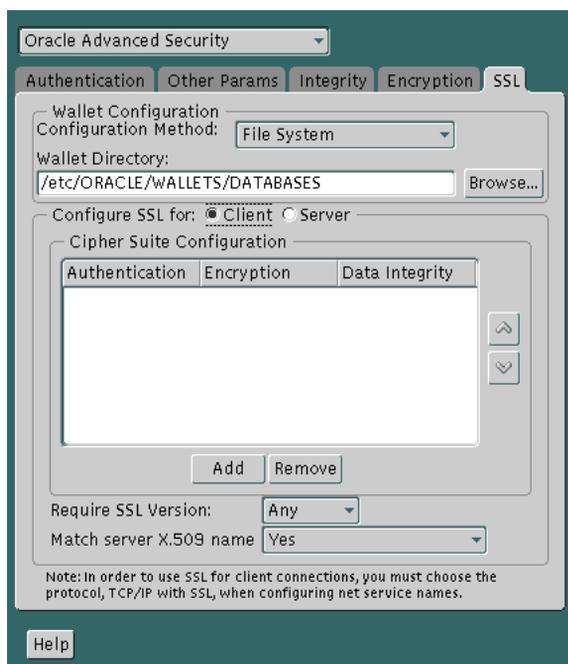
4. Choose the SSL tab.

5. Select Configure SSL for Client.

6. In the Wallet Directory box, enter the directory in which the Oracle wallet is located, or click Browse to find it by searching the file system.

> **Important:**   There are two occasions during the client and the server configuration when you set the location of the Oracle wallet. *Be sure to enter the same location on both occasions.*
>
> - On the occasion described in this section, you set the location of the wallet either by using the Oracle Net Manager or by modifying the sqlnet.ora file.
>
> - Later, you use the Oracle Wallet Manager. See: Step 1: Create a Database Wallet on page 15-44.

7. From the *Match server X.509 name* drop-down list, choose one of the following options:

- **Yes:** Requires that the server's **distinguished name (DN)** match its service name. SSL ensures that the certificate is from the server; connections succeed only if there is a match.

- **No (default):** SSL checks for a match between the DN and the service name, but does not enforce it. Connections succeed regardless of the outcome, but an error is logged if the match fails.

- **Let Client Decide:** Enables the default.

> **Note:** The following alert appears when you select *No:*
>
> *Security Alert*
>
> *Not enforcing the server X.509 name match allows a server to potentially fake its identity. Oracle Corporation recommends selecting YES for this option so that connections are refused when there is a mismatch.*

8. Choose File > Save Network Configuration.

   The sqlnet.ora file is updated with the following entries:

```
SSL_CLIENT_AUTHENTICATION =TRUE
wallet_location =
 (SOURCE=
  (METHOD=File)
  (METHOD_DATA=
   (DIRECTORY=wallet_location)))

SSL_SERVER_DN_MATCH=(ON/OFF)
```

Step 4: Set the SSL Cipher Suites on the Client (Optional)

A cipher suite is a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network entities. During an SSL handshake, two entities negotiate to see which cipher suite they will use when transmitting messages back and forth.

When you install Oracle Advanced Security, several SSL cipher suites are set for you by default. You can override the default by setting the SSL_CIPHER_SUITES parameter. For example, if you use Oracle Net Manager to add the cipher suite SSL_RSA_WITH_RC4_128_SHA, all other cipher suites in the default setting are ignored.

You can prioritize the cipher suites. When the client negotiates with servers regarding which cipher suite to use, it follows the prioritization you set. When you prioritize the cipher suites, consider the following:

- The level of security you want to use. For example, triple-DES encryption is stronger than DES.

- The impact on performance. For example, triple-DES encryption is slower than DES.

- Administrative requirements:

  The cipher suites selected for a client must be compatible with those required by the server. For example, in the case of an Oracle Call Interface (OCI) user, the server requires the client to authenticate itself. You cannot, in this case, use a cipher suite employing Diffie-Hellman anonymous authentication which disallows the exchange of certificates. By contrast, in the case of an Enterprise JavaBeans (EJB) user, the server does not require the client to authenticate itself. In this case, you can use Diffie-Hellman anonymous authentication.

You typically prioritize cipher suites starting with the strongest and moving to the weakest.

Table 7–1 lists the SSL cipher suites supported in the current release of Oracle Advanced Security. These cipher suites are set by default when you install Oracle Advanced Security. This table also lists the authentication, encryption, and data integrity types each cipher suite uses.

Table 7–1   Oracle Advanced Security Cipher Suites

Cipher Suite	Authentication	Encryption	Data Integrity
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES EDE CBC	SHA
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4 128	SHA
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4 128	MD5
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES CBC	SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH anon	3DES EDE CBC	SHA
SSL_DH_anon_WITH_RC4_128_MD5	DH anon	RC4 128	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH anon	DES CBC	SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4 40	MD5

Table 7–1   Oracle Advanced Security Cipher Suites

Cipher Suite	Authentication	Encryption	Data Integrity
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA	DES40 CBC	SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	DH anon	RC4 40	MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH anon	DES40 CBC	SHA

To specify cipher suites for the client:

1. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–5).

4. Choose the SSL tab.

5. Select Configure SSL for Client.

6. Choose the Add button; a dialog box displays available cipher suites (Figure 7–6):

Figure 7–6   SSL Cipher Suites Window

Authentication	Encryption	Data Integrity
RSA_EXPORT	RC4_40	MD5
RSA_EXPORT	DES40_CBC	SHA
RSA	RC4_128	MD5
RSA	NULL	SHA
RSA	NULL	MD5
RSA	3DES_EDE_CBC	SHA
RSA	DES_CBC	SHA
RSA	RC4_128	SHA
DH_anon	3DES_EDE_CBC	SHA
DH_anon	RC4_128	MD5
DH_anon	DES40_CBC	SHA
DH_anon	RC4_40	MD5

Select a Cipher Suite to enable

Ok   Cancel
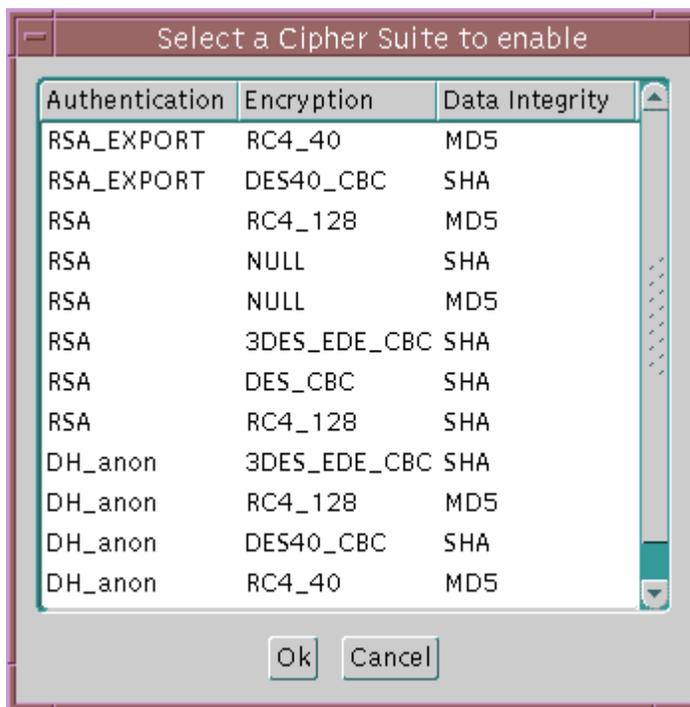
7. Select a suite and choose OK; the Cipher Suite Configuration list is updated
(Figure 7–7):

Figure 7–7   Oracle Advanced Security SSL Window (Client)
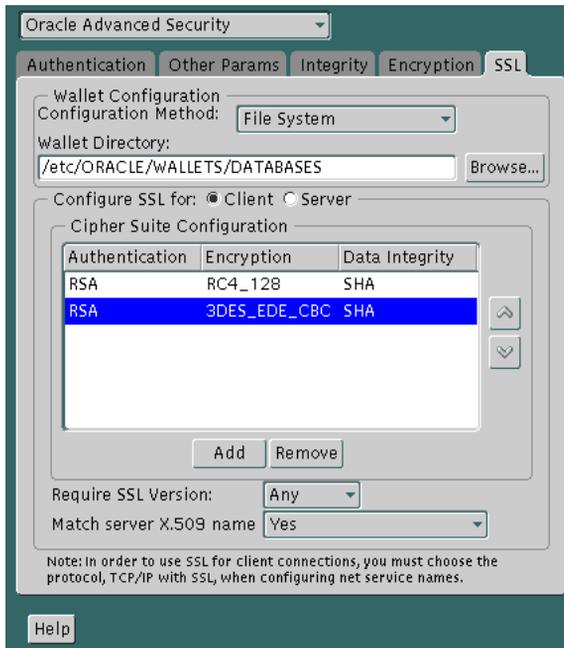
8. Use the up and down arrows to prioritize the cipher suites.

9. Choose File > Save Network Configuration.

    The sqlnet.ora file is updated with the following entry:

    SSL_CIPHER_SUITES= *(SSL_cipher_suite1 [,SSL_cipher_suite2])*

Step 5: Set the Required SSL Version (Optional)

You can set the SSL_VERSION parameter in the sqlnet.ora file. This parameter defines the version of SSL that must run on the systems with which the client communicates. You can require these systems to use SSL 3.0, or any valid, future version. The default setting for this parameter in sqlnet.ora is 0; in Oracle Net Manager, it is Any.

To set the SSL version for the client:

1. Start Oracle Net Manager:

    ■   On UNIX, run netmgr from $ORACLE_HOME/bin.

- On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–5).

4. Choose the SSL tab.

5. Select Configure SSL for Client.

6. In the Require SSL Version scroll box the default is Any; accept this default or select the SSL version you want to configure.

7. Choose File > Save Network Configuration.

   The sqlnet.ora file is updated with the following entry:

   ```
   SSL_VERSION=UNDETERMINED
   ```

Step 6: Set SSL as an Authentication Service (Optional)

The SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file sets the SSL authentication service.

Set this parameter only if both of the following conditions apply:

- You want to use SSL authentication in conjunction with another authentication method supported by Oracle Advanced Security. For example, you want the server to authenticate itself to the client by using SSL and the client to authenticate itself to the server by using Kerberos.

- You are not using Oracle Net Manager to configure the client or the server.

If both of the above conditions apply, add TCP with SSL (TCPS) to this parameter in the sqlnet.ora file by using a text editor. For example:

```
SQLNET.AUTHENTICATION_SERVICES = (BEQ, TCPS, radius)
```

If either or both of the above conditions do not apply, do not set this parameter.

Step 7: Create a Net Service Name that Uses TCP/IP with SSL in the Connect Descriptor

The client must be configured with the location of the listener. For an SSL connection, the client must be configured with a TCP/IP with SSL listener protocol address.

> **See Also:** *Oracle Net Services Administrator's Guide* to create a net service name

Task 3: Configure SSL on the Server

During installation, Oracle sets defaults on both the Oracle database server and on the Oracle client for all SSL parameters except the location of the Oracle wallet. To configure SSL on the server, perform these steps:

- Step 1: Confirm Wallet Creation
- Step 2: Specify Required Server Configuration (Wallet Location)
- Step 3: Set the SSL Cipher Suites on the Server (Optional)
- Step 4: Set the Required SSL Version (Optional)
- Step 5: Set SSL Client Authentication (Optional)
- Step 6: Set SSL as an Authentication Service (Optional)
- Step 7: Create Listening Endpoint that Uses TCP/IP with SSL

> **See Also:** Appendix B, for the dynamic parameter names

Step 1: Confirm Wallet Creation

Before proceeding with the next step, you must confirm that a wallet has been created.

> **See Also:**
> - Chapter 16, Using Oracle Wallet Manager, for general information about wallets
> - Opening an Existing Wallet on page 16-13, for information about opening an existing wallet
> - Creating a New Wallet on page 16-12, for information about creating a new wallet

Step 2: Specify Required Server Configuration (Wallet Location)

To specify required configuration parameters for the server:

1. Start Oracle Net Manager:

   - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

- On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–5).

4. Choose the SSL tab.

5. Select Configure SSL for Server.

6. In the Wallet Directory box, enter the directory in which the Oracle wallet is located, or click the Browse button to find it by searching the file system.

> **Important:**   There are two occasions during the client and the server configuration process when you set the location of the Oracle wallet. *Be sure to enter the same location on both occasions.*
>
> - On the occasion described in this section, you set the location of the wallet either by using Oracle Net Manager or by modifying the `sqlnet.ora` file.
>
> - Later, you use the Oracle Wallet Manager. See: Step 1: Create a Database Wallet on page 15-44.

7. Choose File > Save Network Configuration.

The `sqlnet.ora` and `listener.ora` files are updated with the following entries:

```
wallet_location =
 (SOURCE=
  (METHOD=File)
  (METHOD_DATA=
   (DIRECTORY=wallet_location)))
```

> **Note:**   The listener uses the wallet defined in `listener.ora` (it can use any database wallet). When SSL is configured for a server using Net Manager, the wallet location entered into `listener.ora` is the same as that entered into `sqlnet.ora`. The location of the listener wallet is not relevant to the Oracle client, because the client is only performing an SSL handshake with the listener.
>
> To change the listener wallet location (so that the listener has its own wallet), you can edit `listener.ora` to enter the new location.

Step 3: Set the SSL Cipher Suites on the Server (Optional)

A cipher suite is a set of authentication, encryption, and data integrity algorithms used for exchanging messages between network entities. During an SSL handshake, two entities negotiate to see which cipher suite they will use when transmitting messages back and forth.

When you install Oracle Advanced Security, several SSL cipher suites are set for you by default. You can override the default by setting the SSL_CIPHER_SUITES parameter. For example, if you use Oracle Net Manager to add the cipher suite SSL_RSA_WITH_RC4_128_SHA, all other cipher suites in the default setting are ignored.

You can prioritize the cipher suites. When the client negotiates with servers regarding which cipher suite to use, it follows the prioritization you set. When you prioritize the cipher suites, consider the following:

- The level of security you want to use. For example, triple-DES encryption is stronger than DES.

- The impact on performance. For example, triple-DES encryption is slower than DES.

- Administrative requirements:

  The cipher suites selected for a server must be compatible with those required by the client.

  You typically prioritize cipher suites starting with the strongest and moving to the weakest.

> **Note:**   In Oracle Advanced Security Release 9.0.1, if you set a cipher suite employing Diffie-Hellman anonymous authentication on the server, you must also set the same cipher suite on the client. Otherwise, the connection fails.
>
> If you use a cipher suite employing Diffie-Hellman *anonymous,* you must set the SSL_CLIENT_AUTHENTICATION parameter to FALSE. See: Step 5: Set SSL Client Authentication (Optional) on page 7-29.

Table 7–1 lists the SSL cipher suites supported in the current release of Oracle Advanced Security. These cipher suites are set by default when you install Oracle Advanced Security. This table also lists the authentication, encryption, and data integrity types each cipher suite uses.

To specify cipher suites for the server:

1. Start Oracle Net Manager:

    - On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

    - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–5).

4. Choose the SSL tab.

5. Select Configure SSL for Server.

6. Choose the Add button; a dialog box displays available cipher suites (Figure 7–6).

7. Select a suite and choose OK; the Cipher Suite Configuration list is updated (Figure 7–8):

Figure 7–8   Oracle Advanced Security SSL Window (Server)
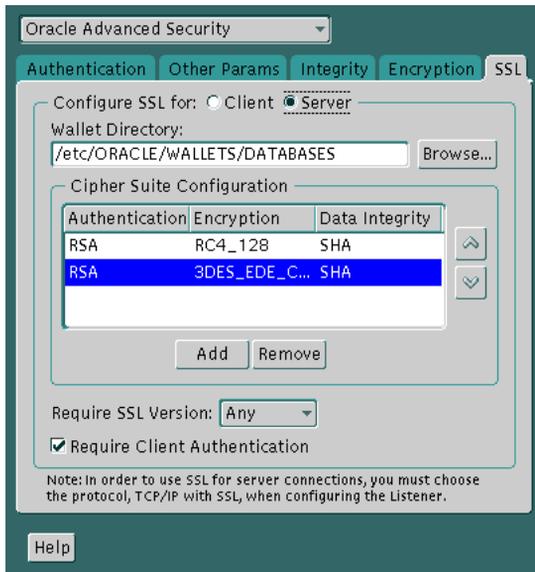
8. Use the up and down arrows to prioritize the cipher suites.

9. Choose File > Save Network Configuration.

   The sqlnet.ora file is updated with the following entry:

   ```
   SSL_CIPHER_SUITES= (SSL_cipher_suite1 [,SSL_cipher_suite2])
   ```

Step 4: Set the Required SSL Version (Optional)

You can set the SSL_VERSION parameter in the sqlnet.ora file. This parameter defines the version of SSL that must run on the systems with which the client communicates. You can require these systems to use SSL 3.0, or any valid, future version. The default setting for this parameter in sqlnet.ora is 0; in Oracle Net Manager, it is Any.

To set the SSL version for the server:

1. Start Oracle Net Manager:

   - On UNIX, run netmgr from $ORACLE_HOME/bin.

   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–5).

4. Choose the SSL tab.

5. Select Configure SSL for Server.

6. In the Require SSL Version scroll box the default is Any; accept this default or select the SSL version you want to configure.

7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file is updated with the following entry:

   ```
   SSL_VERSION=UNDETERMINED
   ```

Note: SSL 2.0 is not supported on the server side.

Step 5: Set SSL Client Authentication (Optional)

The SSL_CLIENT_AUTHENTICATION parameter in the `sqlnet.ora` file controls whether the client is authenticated using SSL. The default value is TRUE.

You must set this parameter to FALSE if you are using a cipher suite that contains Diffie-Hellman anonymous authentication (DH_anon). Also, you can set this parameter to FALSE for the client to authenticate itself to the server by using any of the non-SSL authentication methods supported by Oracle Advanced Security, such as Kerberos or CyberSafe.

To set this parameter to FALSE:

1. Start Oracle Net Manager:

   ■ On UNIX, run `netmgr` from `$ORACLE_HOME/bin`.

   ■ On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.

2. In the Navigator window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security SSL window appears (Figure 7–9):

Figure 7–9   Oracle Advanced Security SSL Window (Server)
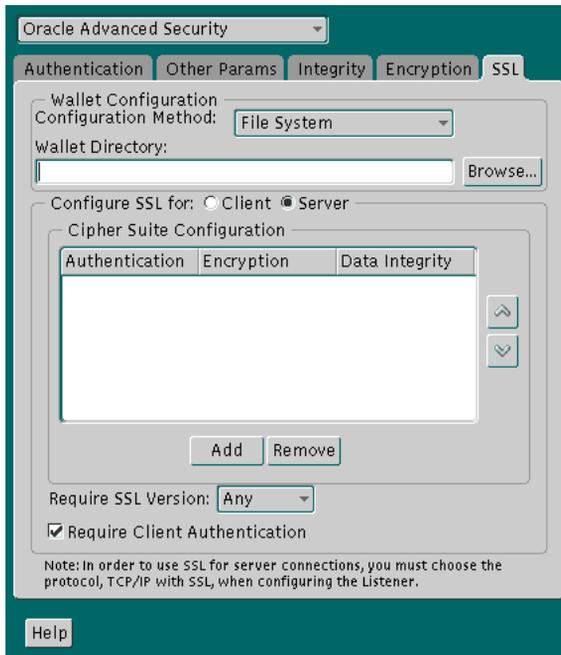
4. Choose the SSL tab.

5. Select Configure SSL for Server.

6. Deselect Require Client Authentication.

7. Choose File > Save Network Configuration.

    The sqlnet.ora file is updated with the following entry:

    ```
    SSL_CLIENT_AUTHENTICATION=FALSE
    ```

Step 6: Set SSL as an Authentication Service (Optional)

The SQLNET.AUTHENTICATION_SERVICES parameter in the sqlnet.ora file sets the SSL authentication service.

Set this parameter only if both of the following conditions apply:

■ You want to use SSL authentication in conjunction with another authentication method supported by Oracle Advanced Security. For example, you want the

server to authenticate itself to the client by using SSL and the client to authenticate itself to the server by using Kerberos.

- You are not using Oracle Net Manager to configure the client or the server.

If both of the above conditions apply, add TCP with SSL (TCPS) to this parameter in the `sqlnet.ora` file by using a text editor.

For example:

```
SQLNET.AUTHENTICATION_SERVICES = (BEQ, TCPS, radius)
```

If either or both of the above conditions do not apply, do not set this parameter.

Step 7: Create Listening Endpoint that Uses TCP/IP with SSL

Configure the listener with a TCP/IP with SSL listening endpoint in the `listener.ora` file. Oracle Corporation recommends a port number 2484 for typical Oracle Net clients and 2482 for client connections to Oracle9*i* JServer.

> **See Also:** *Oracle Net Services Administrator's Guide.*

Task 4: Log on to the Database

If you are using SSL authentication, launch SQL*Plus and enter the following:

```
CONNECT/@dnet_service_name
```

If you are not using SSL authentication, launch SQL*Plus and enter the following:

```
CONNECT username/password@net_service_name
```

8

Configuring Entrust-Enabled SSL Authentication

This chapter describes how to configure and use Entrust-enabled Oracle Advanced Security for Secure Socket Layer (SSL) authentication. It contains the following topics:

- Overview
- System Components
- Entrust Authentication Process
- Enabling Entrust Authentication
- Issues and Restrictions
- Troubleshooting Entrust In Oracle Advanced Security

Overview

A **public**-**key infrastructure** (PKI) includes various elements, such as a public key, bound into a digital certificate, a private key, and certain other security credentials. These credentials can be used for secure authentication over a **Secure Sockets Layer (SSL)** connection, to establish a secure communication channel, and to generate and process digital certificates—including digital signatures. A complete PKI includes the following:

- Certificate revocation status checking
- Easy management of user keys and certificates
- Easy deployment, hiding PKI complexity from users

This section describes the PKI implementation provided by the following:

- Oracle Advanced Security
- Entrust/PKI
- Entrust-Enabled Oracle Advanced Security

Oracle Advanced Security

Oracle Advanced Security includes elements of a PKI, such as Oracle Wallet Manager, which creates and securely stores a user's **public/private key pair**, as well as the **trust points** (the list of root certificates the user trusts). The user's PKI credentials, stored in Oracle Wallet Manager, can be used to create a secure, authenticated session over SSL. However, Oracle Advanced Security does not provide **certificate** creation or certificate revocation status checking, which are important elements of a complete PKI.

For example, although Oracle Wallet Manager can generate a PKCS#10 certificate signing request, users must obtain certificate fulfillment from a **certificate authority** and load the resulting certificate into an Oracle wallet. Oracle wallets only support authentication to Oracle applications.

Entrust/PKI

Entrust/PKI is a PKI product provided by Entrust Technologies, Inc. that provides certificate generation, certificate revocation, and key and certificate management.

Entrust-Enabled Oracle Advanced Security

The integration of Oracle Advanced Security with Entrust/PKI enables users of both Entrust and Oracle to utilize the extensive PKI capabilities of Entrust to enhance the security of their Oracle environment.

Entrust-enabled Oracle Advanced Security provides:

- Enhanced X.509-Based Authentication and Single Sign-On

- Integration with Entrust/PKI Key Management

- Integration with Entrust/PKI Certificate Revocation

> **Note:**
>
> - Oracle Advanced Security has been certified as *Entrust-Ready* by Entrust Technologies Inc., as at Release 8.1.7.
>
> - See Also: `http://www.entrust.com`

Enhanced X.509-Based Authentication and Single Sign-On

Entrust-enabled Oracle Advanced Security supports the use of Entrust credentials for **X.509**-based authentication and single sign-on. Instead of using an Oracle wallet to hold user PKI credentials, Oracle Advanced Security can access PKI credentials created by Entrust/Authority and held in an Entrust profile (an.epf file). Users who have deployed Entrust software within their enterprise are thus able to use it for authentication and single sign-on to Oracle9*i*.

Integration with Entrust/PKI Key Management

Entrust-enabled Oracle Advanced Security uses the extensive key management and rollover functionality provided by Entrust/PKI, which shield users from the complexity of a PKI deployment. For example, users are automatically notified when their certificates are expiring, and certificates are reissued according to administrator-configurable preferences.

Integration with Entrust/PKI Certificate Revocation

Entrust provides a certificate authority component, which natively checks certificate revocation status and enables the revocation of certificates.

Users using Entrust credentials for authentication to Oracle are assured that the revocation status of the certificate is checked, and connections are prevented if the certificate is revoked.

System Components

This section describes the system components required for using Entrust-enabled Oracle Advanced Security:

- Entrust/PKI 5.0.2 for Oracle

- Entrust/Toolkit Server Login 5.0.2

- Entrust IPSEC Negotiator Toolkit 5.0.2

> **Note:** In the following sections, the term **client** refers to a client connecting to an Oracle database, and the term **server** refers to the host on which the Oracle database resides.

Entrust/PKI 5.0.2 for Oracle can be downloaded from the Entrust Web site:

```
http://www.entrust.com
```

Entrust/Toolkit Server Login and Entrust IPSEC Negotiator Toolkit can be downloaded from the Entrust Developer Network by registered members. Users can register for membership and download these products at the following Web address:

```
http://www.entrust.com//developer/memberships/registration.htm
```

Entrust/PKI 5.0.2 for Oracle

Entrust/PKI 5.0.2 for Oracle requires a database for storing information about Entrust users and the infastructure, and a Lightweight Directory Access Protocol (LDAP)-compliant directory for information such as user names, public certificates, and certificate revocation lists.

Entrust/PKI 5.0.2 for Oracle is comprised of the following software components:

- Entrust/Authority

- Entrust/RA

- Entrust/Entelligence

Entrust/Authority

Entrust/Authority is the centerpiece of Entrust/PKI. It performs core certificate authority, certificate, and user management functions, such as creating users and user profiles containing the user's credentials.

> **Note:** Oracle Corporation only supports the use of Entrust-enabled Oracle Advanced Security with versions of Entrust/Authority that run on Oracle9*i*.

> **See Also:** Chapter 7, Configuring Secure Sockets Layer Authentication, for information about certificate authorities.

Entrust/Authority supports unattended login, also called Server Login, which eliminates the need for a **Database Administrator** (DBA) to repeatedly enter a password for the Entrust profile on the server. With unattended login, the DBA need only enter a password once to open the Entrust profile for the server to authenticate itself to multiple incoming connections.

Entrust/RA

Entrust/RA is the administrator's secure interface to Entrust/Authority.

Entrust/Entelligence

Entrust/Entelligence provides support for user key management and single sign-on functionality on both clients and server by enabling Oracle9*i* server process access to incoming SSL connections.

Entrust/Toolkit Server Login 5.0.2

Entrust/Toolkit Server Login Toolkit Release 5.0.2 is required for single sign-on functionality on servers operating on UNIX platforms.

Entrust/Server Login Toolkit provides single sign-on by enabling Oracle9*i* server process access to incoming SSL connections. Without this capability, a database administrator or other privileged user would have to enter the password for the Entrust profile on the server for every incoming connection.

You can download Entrust/Toolkit Server Login from the Entrust Web site:

```
http://www.entrust.com/developer/software/files/desc_serverlogin.cfm
```

Entrust IPSEC Negotiator Toolkit 5.0.2

The Entrust IPSEC Negotiator Toolkit Release 5.0.2 is required on both clients and servers for integrating the Oracle Advanced Security SSL stack with Entrust/PKI, enabling SSL authentication to use Entrust profiles.

You can download the IPSEC Negotiator Toolkit from the Entrust Web site:

```
http://www.entrust.com/developer/software/index.htm
```

Entrust Authentication Process

Figure 8–1 illustrates the following Entrust authentication process:

1. The Entrust user on the Oracle client establishes a secure connection with the server using SSL and Entrust credentials.

2. The Oracle SSL adapter on the server communicates with the Entrust Authority to check the certificate revocation status of the Entrust user.

> **Note:** Figure 8–1 does not include client and server profiles creation, which is presumed.

Figure 8–1 Entrust Authentication Process
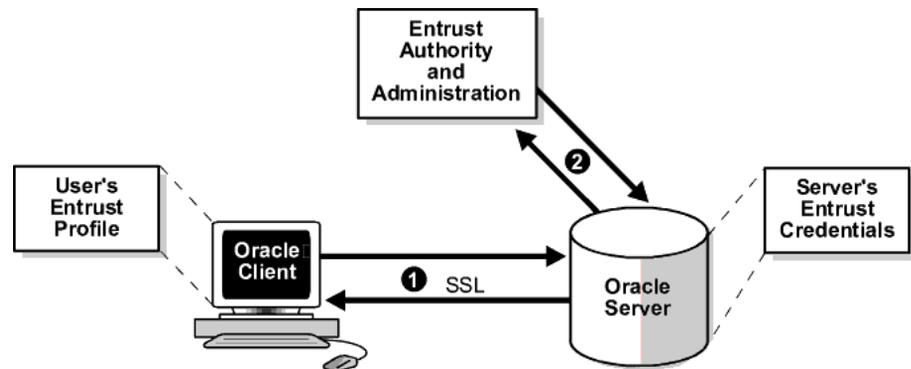
> **See Also:** How SSL Works in an Oracle Environment: The SSL Handshake on page 7-6

Enabling Entrust Authentication

This section describes the following tasks that enable Entrust-enabled Oracle Advanced Security SSL authentication:

- Creating Entrust Profiles
- Installing Oracle Advanced Security and Related Products
- Configuring SSL on the Client and Server
- Configuring Entrust on the Client
- Configuring Entrust on the Server
- Creating Database Users
- Logging Into the Database

Creating Entrust Profiles

This section describes how to create Entrust profiles. Entrust profiles can be created by either administrators or users. On UNIX platforms, administrators create the Entrust profiles for all clients. On Windows NT platforms, users can be permitted to create their Entrust profiles.

Administrator-Created Entrust Profiles

Administrators create Entrust profiles as follows:

1. The Entrust administrator adds the Entrust user using the Entrust/RA tool.

   > **See Also:** The Entrust administration documentation for information about creating Entrust Users

2. The administrator enters the user's name and password.
3. The Entrust Authority creates the profile, or `.epf` file.
4. The administrator securely sends all profile-related files to the user. The preset password can be changed by the user.

User-Created Entrust Profiles

Entrust users create their own Entrust profiles as follows:

1. The Entrust administrator adds the Entrust user using the Entrust/RA tool. In the New User dialog box, the Create Profile option should be deselected.

   > **See Also:** The Entrust administration documentation for information about creating Entrust profiles

2. The user receives a secure e-mail notification from the administrator that contains a reference number, authorization code, and expiration date.

3. The user navigates to the Create Entrust Profiles screen in Entrust/Entelligence as follows:

   Start>Programs>Entrust>Entrust Profiles>Create Entrust Profiles

4. The user enters the reference number, authorization code, and expiration date provided in the e-mail notification, creating a profile, or .epf file, and the Entrust initialization file.

Installing Oracle Advanced Security and Related Products

For Oracle Advanced Security Release 9.0.1, Entrust support installs in *Typical* mode. A single Oracle installation supports the use of *both Oracle Wallets and Entrust Profiles.*

> **See Also:** The Oracle9*i* installation documentation for your platform.

Notes:

- Installing Entrust on a UNIX server uses different parameters than in prior releases
- See Also: Configuring Entrust on a UNIX Server on page 8-11

Configuring SSL on the Client and Server

Configure SSL on the client and server.

> **See Also:** Chapter 7, Configuring Secure Sockets Layer
> Authentication, for information about configuring SSL on the client
> and server; skip the section that describes the Oracle wallet
> location.

Configuring Entrust on the Client

The steps for configuring Entrust on the client vary according to the type of
platform:

- Configuring Entrust on a UNIX Client

- Configuring Entrust on a Windows NT Client

Configuring Entrust on a UNIX Client

If the client resides on a non-Windows NT platform, perform the following steps:

1. Set the `JAVA_HOME` variable to JDK or JRE location.

   For example:

   ```
   >setenv JAVA_HOME $ORACLE_HOME/JRE
   ```

2. Set `WALLET_LOCATION` in the `sqlnet.ora` file.

   For example:

   ```
   WALLET_LOCATION =
     (SOURCE =
         (METHOD = ENTR)
         (METHOD_DATA =
             (PROFILE = profile_location)
             (INIFILE = initialization_file_location)
         )
     )
   ```

Configuring Entrust on a Windows NT Client

If the client resides on a Windows NT platform, ensure that the
Entrust/Entelligence component is installed on the client and perform the following
steps to set up the entrust credentials.

1. Set `WALLET_LOCATION` in the `sqlnet.ora` file.

   For example:

```
WALLET_LOCATION =
  (SOURCE =
      (METHOD = ENTR)
      (METHOD_DATA =
            (INIFILE = initialization_file_location)
      )
  )
```

where *initialization_file_location* is typically `c:\WinNT`.

2. Choose the Entrust icon on the system tray to open the Entrust_Login dialog box.

3. Log on to Entrust by entering the profile name and password.

Configuring Entrust on the Server

The steps for configuring Entrust on the server vary according to the type of platform:

- Configuring Entrust on a UNIX Server
- Configuring Entrust on a Windows NT Server

Configuring Entrust on a UNIX Server

If the server is a UNIX platform, ensure that the Entrust/Server Login Toolkit component is installed on the server and perform the following steps:

> **See Also:** System Components on page 8-4 for information about downloading the Entrust/Toolkit Server Login.

1. Stop the Oracle database instance.

2. Set the `wallet_location` parameter in the `sqlnet.ora` and `listener.ora` files to specify the paths to the server's profile and the Entrust initialization file:

```
WALLET_LOCATION =
  (SOURCE =
      (METHOD = ENTR)
      (METHOD_DATA =
          (PROFILE = profile_location)
          (INIFILE = initialization_file_location)
      )
  )
```

3. Enter the binder command to create unattended login credentials, or `.ual` files.

For example:

```
binder
```

4. Enter the path to the profile, the password, and the path to the Entrust initialization file. A message informs you that you have successfully created a credential file.

5. Start the Oracle database instance.

Configuring Entrust on a Windows NT Server

If the server is a Windows NT platform, ensure that the Entrust/Entelligence component is installed on the client and perform the following steps:

> **See Also:** System Components for information about downloading Entrust/Entelligence.

1. Stop the Oracle database instance.

2. Set the `wallet_location` parameter in the `sqlnet.ora` and `listener.ora` files to specify the paths to the server's profile and the Entrust initialization file:

```
WALLET_LOCATION =
  (SOURCE =
      (METHOD = ENTR)
      (METHOD_DATA =
          (PROFILE = profile_location)
          (INIFILE = initialization_file_location)
      )
  )
```

3. Run the Entrust binder command to create unattended login credentials, or `.ual` files as follows:

Start>Programs>Entrust Toolkit>Server Login>Entrust Binder

4. Enter the path to the profile, the password, and the path to the Entrust initialization file. A message informs you that you have successfully created a credential file.

5. Start the Oracle database instance.

Creating Database Users

Create global user in the database based on the **distinguished name (DN)** of each Entrust user.

For example:

```
SQL> create user jdoe identified globally as
'cn=jdoe,o=oracle,c=us';
```

where `"cn=jdoe, o=oracle, c=us"` is the Entrust distinguished name of the user.

Logging Into the Database

1. Use SQL*Plus to connect to the Oracle instance as follows:

```
sqlplus /@tns_service_name
```

where `tns_service_name` is the service name of the Oracle instance.

The Entrust_Login dialog box appears.

2. Enter the path to the profile and the password.

3. If you did not specify a value for the `WALLET_LOCATION` parameter, you are prompted to enter the path to the Entrust initialization file.

> **Note:** Oracle Corporation recommends that the initialization file be specified in the `WALLET_LOCATION` parameter file.

Issues and Restrictions

The Entrust-ready designation from Entrust typically requires that a partner product integration with Entrust is done using an Entrust toolkit. This means that an application must be specifically modified to work with Entrust.

For example, Oracle has modified its SSL libraries to access an Entrust profile instead of an Oracle wallet. Accordingly, the Entrust profile is not accessible from standard SSL libraries.

In addition, the following restrictions apply:

■ The use of Entrust components for digital signatures in applications based on Oracle is not supported.

- The Entrust-enabled Oracle Advanced Security integration is only supported with versions of Entrust/PKI Release 5.0.2 running on Oracle9*i*.

- The use of earlier releases of Entrust/Authority with Entrust-enabled Oracle Advanced Security is not supported.

- Interoperability between Entrust and non-Entrust PKIs is not supported.

- Entrust has certified Oracle Internet Directory version 2.1.1 for Release 8.1.7 (and subsequent).

Troubleshooting Entrust In Oracle Advanced Security

This section describes how to diagnose errors returned from Entrust to Oracle Advanced Security users.

> **Note:** Entrust returns the following generic error message to Oracle Advanced Security users:
>
> ```
> ORA-28890 "Entrust Login Failed"
> ```
>
> This troubleshooting section describes how to get more details about the underlying error, and how to diagnose the problem.

ORA-28890 Entrust Login Failed

Problem

SQLPLUS login on an Entrust-enabled Oracle client errors out with the following generic error message:

```
ORA-28890 Entrust Login Failed
```

Getting Details

To get more detail on the exact Entrust error, turn on TRACING for SQLPLUS by specifying the following parameters in `sqlnet.ora`:

- TRACE_LEVEL_CLIENT=16
- TRACE_DIRECTORY_CLIENT=c:\temp
- TRACE_FILE_CLIENT=clitrc

Search for the word `IKMP` within the created TRACE file. The TRACE file contains information about the exact error code returned by Entrust API.

Checklist

1. **Windows NT:** Log into Entrust/Entelligence, if you have not already done so, and retry.

2. **Windows NT:** Confirm through the Windows/Control Panel/Services applet that the Entrust Login Interface service has started and is running.

3. **Windows NT:** If the parameter SSL_ENTRUST_INI_FILE is not specified in `sqlnet.ora`, the Entrust initialization file (entrust.ini) must reside in c:\WINNT.

4. Due to a known FIPS mode incompatibility, Entrust logins may fail with the following error message:

   ```
   The software authentication filed. (error code - 162).
   ```

   Contact Entrust support to resolve this issue.

5. Due to a known symbol conflict between Entrust and Oracle libraries, Entrust login might fail with the following error message:

   ```
   Algorithm self-test failed. (error code - 176).
   ```

   Contact Entrust support to resolve this issue.

6. Confirm that Entrust/Authority, as specified in the Entrust Initialization file, is accessible and running.

7. Confirm that the profile password is correctly entered.

8. If Oracle database server fails to log into Entrust, confirm that the unattended credential file (.ual) is generated using a valid password. Also, confirm that the versions for Entrust ServerLoginToolkit and Entrust IPSEC Negotiator Toolkit match (i.e., that the IPSEC Toolkit 5.0.2 works with ServerLoginToolkit 5.0.2).

9. Ensure that the Entrust initialization file has the following entry in the first section, Entrust Settings:

   ```
   IdentityLibrary = location
   ```

   where *location* is the location of `libidapi.so`, including the file name.

General Problems and Guidelines

1. **Windows NT:** Oracle Universal Installer might not list Oracle Entrust Adapter as one of the choices for the CUSTOM install option. This is because it is expecting the registry key

   ```
   \\HKEY_CURRENT_USER\Software\Entrust
   Technologies\Toolkits\Version\IPSec
   ```

   to be `"5.0.1"` (or `"5.0.2"`).

   Contact Oracle Support to obtain an Installer patch that resolves this issue.

2. As far as possible, confirm that all of the Entrust toolkits (IPSEC and ServerLogin) are the same version.

9

Configuring Multiple Authentication Methods

This chapter describes how to configure multiple authentication methods under Oracle Advanced Security, and how to use conventional user name and password authentication, even if you have configured another authentication method. This also chapter describes how to configure your network so that Oracle clients can use a specific authentication method, and Oracle servers can accept any method specified.

This chapter contains the following topics:

- Connecting with User Name and Password
- Disabling Oracle Advanced Security Authentication
- Configuring Multiple Authentication Methods
- Configuring Oracle9i for External Authentication

Connecting with User Name and Password

To connect to an Oracle database server using a user name and password when an Oracle Advanced Security authentication method has been configured, disable the external authentication (See: Disabling Oracle Advanced Security Authentication on page 9-3).

With the external authentication disabled, a user can connect to a database using the following format:

```
% sqlplus username/password@net_service_name
```

For example:

```
% sqlplus scott/tiger@emp
```

> **Note:** You can configure multiple authentication methods, including both externally authenticated users and password authenticated users, on a single database.

Disabling Oracle Advanced Security Authentication

To disable authentication methods:

1. Start Oracle Net Manager:

    - On UNIX:

        Run `netmgr` from $ORACLE_HOME/bin

    - On Windows NT:

        Select Start>Programs>Oracle-HOME_NAME>Network
        Administration>Oracle Net Manager

2. In the Navigator window, expand Local > Profile.

3. From the list in the right window pane, select Oracle Advanced Security; the
    Oracle Advanced Security tabbed window appears (Figure 9–1):

Figure 9–1   Oracle Advanced Security Authentication Window
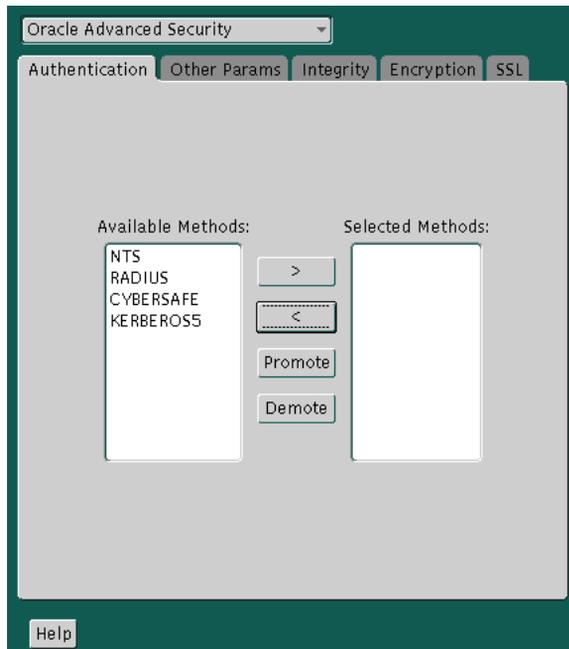
4. Choose the Authentication tab.

5. Sequentially move all authentication methods from the Selected Method list to the Available Methods list by selecting a method and choosing the left arrow [<].

6. Choose File > Save Network Configuration.

   The `sqlnet.ora` file is updated with the following entry:

   ```
   SQLNET.AUTHENTICATION_SERVICES = (NONE)
   ```

Configuring Multiple Authentication Methods

Many networks use more than one authentication method on a single security server. Accordingly, Oracle Advanced Security lets you configure your network so that Oracle clients can use a specific authentication method, and Oracle database servers can accept any method specified.

You can set up multiple authentication methods on both client and server systems either by using Oracle Net Manager, or by using any text editor to modify the `sqlnet.ora` file.

To add authentication methods to both clients and servers:

1. Start Oracle Net Manager:

    -   On UNIX:

        Run `netmgr` from $ORACLE_HOME/bin

    -   On Windows NT:

        Select Start>Programs>Oracle-HOME_NAME>Network Administration>Oracle Net Manager

2. In the Navigator window, expand Local > Profile.

3. From the list in the right window pane, select Oracle Advanced Security.

    The Oracle Advanced Security tabbed window appears (Figure 9–1).

4. Choose the Authentication tab.

5. Select a method listed in the Available Methods list.

6. Sequentially move selected methods to the Selected Methods list by choosing the right arrow [>].

7. Arrange the selected methods in order of desired use. To do this, select a method in the Selected Methods list, and choose Promote or Demote to position it in the list.

8. Choose File > Save Network Configuration.

    The `sqlnet.ora` file is updated with the following entry, listing the selected authentication methods:

    ```
    SQLNET.AUTHENTICATION_SERVICES =
    (RADIUS|CYBERSAFE|KERBEROS5)
    ```

> **Note:**
>
> - SecurID functionality is available through RADIUS; RADIUS support is built into the RSA ACE/Server.
>
> - See Also: Chapter 4, Configuring RADIUS Authentication

Configuring Oracle9*i* for External Authentication

This section describes the parameters you must set to configure Oracle9*i* for network authentication, using the following tasks:

- Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora
- Verifying that REMOTE_OS_AUTHENT Is Not Set to TRUE
- Setting OS_AUTHENT_PREFIX to a Null Value

> **See Also:**
> - The corresponding chapter in this guide for information about configuring a particular authentication method
> - Appendix B, Authentication Parameters

Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora

The following parameter must be set in the `sqlnet.ora` file for all clients and servers to enable each to use a supported authentication method:

```
SQLNET.AUTHENTICATION_SERVICES=(oracle_authentication_method)
```

For example, for all clients and servers using Kerberos authentication, the `sqlnet.ora` parameter must be set as follows:

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)
```

Verifying that REMOTE_OS_AUTHENT Is Not Set to TRUE

To verify that REMOVE_OS_AUTHENT is not set to TRUE, add the following parameter to the initialization file—in each database instance—when you configure the authentication method:

```
REMOTE_OS_AUTHENT=FALSE
```

> **Attention:** Setting REMOTE_OS_AUTHENT to TRUE can cause a security exposure, because it lets someone using a non-secure protocol, such as TCP, perform an operating system-authorized login (formerly referred to as an OPS$ login).

If REMOTE_OS_AUTHENT is set to FALSE, and the server cannot support any of the authentication methods requested by the client, the authentication service negotiation fails and the connection terminates.

If the parameter is set as follows in the `sqlnet.ora` file on either the client or server, the database attempts to use the supplied user name and password to login the user:

```
SQLNET.AUTHENTICATION_SERVICES=(NONE)
```

If REMOTE_OS_AUTHENT is set to FALSE, however, the connection fails.

Setting OS_AUTHENT_PREFIX to a Null Value

Authentication service-based user names can be long, and Oracle user names are limited to 30 characters. Oracle Corporation strongly recommends that you enter a null value for the OS_AUTHENT_PREFIX parameter in the initialization file used for the database instance as follows:

OS_AUTHENT_PREFIX=""

Note: The default value for OS_AUTHENT_PREFIX is OPS$; however, you can set it to any string.

Attention: **If a database already has the** OS_AUTHENT_PREFIX **set to a value other than** NULL **(" "),** *do not change it,* **since it can inhibit previously created, externally identified users from connecting to the Oracle server.**

To create a user, launch SQL*Plus and enter the following:

```
SQL> CREATE USER os_authent_prefix username IDENTIFIED EXTERNALLY;
```

When OS_AUTHENT_PREFIX is set to a null value (" "), enter the following to create the user king:

```
SQL> CREATE USER king IDENTIFIED EXTERNALLY;
```

The advantage of creating a user in this way is that the administrator no longer needs to maintain different user names for externally identified users. This is true for all supported authentication methods.

> **See Also:**
> - *Oracle9i Database Administrator's Guide*
> - *Oracle9i Heterogeneous Connectivity Administrator's Guide*

Part IV

Oracle DCE Integration

This part describes Oracle Distributed Computing Environment Integration (DCE). It contains the following chapters:

- Chapter 10, Overview of Oracle DCE Integration

- Chapter 11, Configuring DCE for Oracle DCE Integration

- Chapter 12, Configuring Oracle9i for Oracle DCE Integration

- Chapter 13, Connecting to an Oracle Database in DCE

- Chapter 14, DCE and Non-DCE Interoperability

> **Note:** Check the platform-specific installation documentation to verify that Oracle Advanced Security supports Oracle DCE integration on your platform.

10

Overview of Oracle DCE Integration

Oracle DCE Integration enables Oracle applications and tools to access Oracle9*i* servers in a distributed computing environment. This chapter briefly describes the Distributed Computing Environment (DCE) and the Oracle DCE Integration product. It contains the following topics:

- Oracle DCE Integration Requirements
- The Distributed Computing Environment
- Components of Oracle DCE Integration
- Flexible DCE Deployment
- Release Limitations

> **See Also:** Related Documentation on page -xxvi.

Oracle DCE Integration Requirements

System Requirements

Oracle DCE Integration requires Oracle Net and Oracle9*i*. It is based on the Open Software Foundation (OSF) DCE protocol (V1.1 and later).

Note that OSF has merged with X/OPEN, another standards group, to form The Open Group. This group is committed to continuing DCE support.

Backward Compatibility

Oracle servers running DCE Integration 2.3.2 and later are backward compatible with clients running SQL*Net/DCE 2.1.6 or 2.2.3; however, Release 2.1.6 clients cannot take advantage of external roles.

A client running DCE Integration 2.3.2 or later cannot connect to a SQL*Net/DCE 2.1.6 or 2.2.3 server. A DCE Integration Release 2.3.2 or later client requires a Release 2.3.2 or later server in order to connect to a database.

The Distributed Computing Environment

The Distributed Computing Environment (DCE) from the Open Group is a set of integrated network services that works across multiple systems to provide a distributed environment. The network services include remote procedure calls (RPCs), directory service, security service, threads, distributed file service, diskless support, and distributed time service.

DCE is the middleware between distributed applications and the operating system/network services and is based on a client/server model of computing. By using the services and tools that DCE provides, users can create, use, and maintain distributed applications that run across a heterogeneous environment.

Components of Oracle DCE Integration

Oracle DCE Integration has two components: DCE Communication/Security and DCE CDS Native Naming.

- DCE Communication/Security
- DCE Cell Directory Services Native Naming

DCE Communication/Security

This component has three principal features:

Authenticated RPC

Oracle DCE Integration provides authenticated Remote Procedure Call (RPC) as the transport mechanism that enables multi-vendor interoperability. RPC also uses some of the other DCE services, including directory and security services, to provide location transparency and secure distributed computing.

Integrated Security and Single Sign-On

Oracle DCE Integration works with the DCE Security service to provide security within DCE cells. It enables a user logged onto DCE to securely access any Oracle database without having to specify a user name or password. This is sometimes called **external authentication** to the database, or **single sign-on**. Clients and servers that are not running DCE authentication services can interoperate with systems that have DCE security by specifying an Oracle password.

Data Privacy and Integrity

Oracle DCE Integration uses the multiple levels of security that DCE provides to ensure data authenticity, privacy, and integrity. Users have a range of choices, from no protection to full encryption for each connection, with a guarantee that no data is modified in transit.

> **Note:** For parts of the network that do not use DCE, you can use the other security and authentication services that are part of Oracle Advanced Security. These services work with SQL*Net release 2.1 and above or with Oracle Net. They provide message integrity and data encryption services in non-DCE environments, letting administrators ensure that all network traffic is protected against unauthorized viewing or modification, regardless of the start or end point.

DCE Cell Directory Services Native Naming

The DCE Cell Directory Service (CDS) Native Naming component includes naming and location transparency.

DCE Integration registers Oracle9*i* connect descriptors in the DCE CDS, letting them be transparently accessed across the entire DCE environment. Users can connect to Oracle database servers in a DCE environment using familiar Oracle service names.

The DCE Cell Directory Service offers a distributed, replicated repository service for name, address, and attributes of objects across the network. Because servers register their name and address information in the CDS, Oracle clients can make location-independent connections to Oracle9*i* servers. Services can be relocated without any changes to the client configuration. An Oracle utility is provided to load the Oracle service names with corresponding connect descriptors into CDS. After this is done, Oracle connect descriptors can be viewed from a central location with standard DCE tools.

For location of services across multiple cells, either of the following options can be used:

- DCE Global Directory Service (GDS)
- Internet Domain Naming Service (DNS)

See Also:

- To configure DCE to use CDS naming, see Chapter 11, Configuring DCE for Oracle DCE Integration.

- To configure Oracle clients and servers to use CDS, see Chapter 12, Configuring Oracle9i for Oracle DCE Integration.

- For information on how Oracle Native Naming works with other Oracle name services, see the *Oracle Net Services Administrator's Guide*.

Flexible DCE Deployment

Oracle Advanced Security provides flexibility in your use of DCE services. You have the following options:

- You can use full DCE integration in your environment to integrate with all the DCE Secure Core services (RPC, directory, security, threads).

- You can use only the DCE directory services by using the DCE CDS Native Naming adapter, along with any conventional protocol adapter, such as TCP/IP.

Release Limitations

The following are limitations in Release 9.0.1 of Oracle Advanced Security:

- Only one listener address that uses the DCE protocol is permitted for each node.

- Database links must specify a user name and password to connect.

- This release of DCE Integration does not support the Oracle Multi-Protocol Interchange.

- This release does not work with the Oracle shared server.

11

Configuring DCE for Oracle DCE Integration

This chapter describes how to configure the Distributed Computing Environment (DCE) to use Oracle DCE Integration—after Oracle DCE Integration has been installed.

> **See Also:** Chapter 10, Overview of Oracle DCE Integration

To Configure DCE for Oracle DCE Integration:

The following tasks, performed by the DCE cell administrator, assume that a DCE cell has been configured and the systems being used are part of that cell:

- Task 1: Create New Principals and Accounts
- Task 2: Install the Key of the Server into a Keytab File
- Task 3: Configure DCE CDS for Use by Oracle DCE Integration

Task 1: Create New Principals and Accounts

Use the following procedure model to add server principals:

```
% dce_login cell_admin password
% rgy_edit
Current site is: registry server at /.../cell1/subsys/dce/sec/master
rgy_edit=>do p
Domain changed to: principal
rgy_edit=> add oracle
rgy_edit=> do a
Domain changed to: account
rgy_edit=> add oracle -g none -o none -pw oracle_password -mp cell_admin_
password
rgy_edit=> quit
bye
```

In this example, a DCE principal named `oracle` is created. The principal has a corresponding account with a password set to `oracle_password`. The account does not belong to any DCE group or DCE profile.

> **Note:** Perform this task on the server only once after DCE Integration has been installed; *do not perform this task on client systems.*

Task 2: Install the Key of the Server into a Keytab File

Install the key of the server into a keytab file, `dcepa.key`. This file contains the password of the principal under which the Oracle Net listener starts. The Oracle Net listener reads this file to authenticate itself to DCE. To generate the keytab file, enter the following:

```
% dce_login cell_admin password
```

```
% rgy_edit
Current site is: registry server at /.../cell1/subsys/dce/sec/master
rgy_edit=> ktadd -p oracle -pw Oracle_password -f
$ORACLE_HOME/dcepa/admin/dcepa.key
rgy_edit=>quit
bye
```

> **Note:**
>
> - Perform this task on the server only once after DCE Integration has been installed; *do not perform this task on client systems.*
>
> - Remember to substitute the full pathname for the $ORACLE_HOME variable. If the specified directories do not exist, create them before running the command; to create the directories. enter the following:
>
>   ```
>   mkdir $ORACLE_HOME/dcepa
>   mkdir $ORACLE_HOME/dcepa/admin
>   ```

Task 3: Configure DCE CDS for Use by Oracle DCE Integration

Step 1: Create Oracle Directories in the CDS Namespace

Enter the following after installing DCE Integration for the first time in a cell; create directories on all CDS replicas:

```
% dce_login cell_admin

Enter Password:(password not displayed)
$ cdscp
cdscp> create dir /.:/subsys/oracle
cdscp> create dir /.:/subsys/oracle/names
cdscp> create dir /.:/subsys/oracle/service_registry
cdscp> exit
```

> **Note:**
>
> - The directory `/.:/subsys/oracle/names` contains objects that map Oracle Net service names to connect descriptors, which are used by the CDS naming adapter.
>
> - The directory `/.:/subsys/oracle/service_registry` contains objects that map the service name in DCE addresses to the network endpoint that is used by both DCE protocol adapter clients and servers.

Step 2: Give Servers Permission to Create Objects in the CDS Namespace

Enter the following to add the principal `oracle` to the CDS-server group:

```
$ dce_login cell_admin
Enter Password:    (password not displayed)
$ rgy_edit
rgy_edit=> domain group
Domain changed to: group
rgy_edit=> member subsys/dce/cds-server -a oracle
rgy_edit=> exit
```

Step 3: Load Oracle Service Names into CDS

Load Oracle service names into the Cell Directory Service, as described in Chapter 12, Configuring Oracle9i for Oracle DCE Integration.

12

Configuring Oracle9*i* for Oracle DCE Integration

This chapter describes how to configure Oracle9*i* and Oracle Net to use Oracle DCE Integration after it has been successfully installed.

This chapter contains the following topics:

- DCE Address Parameters
- Configuring Oracle9i and Oracle Net

DCE Address Parameters

DCE addresses in the `listener.ora` and `tnsnames.ora` configuration files are defined by DCE parameters, illustrated below:

```
ADDRESS=(PROTOCOL=DCE)(SERVER_PRINCIPAL=server_name)(CELL_NAME=cell_name)
(SERVICE=dce_service_name))
```

These parameters are described by Table 12–1:

Table 12–1   DCE Address Parameters and Definitions

Component	Description
PROTOCOL	A mandatory field that identifies the DCE RPC protocol.
SERVER_PRINCIPAL	A mandatory field for the server and an optional field for the client. The server authenticates itself to DCE as this principal. This field is mandatory in the listener configuration file (`listener.ora`) and specifies the principal the server will start under. This field is optional in your local naming configuration file (`tnsnames.ora`) and specifies the principal of the server the client must connect to. If not specified, then one-way authentication is used. In this case, the client does not care what principal the server is running under.
CELL_NAME	An optional parameter. If present, it specifies the DCE cell name of the database. If this parameter is not set, the cell name defaults to the local cell (useful for single-cell environments). Optionally, the SERVICE parameter (described below) may specify the complete path (including the cell name) to the service, making this parameter unnecessary.
SERVICE	A mandatory field for both server and client. For the server, this is the service registered with CDS. For the client, this is the service name used when querying CDS for the location of the Oracle DCE servers. The default directory for storing service names in CDS is `/.../cellname/subsys/oracle /service_registry`. This service name can fully specify the path in CDS.

You can specify a service as follows:

```
SERVICE=/.../cell_name/subsys/oracle/service_registry/dce_service_name
```

Alternatively, you can specify:

```
SERVICE=dce_service_name
```

if `CELL_NAME=cell_name` is also specified.

In this case, the cell name defaults to the local cell. However, this way of specifying service names only works if you are operating within a single cell.

> **Note:** The *dce_service_name* in the service field might not be the same as that used by Oracle Net. The service name used by Oracle Net is mapped to the connect descriptor in a local naming configuration file (`tnsnames.ora`). The *dce_service_name* is part of the address within the connect descriptor.

Configuring Oracle9*i* and Oracle Net

To configure Oracle9*i* and Oracle Net to use Oracle DCE Integration, perform the following tasks:

- Task 1: Configure the Server
- Task 2: Create and Name Externally-Authenticated Accounts
- Task 3: Set up DCE Integration External Roles
- Task 4: Configure DCE for SYSDBA and SYSOPER Connections to Oracle Databases
- Task 5: Configure the Client
- Task 6: Configure Clients to Use DCE CDS Naming

Task 1: Configure the Server

To configure a server for DCE Integration, do the following:

1. Configure the listener configuration file (`listener.ora`) with DCE address information for all servers.

2. For servers in distributed systems that require database link connections to other servers, configure the `sqlnet.ora` and `protocol.ora` files with DCE address information.

   > **Note:** In this release, the configuration files `listener.ora,` `sqlnet.ora,` `tnsnames.ora,` and `protocol.ora` are located in the $ORACLE_HOME/network/admin directory.

   For a database server to receive connections from Oracle Net clients in a DCE environment, there must be an Oracle Net listener active on the server platform. This process listens for connections on a network address that is defined in the `listener.ora` configuration file.

   The SERVER_PRINCIPAL parameter designates what DCE principal the listener should be running under. In the sample below, the listener is running under principal oracle.

The following is a sample DCE address as it would appear in the `listener.ora` file.

```
LSNR_DCE=
  (ADDRESS=
      (PROTOCOL=DCE)
      (SERVER_PRINCIPAL=oracle)
      (CELL_NAME=cell1)
      (SERVICE=dce_svc))
SID_LIST_LSNR_DCE=
      (SID_DESC=
      (SID_NAME=ORASID)
       (ORACLE_HOME=/private/oracle9))
```

Task 2: Create and Name Externally-Authenticated Accounts

To use DCE authentication for logging onto an Oracle database, you must create database accounts that are authenticated externally. To enable secure external authentication, do the following:

> **Note:** The privileges shown in this section are the *minimum access privileges necessary.* The actual set of privileges needed depends upon the instance or application.

1. Verify that these lines are in the initialization parameter file:

   ```
   REMOTE_OS_AUTHENT=FALSE
   OS_AUTHENT_PREFIX=""
   ```

2. Verify that the initialization parameter file does not have a multi-threaded server (MTS) entry for DCE. For example, an entry such as the following is not permitted:

   ```
   mts_dispatchers="(PROTOCOL=dce)(DISPATCHERS=3)"
   ```

3. Ensure that you are logged on as a member of the DBA group. Restart the database instance for the changes to take effect.

4. At the SQL*Plus prompt, define users. Before doing so, decide whether you are, or ever will be, operating in a multi-cell DCE environment in which you let Oracle access across cell boundaries. The way you define users depends on whether they are connecting within a single cell or across cell boundaries.

Local Cell:

If users are connecting within a local cell, use the following format:

```
SQL> CREATE USER server_principal IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO server_principal;
```

For example:

```
SQL> CREATE USER oracle IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO oracle;
```

The entire CELL_NAME/SERVER_PRINCIPAL string must be 30 characters or less *(this is an* Oracle9*i restriction—not a restriction of the DCE adapter).*

For example:

```
SQL> CREATE USER "CELL1/ORACLE" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL1/ORACLE";
```

Multiple Cells:

If connecting to the database across multiple cells, specify both the *cell_name* and the *server_principal*, as illustrated below:

```
SQL> CREATE USER "CELL_NAME/SERVER_PRINCIPAL" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL_NAME/SERVER_PRINCIPAL";
```

You must enclose the externally-identified account name in double quotation marks, because the slash is a reserved character. Also, if the account (user) name is double-quoted, it must be capitalized.

For example:

```
SQL> CREATE USER "CELL1/ORACLE" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL1/ORACLE";
```

When using this format, set the following parameter in the `protocol.ora` configuration file to FALSE:

```
dce.local_cell_usernames=false
```

References to an Oracle account created in this manner must include the schema/account in the correct format. Consider requests for access to tables from another account. When a user references the tables in another account created within a local cell, the command might appear as follows:

```
SQL> SELECT * FROM oracle.emp
```

If a user wants to access tables in another account created for connections across cells, the command might appear as follows:

```
SQL> SELECT * FROM "CELL1/ORACLE" .emp
```

> **See Also:** *Oracle9i Heterogeneous Connectivity Administrator's Guide,* for more information about external authentication

Task 3: Set up DCE Integration External Roles

To set up external roles for DCE Integration, and enable connection to an Oracle database as SYSOPER or SYSDBA with DCE credentials, do the following:

1. Set the following parameter in the initialization parameter file:

```
OS_ROLES=TRUE
```

2. Restart the database.

3. Ensure that the DCE groups that map to Oracle roles adhere to the following syntax:

```
ORA_global_name_role[_[a][d]]
```

Table 12–2 describes the syntax components:

Table 12–2   Setting Up External Role Syntax Components

Component	Definition
ORA	Designates that this group is used for Oracle purposes
GLOBAL_NAME	The global name for the database
ROLE	The name of the role, as defined in the data dictionary
A or a	Optional character indicating that the user has admin privileges for this role
D or d	Optional character indicating the role is to be enabled by default at connect time

> **See Also:** *Oracle9i Database Administrator's Guide for more information about external roles*

4. Authenticate to DCE a user who is a member of a DCE group by entering the following commands:

```
dce_login
klist
```

Sample Output:

```
% dce_login oracle
```

Enter Password:

```
% klist
dce identity information:
Warning: Identity information is not certified
Global Principal: /.../ilab1/oracle
Cell:      001c3f90-01f5-1f72-ba65-02608c2c84f3 /.../ilab1
Principal: 00000068-0568-2f72-bd00-02608c2c84f3 oracle
Group:     0000000c-01f5-2f72-ba01-02608c2c84f3 none
Local Groups:
0000000c-01f5-2f72-ba01-02608c2c84f3 none
0000006a-0204-2f72-b901-02608c2c84f3 subsys/dce/cds-server
00000078-daf4-2fe1-a201-02608c2c84f3 ora_dce222_dba
00000084-89c8-2fe8-a201-02608c2c84f3 ora_dce222_connect_d
00000087-8a13-2fe8-a201-02608c2c84f3 ora_dce222_resource_d
00000080-f681-2fe1-a201-02608c2c84f3 ora_dce222_role1_ad
.
.
.
```

5. Connect to the database as usual.

The following sample output lists external roles (DBA, CONNECT, RESOURCE, and ROLE1) that have been mapped to DCE groups:

```
SQL> SELECT * FROM session_roles;

ROLE
----------------------------
CONNECT
RESOURCE
ROLE1

SQL> SET ROLE all;

Role set.
```

```
SQL> SELECT * FROM session_roles;

ROLE
----------------------------
DBA
EXP_FULL_DATABASE
IMP_FULL_DATABASE
CONNECT
RESOURCE
ROLE1

6 rows selected.

SQL> EXIT
```

Task 4: Configure DCE for SYSDBA and SYSOPER Connections to Oracle Databases

To configure DCE so that you can connect to an Oracle database as SYSOPER or SYSDBA with DCE credentials, do the following:

1. Create DCE groups that map to Oracle DBA and OPERATOR roles. DCE group names should adhere to the syntax described by Task 3: Set up DCE Integration External Roles on page 12-7. Add the externally authenticated user oracle as a member of the group(s).

```
$ dce_login cell_admin cell_admin_password
$ rgy_edit
rgy_edit=> domain group
Domain changed to: group
rgy_edit=> add ora_dce222_dba_ad
rgy_edit=> add ora_dce222_operator_ad
rgy_edit=> member ora_dce222_dba_ad -a oracle
rgy_edit=> member ora_dce222_operator_ad -a oracle
```

2. Add the GLOBAL_NAME parameter to the DCE address or TNS service name in the local configuration file tnsnames.ora.

```
ORADCE=
    (ADDRESS=
                (PROTOCOL=DCE)
                (SERVER_PRINCIPAL=oracle)
                (CELL_NAME=cell1)
                (SERVICE=dce_svc))
    (CONNECT_DATA=
```

```
                       (SID=ORASID)
                       (GLOBAL_NAME=dce222)))
```

3. Create the database user `oracle` as described by Task 2: Create and Name Externally-Authenticated Accounts on page 12-5.

4. Get DCE credentials for the externally authenticated user:

```
$ dce_login oracle oracle_password
$klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../dce.dlsun685.us.oracle.com/oracle
        Cell:        00af8052-7e94-11d2-b261-9019b88baa77
/.../dce.dlsun685.us.ora
cle.com
        Principal: 0000006d-88b9-21d2-9300-9019b88baa77 oracle
        Group:       0000000c-7e94-21d2-b201-9019b88baa77 none
        Local Groups:
                0000000c-7e94-21d2-b201-9019b88baa77 none
                0000006a-7e94-21d2-ad01-9019b88baa77 subsys/dce/cds-server
                00000076-8b53-21d2-9301-9019b88baa77 ora_dce222_dba_ad
                00000077-8b53-21d2-9301-9019b88baa77 ora_dce222_operator_ad

Identity Info Expires: 1999-12-04-10:28:22
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_43ae2600
Default principal: oracle@dce.dlsun685.us.oracle.com
Server: krbtgt/dce.dlsun685.us.oracle.com@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:22 to 1999-12-04-10:28:22
Server: dce-rgy@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:22 to 1999-12-04-10:28:22
Server: dce-ptgt@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:26 to 1999-12-04-02:28:26
Client: dce-ptgt@dce.dlsun685.us.oracle.com      Server:
krbtgt/dce.dlsun685.us.o
racle.com@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:26 to 1999-12-04-02:28:26
Client: dce-ptgt@dce.dlsun685.us.oracle.com      Server:
dce-rgy@dce.dlsun685.us.
oracle.com
        valid 1999-12-04-00:28:27 to 1999-12-04-02:28:26
```

> **Note:** List output shows the DCE group membership of `oracle`.

5. Connect to the Oracle database as SYSBDA or SYSOPER.

For example:

```
SQL> connect /@oradce as SYSDBA
```

Task 5: Configure the Client

To configure a client for DCE Integration, you must configure the following Oracle Net files with DCE address and parameter information:

- `protocol.ora`
- `sqlnet.ora`

Typically, CDS is used for name resolution. Thus, a local naming configuration file (`tnsnames.ora`) is not used, except when loading names and addresses into CDS.

Parameters in protocol.ora

There are four DCE parameters located in the `protocol.ora` file. Each parameter begins with the prefix `DCE.` to distinguish it from parameters relevant to other protocols. If default values are used for these four parameters, DCE Integration does not require a `protocol.ora` file. The parameters and their current defaults follow:

- DCE.AUTHENTICATION=*dce_secret*
- DCE.PROTECTION=*pkt_integ*
- DCE.TNS_ADDRESS_OID=1.3.22.1.5.1
- DCE.LOCAL_CELL_USERNAMES=TRUE

Configuration parameters are not case-sensitive: you can enter them in either uppercase or lowercase.

DCE.AUTHENTICATION

The DCE.AUTHENTICATION parameter is optional. It indicates the authentication value to be used for each DCE RPC. The client DCE_AUTHENTICATION value must be the same as the server DCE_AUTHENTICATION value. If this entry is not specified, cell-wide default authentication is used. The options follow:

Option	Description
NONE	No authentication
DCE_SECRET	DCE shared-secret key authentication (Kerberos)
DCE_SECRET	Default authentication level and recommended value
DEFAULT	Cell default

DCE.PROTECTION

DCE.PROTECTION is an optional field that specifies the data integrity protection levels for data transmission. The client DCE_PROTECTION level must be equal to or greater than the server DCE_PROTECTION level. If this entry is not specified, cell-wide default protection is used. The options follow:

Option	Description
NONE	Perform no protection for the current connection
DEFAULT	Use the default cell-wide protection level
CONNECT	Perform protection only when the client establishes a relationship with the server
CALL	Perform protection only at the beginning of each remote procedure call when the server receives the request
PKT	Ensure that all data received is from the expected client
PKT_INTEG	Ensure and verify that none of the data transferred between the client and server has been modified
PRIVACY	Perform protection as specified by all of the previous levels and also encrypt each RPC argument value and all user data in each call

DCE.TNS_ADDRESS_OID

DCE.TNS_ADDRESS_OID is an optional parameter that enables you to specify an alternative to the default value as follows:

```
DCE.TNS_ADDRESS_OID=1.3.22.1.x.x
```

> **See Also:** Step 2: Modify the CDS Attributes File and Restart the CDS on page 12-14.

DCE.LOCAL_CELL_USERNAMES

DCE.LOCAL_CELL_USERNAMES is an optional parameter that defines the format used to specify the principal name (username), with or without the cell name. The choice you make for this parameter should be determined by whether or not users are making connections across cells—with unique names. The default for DCE.LOCAL_CELL_USERNAMES is now TRUE (it was set to FALSE in the DCE Integration 2.1.6 release).

The associated options follow:

Option	Description
TRUE	The default value. Select TRUE if using just the SERVER_ PRINCIPAL format, without the CELL_NAME.
	An example of a user specified in this format is as follows:
	oracle
	TRUE is an appropriate option if users are making connections within a single cell, or if naming conventions in the network assure that users in different cells do not have duplicate names.
FALSE	Select FALSE when using the CELLNAME/SERVER_PRINCIPAL format. An example of a user specified in this format is as follows:
	CELL1/ORACLE
	FALSE is an appropriate option if users are making connections across cells and there can be users in different cells with identical name

Task 6: Configure Clients to Use DCE CDS Naming

Clients typically use CDS to resolve Oracle service names to addresses. Perform the following steps to configure CDS:

- Step 1: Enable CDS for use in Performing Name Lookup

- Step 2: Modify the CDS Attributes File and Restart the CDS

- Step 3: Create a tnsnames.ora File for Loading Oracle Connect Descriptors into CDS

- Step 4: Load Oracle Connect Descriptors into CDS

- Step 5: Delete or Rename the tnsnames.ora File

- Step 6: Modify the sqlnet.ora File to Resolve Names in CDS

> **Note:** Upon completion of this task, you can connect to an Oracle
> database in your DCE environment.

Step 1: Enable CDS for use in Performing Name Lookup

To use CDS for name resolution, the DCE Integration CDS Naming Adapter must
be installed on all clients and servers that use CDS. Also, the CDS namespace must
have been configured for use by DCE Integration.

> **See Also:** DCE Integration installation instructions, and Task 3:
> Configure DCE CDS for Use by Oracle DCE Integration on
> page 11-3.

For example, a service name such as `ORADCE` and its network address can be stored
in `DCE CDS`.

Users can typically connect to Oracle services using the familiar Oracle service
name if there are no domains or the database is in the user's default domain, as in
the following example:

```
sqlplus /@ORADCE
```

This example assumes that DCE externally-authenticated accounts are in use.

As an alternative name resolution service, use a local naming configuration file,
`tnsnames.ora`, when CDS is inaccessible. To do so, locate names and addresses of
all Oracle servers in the local `tnsnames.ora` file.

Step 2: Modify the CDS Attributes File and Restart the CDS

On all DCE machines where CDS naming will be used, add the object ID (OID) for
the CDS attribute TNS_Address to the CDS attributes file. (The object ID must be
the same across all machines.)

1. Add a line in the following format to the `/opt/dcelocal/etc/cds_`
   `attributes` file:

   ```
   1.3.22.1.5.1    TNS_Address    char
   ```

   The first four digits of this TNS_Address attribute value, `1.3.22.1.x.y`, are
   fixed, under DCE naming conventions. If the default TNS_Address object ID
   value `1.3.22.1.5.1` already exists in the `cds_attributes` file, you must
   specify a value for the object ID that is not already in use.

If you are unable to use the default value for the Object ID, you must specify the object ID in the `protocol.ora` file on the client.

If you had to specify a value other than the default value `1.3.22.1.5.1`, you must add the following parameter to the `protocol.ora` file:

```
DCE.TNS_ADDRESS_OID=1.3.22.1.x.y
```

Make sure that the object ID value in the cds_attributes file matches the value specified in the `DCE.TNS_ADDRESS_OID` parameter in the `protocol.ora` file.

2. Restart CDS on the system.

The command to restart CDS varies between different operating systems. On the Solaris platform, for example, you can use the following command to restart CDS:

```
/opt/dcelocal/etc/rc.dce restart
```

Step 3: Create a tnsnames.ora File for Loading Oracle Connect Descriptors into CDS

To load the Oracle service names and addresses into CDS, create or modify a local naming configuration file, `tnsnames.ora`. This file is used to map service names to addresses for use by Oracle Net.

This section describes the parameters that must be included in the `tnsnames.ora` file. The file contains a list of Oracle service names mapped to connect descriptors of destinations or endpoints in the network. The sample DCE address below shows a network address for an Oracle server with the Oracle service name ORADCE. It is used to connect to the service registered as `DCE_SVC` in the CDS directory

```
 /.../cell_name/subsys/oracle/names.
ORADCE=(DESCRIPTION=(ADDRESS=(PROTOCOL=DCE)(SERVER_PRINCIPAL=oracle)(CELL_
NAME=cell1)(SERVICE=DCE_SVC))(CONNECT_DATA=(SID=ORASID)))
```

> **Note:** In this example, the Oracle service name and the DCE service name are different, although they are frequently the same.

Parameter			
Name	**Type**	**Mandatory?**	**Description**
PROTOCOL=DCE	keyword value pair	Yes	Appears in the address sections of (i) `listener.ora`, a listener configuration file, and (ii) `tnsnames.ora`, a local naming configuration file.
SERVER_ PRINCIPAL	DCE Parameter	No	Appears in `tnsnames.ora`
SERVICE	DCE Parameter	Yes	The value given for the DCE parameter (SERVICE=*dce_service_name*) must be the same in `listener.ora` and `tnsnames.ora`
SID	Oracle Parameter	Yes	Identifies the Oracle system ID; each SID value must be unique on a node. This parameter is used locally only, and is not used in DCE CDS.

> **See Also:** *Oracle Net Services Administrator's Guide,* for information about `tnsnames.ora`, the local naming configuration file.

Step 4: Load Oracle Connect Descriptors into CDS

A separate utility called `tnnfg` is provided with Oracle DCE Integration to load connect descriptors into CDS. If you configure a new service name and address in `tnsnames.ora`, `tnnfg` adds the new service name and address to CDS. If you change the address for a particular service name, `tnnfg` updates the address for a particular service name.

To load the Oracle service names or aliases from `tnsnames.ora` into CDS, enter the following at the system prompt:

```
% dce_login cell_admin
% tnnfg dceload full_pathname_to_tnsnames.ora
% Enter Password:(password will not display)
```

Be sure to enter the full pathname of the `tnsnames.ora` *file, and ensure that the* `sqlnet.ora` *file exists in the same directory as the* `tnsnames.ora` *file.*

Step 5: Delete or Rename the tnsnames.ora File

You can keep `tnsnames.ora` available as a backup in case CDS becomes unavailable. To assure that CDS is routinely searched instead of `tnsnames.ora`, configure the `NAMES.DIRECTORY_PATH` parameter in a profile (`sqlnet.ora`), as described by Step 6: Modify the sqlnet.ora File to Resolve Names in CDS (the next section).

Step 6: Modify the sqlnet.ora File to Resolve Names in CDS

The parameters required in a profile (`sqlnet.ora`) depend upon the version of SQL*Net or Oracle Net you are using.

For a client or server to use DCE CDS Naming, the administrator must do the following:

1. Ensure that the CDS Naming Adapter has been installed on that node.

2. Add the following parameter to the `sqlnet.ora` file:

   ```
   NAMES.DIRECTORY_PATH=(dce, tnsnames, onames)
   ```

   The first name resolution service listed as a value for this parameter is used. If it is unavailable for any reason, the next name resolution service is used, and so forth.

13

Connecting to an Oracle Database in DCE

This chapter describes how to connect to an Oracle database after installing Oracle DCE Integration, and configuring both DCE and Oracle to use Oracle DCE Integration.

This chapter contains the following topics:

- Starting the Listener
- Connecting to an Oracle Database Server in the DCE Environment

Starting the Listener

To start the listener, do the following:

1. Enter the following commands:

    ```
    % dce_login principal_name password
    % lsnrctl start listener_name
    ```

    For example, if the listener name is LSNR_DCE in the `listener.ora` file, enter the following:

    ```
    % dce_login oracle orapwd
    % lsnrctl start LSNR_DCE
    ```

2. Verify that the server has registered its binding handler with rpcd:

    ```
    % rpccp show mapping
    ```

    Look for the line that includes the `dce_service_name` that is part of the listener address.

3. Verify that the service has been created by searching for the `dce_service_name` as follows:

    ```
    % cdscp show object "/.:/subsys/oracle/service_registry/dce_service_name"
    ```

    For example:

    The following command shows you the mapping in the CDS namespace that the listener has chosen for the endpoint:

    ```
    % cdscp show object "/.:/subsys/oracle/service_registry/dce_svc"

                 SHOW
              OBJECT    /.../subsys/oracle/service_registry/dce_svc
                  AT    1999-05-15-17:10:52
    RPC_ClassVersion = 0100
             CDS_CTS = 1999-05-16-00:05:01.221106100/aa-00-04-00-3e-8c
             CDS_UTS = 1999-05-16-00:05:01.443343100/aa-00-04-00-3e-8c
           CDS_Class = RPC_Server
    CDS_ClassVersion = 1.0
          CDS_Towers = :
                Tower = ncacn_ip_tcp:144.25.23.57[]
    ```

Connecting to an Oracle Database Server in the DCE Environment

Connect to an Oracle server in the DCE environment using one of the following methods:

- Method 1
- Method 2

Method 1

After externally-identified accounts have been set up, you can take advantage of DCE authentication to log into Oracle without providing any user name/password information. To use this single sign-on capability, just log in to DCE using a command like the following:

```
% dce_login principal_name password
```

For example:

```
% dce_login oracle orapwd
```

> **Note:** You only need to enter the `dce_login` command once. If you are already logged into DCE, you do not need to log in again.

You can now connect to an Oracle server without using a user name or password. Enter a command like the following:

```
% sqlplus /@net_service_name
```

where *net_service_name* is the database service name.

For example:

```
% sqlplus /@ORADCE
```

Method 2

From a client, you can still connect with a user name/password:

```
% sqlplus username/password@net_service_name
```

where *net_service_name* is the Oracle Net service name.

For example:

```
% sqlplus scott/tiger@ORADCE
```

14

DCE and Non-DCE Interoperability

This chapter describes how clients outside DCE can connect to Oracle servers in DCE, and how `tnsnames.ora`, a local naming configuration file, can be used for name lookup when CDS is accessible.

This chapter contains the following topics:

- Connecting Clients Outside DCE to Oracle Servers in DCE
- Sample Parameter Files
- Using tnsnames.ora for Name Lookup When CDS Is Inaccessible

Connecting Clients Outside DCE to Oracle Servers in DCE

Clients without access to DCE and CDS can still connect to Oracle servers in DCE using TCP/IP or some other protocol if a listener is configured to do this. If a listener has been configured in the `listener.ora` file on the server, non-DCE clients can use normal Oracle9*i* and Oracle Net procedures to connect to an Oracle server in DCE.

> **Note:** In this case, DCE security is not available to clients. Also, service names are resolved to network addresses and located in a `tnsnames.ora` file on the client, not using the CDS name server.

The following section includes samples of `listener.ora` and `tnsnames.ora` files as they would be configured if a client from outside of DCE wanted to connect to Oracle database servers in a DCE environment.

Sample Parameter Files

At least the following two Oracle parameter files are needed for successful client/server communications; create and modify these files using a text editor:

The parameter files are described in the following sections:

- The listener.ora File
- The tnsnames.ora File

The listener.ora File

The `listener.ora` file resides on the listener node. It defines listener characteristics and the addresses at which the listener listens.

In the following example, each element is displayed on a separate line, to show the file's structure. This is the recommended format, but you do not have to put each element on a separate line. Be sure to include all the appropriate parentheses, and to indent if you must continue an element on the next line.

This example assumes the UNIX operating system and the TCP/IP protocol for one listener, and the DCE protocol for another listener. A single listener can have multiple addresses. For example, instead of having two separate listeners for different database instances on a server node, you could have *one listener for both,* listening on both TCP/IP and on DCE. However, performance is improved with separate listeners.

```
LSNR_TCP=
     (ADDRESS_LIST=
         (ADDRESS=
                (PROTOCOL=IPC)
                (KEY=DB1)
         )
         (ADDRESS=
                (PROTOCOL=tcp)
                (HOST=rose)
                (PORT=1521)
         ))

  SID_LIST_LSNR_TCP=
         (SID_DESC=
                (SID_NAME=ORASID)
                (ORACLE_HOME=/usr/jprod/Oracle9i)
         )
LSNR_DCE=
```

```
      (ADDRESS=
        (PROTOCOL=DCE)
        (SERVER_PRINCIPAL=oracle)
        (CELL_NAME=cell1)
        (SERVICE=dce_svc))
       SID_LIST_LSNR_DCE=
        (SID_DESC=
          (SID_NAME=ORASID)
          (ORACLE_HOME=/usr/prod/oracle8))
#For all listeners, the following parameters list sample
#default values.

PASSWORDS_LISTENER=
STARTUP_WAIT_TIME_LISTENER=0
CONNECT_TIMEOUT_LISTENER=10
TRACE_LEVEL_LISTENER=OFF
TRACE_DIRECTORY_LISTENER=/usr/prod/Oracle9i/network/trace
TRACE File_LISTENER=listener.trc
LOG_DIRECTORY_LISTENER=/usr/prod/Oracle9i/network/log
LOG_FILE_LISTENER=listener.log
```

The tnsnames.ora File

This file resides on both the client and the server nodes. It lists the service names and addresses of all services on the network.

The following sample tnsnames.ora file maps the service name ORATCP to the connect descriptor that includes a TCP/IP address and the service name ORADCE to a connect descriptor that includes a DCE address.

```
ORATCP = (DESCRIPTION=
          (ADDRESS=
            (PROTOCOL=TCP)
            (HOST=rose)
            (PORT=1521)
          )
          (CONNECT_DATA=
            (SID=DB1)
          )
        )
ORADCE=(DESCRIPTION=
        (ADDRESS=
          (PROTOCOL=DCE)
          (SERVER_PRINCIPAL=oracle)
          (CELL_NAME=cell1)
```

```
 (SERVICE=dce_svc)
)
(CONNECT_DATA=
        (SID=ORASID)
)
)
```

To access the DB1 database, a user can use ORATCP to identify the appropriate connect descriptor.

For example:

```
sqlplus scott/tiger@oratcp
```

Using tnsnames.ora for Name Lookup When CDS Is Inaccessible

Typically, names are resolved into network addresses by CDS. Although the main purpose of the `tnsnames.ora` file (in the context of native naming adapters) is to load Oracle service names and network addresses into CDS, it could be used temporarily as a backup name resolution service if CDS is inaccessible.

SQL*Net Release 2.2 and Earlier

To use the `tnsnames.ora` file for name lookup and resolution, remove (or comment out) the "native name" parameters from the `sqlnet.ora` file on the client. To comment out the lines, add a pound sign (#) at the beginning of each line.

For example:

```
#native_names.use_native=true
#native_names.directory_path=(dce)
```

SQL*Net Release 2.3 and Oracle Net

You can use `tnsnames.ora` for name lookup and resolution when DCE CDS is unavailable if you have `TNSNAMES` listed as a value for the `NAMES.DIRECTORY_PATH` parameter in the `sqlnet.ora` file on the client.

For example:

```
names.directory_path=(dce, tnsnames)
```

This parameter enables you to list more than one names resolution method. The methods are tried in order. In this example, DCE is attempted first. If it is unsuccessful, TNSNAMES is tried next.

Part V

Oracle9*i* Enterprise User Security

This part describes Oracle9*i* directory and security integration functionality, which enables single sign-on in a client/server environment.

This part contains the following chapters, which describe how to set up enterprise user security in an Oracle database environment:

- Chapter 15, Managing Enterprise User Security

- Chapter 16, Using Oracle Wallet Manager

- Chapter 17, Using Oracle Enterprise Login Assistant

- Chapter 18, Using Oracle Enterprise Security Manager

15

Managing Enterprise User Security

Enterprise User Security lets you create and administer large numbers of users in a secure, **LDAP**-compliant directory service. This chapter describes the configuration and setup of Enterprise User Security.

This chapter contains the following topics:

Part I: Overview / Concepts:

- Overview of Enterprise User Security

- Shared Schemas

- Current User Database Links

- Enterprise User Security Components

- Deployment Considerations

Part II: Initial Configuration for SSL and Password Authentication

Part III: Final Configuration for SSL Authentication

Part IV: Final Configuration for Password Authentication

Part V: TroubleShooting Enterprise User Login

> **See Also:** Chapter 18, Using Oracle Enterprise Security Manager

Part I: Overview / Concepts

This part introduces Oracle Enterprise User Security, and describes its fundamental concepts.

Part I contains the following sections:

- Overview of Enterprise User Security
- Shared Schemas
- Current User Database Links
- Enterprise User Security Components
- Deployment Considerations

Overview of Enterprise User Security

This section describes fundamental concepts related to Enterprise User Security:

- Introduction to Enterprise User Security
- Enterprise Users and Authentication Methods
- Enterprise Users and Password Authentication
- Elements of Enterprise User Security
- The Enterprise User Security Process with SSL
- The Enterprise User Security Process with Passwords

Introduction to Enterprise User Security

Administrators must manage complex user information for the entire enterprise, keeping it both current and secure. This task becomes increasingly difficult as the use of technology expands and the user turnover rate increases throughout an enterprise. In a typical enterprise, every user can have multiple accounts on different databases, requiring users to remember passwords for each of these accounts. This can produce too many passwords for users to remember, and too many accounts for administrators to effectively manage.

With perhaps thousands of users accessing thousands of database accounts, administrators must devote substantial resources to user administration. Common information used by multiple applications—such as usernames, user office locations and telephone numbers, and system roles and privileges—is typically fragmented across the enterprise, contributing to data that is redundant, inconsistent, and difficult to manage.

There are security problems as well. For example, any time a user leaves a company or changes jobs, that user's privileges *should be changed the same day* in order to guard against their misuse. However, in a large enterprise, with user accounts and passwords distributed over multiple databases, an administrator may be unable to make such timely changes. In addition, users may elect to write down their passwords (making them easy for others to copy), or make them easy to remember (making them easy for others to guess), or simply choose the same password for multiple applications—all of which compromise the security of the enterprise.

Enterprise User Security addresses these user, administrative, and security challenges by centralizing storage and management of user-related information in an LDAP-compliant directory service. When an employee changes jobs in such an environment, the administrator need only modify information in one location—the

directory—to make effective changes in multiple databases and systems. This centralization can substantially lower administrative costs while materially improving enterprise security.

Enterprise users benefit from Enterprise User Security as well, through **single sign-on** or **single password authentication**, depending on the configuration chosen by the administrator.

Single sign-on lets users authenticate once, with subsequent authentications taking place transparently. It address the multiple password problem, and provides stronger, PKI-based authentication, combined with an improved user experience.

Single password authentication lets enterprise users authenticate to multiple databases with a single, global password—although each database requires a separate authentication. The password is securely stored in the centrally located, LDAP-compliant directory, and protected with security mechanisms including encryption and an **Access Control List (ACL)**. This approach reduces the number of passwords to remember, and eliminates the overhead of setting up SSL—both of which improve usability.

Enterprise User Security supports the following LDAP-compliant directory services:

- Oracle Internet Directory Release 2.1.1 or later

- Microsoft Active Directory
  (supports Oracle8i functionality only)

Oracle Advanced Security also provides a tool, Oracle Enterprise Security Manager, to create user entries in the directory and manage authorizations for those users.

> **See Also:** Chapter 18, Using Oracle Enterprise Security Manager

Enterprise Users and Authentication Methods

Enterprise User Security lets administrators manage two types of users in a single LDAP-compliant directory:

- Password-authenticated enterprise users

- SSL-authenticated enterprise users

Password-authenticated enterprise users use **single password authentication**.

SSL-authenticated users enjoy **single sign-on** and strong authentication, using industry-standard, interoperable X.509 v3 certificates—over Secure Sockets Layer (SSL) v3.

Each authentication method has its own set of selection criteria, as summarized by Table 15–1:

Table 15–1   Enterprise User Authentication: Selection Criteria

Selection Criteria, Applicable to:	
Password Authentication	**SSL-Authentication**
Supports password-based logins.	Provides stronger, systemwide security.
Does not support PKI, or PKI certificate-based client authentication.	Supports PKI, certificate-based authentication.
Does not employ SSL, wallets, or X.509 certificates.	Supports PKI, SSL, and industry-standard X.509 certificates.
Supports single, global user passwords, with separate authentications required for each database and application (**single password authentication**).	Supports **single sign-on** using SSL.
Provides faster processing, because there is no SSL processing required between clients and servers (supports Oracle Advanced Security native encryption). *Note: if your organization requires SSL, you cannot use password authentication across the enterprise.*	Somewhat slower connection processing, but with higher network security. *Note: if your organization requires SSL across the enterprise, you must use SSL-authentication.*
From the user/client point of view, password authentication may be viewed as easier-to-use, particularly for notebooks and home workstations.	SSL is more difficult to configure initially, but it provides stronger security.
Password authentication is compatible with either a two-tier or three-tier environment.	SSL-authentication is compatible with either a two-tier or three-tier environment.
Password authentication supports Oracle Release 7.3 or 8.0 clients, with an Oracle9*i* database.	SSL-authentication supports Oracle8*i* or Oracle9*i* clients with an Oracle9*i* database.

> **Note:**
>
> - This release extends Enterprise User Security support into three-tier environments. Oracle9*i* **proxy authentication** features enable (i) proxy of user names and passwords through multiple tiers, and (ii) credential proxy of X.509 certificates and distinguished names through multiple tiers. *This combination applies to both SSL-authenticated and password-authenticated enterprise users.*
>
> - See Also: Oracle9i Application Developer's Guide - Fundamentals

Enterprise Users and Password Authentication

Prior releases of Oracle Advanced Security use SSL processing to establish secure channels between (i) the client and the server, and (ii) the database server and the **LDAP**-compliant directory. The client authentication mechanism uses SSL and X.509 v3 certificates, requiring installed Oracle wallets on both the client and the server.

This release complements certificate-based authentication with password-based authentication for enterprise users, including the following principal features:

- **Password-based logins are enabled for enterprise users.** Enterprise users can use a single enterprise username and password to connect to multiple databases.

- **User credentials are stored in an LDAP-compliant directory.** Enterprise user credentials and authorizations are stored in a centralized LDAP-compliant directory. This includes the username and encrypted password verifier, enterprise roles, and mappings to individual schemas on each database to which the user has access rights.

- **Client-side SSL and wallets are not required.** Implementation of password-based authentication for enterprise users eliminates the requirement to install SSL and credential management tools on the client.

> **Note:** Installation and use of both SSL and Oracle wallets are still required on the server side—to establish a secure channel between the database and the LDAP-compliant directory, and between databases for current user database links.

Elements of Enterprise User Security

The following directory entries relate to Enterprise User Security:

- Enterprise Users
- Enterprise Roles
- Enterprise Domains
- User-Schema Mappings
- Database Server Entries
- Administrative Groups
- Security of User Database Login Information
- Enterprise User Security Elements

Enterprise Users

An **enterprise user** is one that is defined and managed in a directory. Each enterprise user has a unique identity across an enterprise. Enterprise user entries can reside at any location within the directory.

*The entries described below can only reside within an **Oracle Context**.*

Enterprise Roles

Enterprise users can be assigned **enterprise roles**, which determine their access privileges on databases. These enterprise roles are also stored and managed in a directory.

An enterprise role consists of one or more **global roles**, each one of which is defined in a specific database. A **global role** includes privileges contained in a database, but the global role is managed in a directory. An enterprise role is thus a container of **global roles**. For example, the enterprise role USER could contain the **global role** HRCLERK with its privileges on the Human Resources database, and the ANALYST role with its privileges on the Payroll database.

An **enterprise role** can be assigned to one or more enterprise users. For example, you could assign the enterprise role USER to a number of enterprise users who hold the same job. This information is protected in the directory, and only the administrator can manage users and assign their roles. A user can be granted local roles and privileges in a database in addition to enterprise roles.

An **enterprise domain** subtree includes enterprise role entries, each of which contains information about associated global roles on each server and authorized enterprise users. These are created and managed by the Domain Administrator by using Oracle Enterprise Security Manager.

> **See Also:** Administering Enterprise Roles on page 18-36

> **Note:** The database obtains a user's global roles when the user logs in. If you change a user's global roles in the directory, those changes do not take effect until the next time the user logs in.

Enterprise Domains

An **enterprise domain** is a group of databases and **enterprise roles**. An example of a domain could be the engineering division in an enterprise or a small enterprise itself. It is here, at the enterprise domain level, that the Domain Administrator, using Oracle Enterprise Security Manager, assigns enterprise roles to users and manages enterprise security. An enterprise domain subtree in a directory is composed of three types of entries: enterprise role entries (discussed by Enterprise Roles on page 15-7), User-Schema Mappings, and the Domain Administrator's group for that domain.

User-Schema Mappings

A user-**schema mapping** entry contains mapping information between a full or partial DN and an Oracle database user name. The users referenced in the mapping are connected to the specified schema when they connect to the database. User schema mapping entries can appear under database server entries, where they apply only to that database. They can also appear under domain entries, where they apply to all databases in that domain.

Mapping entries are created by the Domain Administrator for a particular domain, and are created by the **Database Administrator** for a particular database.

> **See Also:** Mapping an Enterprise User to a Shared Schema on page 15-23

Database Server Entries

A database server entry (represented as cn=sales in Figure 15–1) contains information about a database server. It is created by the Oracle Database Configuration Assistant during the database registration. A database server entry is the parent of database level *mapping entries* that contain mapping information between full or partial DNs and Oracle shared schema names. Database-level mapping entries are created by the **Database Administrator** by using Oracle Enterprise Security Manager. A **Database Administrator**'s group, containing administrators for that database, is located under the server entry.

Figure 15–1   Related Entries in an Oracle Context

* Examples of user-
  created entries.
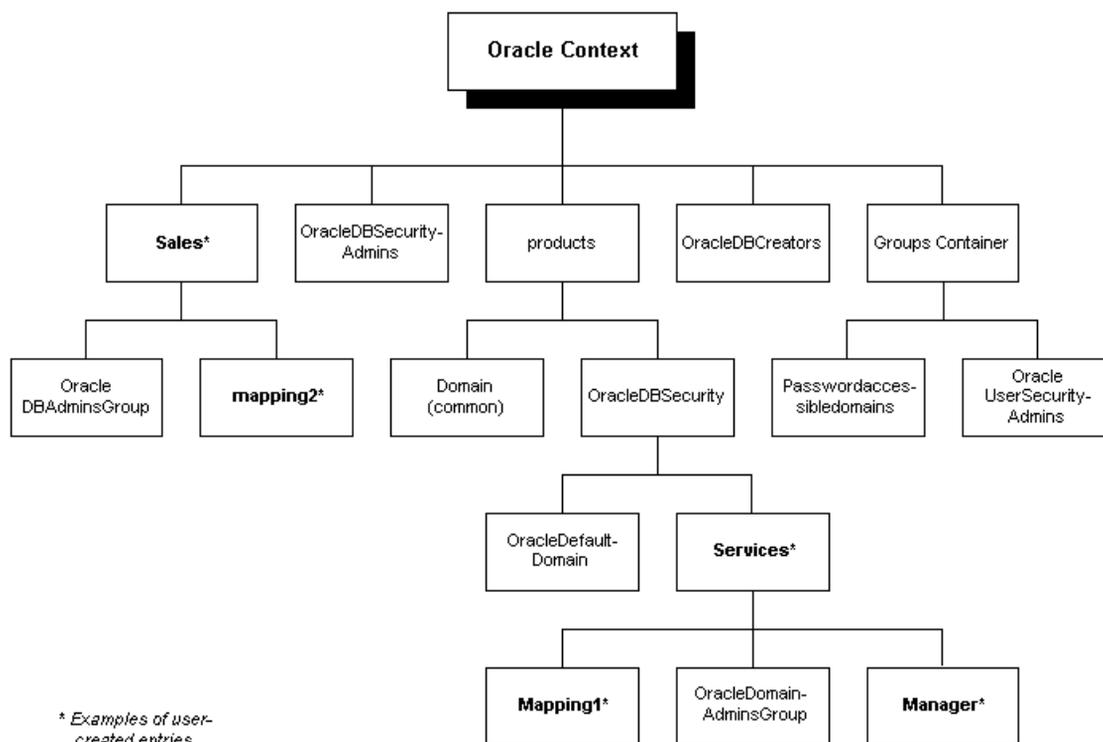
Administrative Groups

The Oracle Context contains Enterprise User Security-related administrative groups, each with its associated **Access Control List (ACL)**. The user who creates the Oracle Context with NetCA automatically becomes the first member of each of these groups. The relevant administrative groups in an Oracle Context are described in Table 15–2:

Table 15–2   Administrative Groups in an Oracle Context

Administrative Group	Description
OracleDBCreators	Members of the OracleDBCreators group (`cn=OracleDBCreators,cn=OracleContext...`) are in charge of creating new databases, and this includes registering each database in the directory by using the Oracle Database Configuration Assistant. They have `create` and `modify` access to database service objects and attributes. They can also modify the Default Domain.
	NetCA establishes these access rights during Oracle Context creation.
	In addition to the Oracle Context creator, other users can be added to this group by members of the OracleDBSecurityAdmins group by using Oracle Enterprise Security Manager.
OracleDBSecurityAdmins	Members of OracleDBSecurityAdmins (`cn=OracleDBSecurityAdmins,cn=OracleContext...`) have root privileges for the OracleDBSecurity subtree. They have `create`, `modify`, and `read` access for Enterprise User Security. They have permissions on all of the domains in the enterprise and are responsible for:
	■ Administering the OracleDBSecurityAdmins and OracleDBCreators groups
	■ Creating new enterprise domains
	■ Moving databases from one domain to another within the enterprise
	NetCA sets up these access rights during Oracle Context creation.
	In addition to the Oracle Context creator, members of this group can add other users to this group by using Oracle Enterprise Security Manager.

Table 15–2   Administrative Groups in an Oracle Context

Administrative Group	Description
OracleUserSecurity-Admins	Members of OracleUserSecAdmins (cn=OracleUserSecurityAdmins,cn=Groups,cn=OracleContext ...) are responsible for Oracle user security. For example, by default they can read wallet password hints and modify user passwords. The relevant ACL is set at the root of the directory by default.
Password Accessible Domains	Members of <cn=password accessible domains> are enterprise domains that contain databases enabled for password-authorized enterprise users.

You can also have a Domain Administrator responsible for managing a single domain. This administrator has lesser privileges than the **Database Security Administrator**. Similarly, you can have a **Database Administrator** responsible for a single database directory entry. These admin groups reside directly under the relevant database or domain entry.

> **Note:**   Do not modify the ACLs for the objects contained in an Oracle Context; doing so breaks the security configuration for these objects—and may break enterprise user functionality as well.

Security of User Database Login Information

Overview

In all secure password-based authentication methods, a server authenticates a client with a password verifier, typically a hashed version of the password—that must be rigorously protected. Password-based authentication to an Oracle database is no different; there is an Oracle proprietary password verifier, and it must be protected as well. This is true if the verifier is stored locally in the database or centrally in the directory.

In the current release an enterprise user's centrally-stored database password can be stored in a central directory service for access by multiple databases, and is viewable and shared by all trusted databases to which the user has access. Although the password verifier stored in the directory is not the **cleartext** password, it is still necessary to protect it from casual or unauthorized access. It is therefore *extremely important* to define password-related ACLs in the directory that are as restrictive as possible, while still enabling necessary access and usability.

Oracle tools help set up ACLs in the directory to protect these password verifiers. The Oracle recommended approach is intended to balance both security and usability considerations. If you require maximum security and can set up wallets for all users, you should require SSL connections (only) from users to databases. This SSL-only approach circumvents the entire directory password protection issue.

Setting Up ACLs

In the current release, there is a new *Password-accessible Domains group* in each Oracle Context. This group is created automatically when the context is created, and can be managed using Oracle Enterprise Security Manager. Enterprise domains with member databases that must view users' database password verifiers in the directory are placed into this group. Enterprise Security Manager can be used to place an appropriate ACL on user subtrees, so that databases in this group can read the password verifier for users in that subtree.

There are two steps involved in this (Oracle-supported) approach:

1. For a selected (Oracle9*i*) Oracle Context, determine which databases can accept password-authenticated connections. Use Oracle Enterprise Security Manager to place the domains containing those databases into the *Password-accessible Domains group*.

2. For a selected (Oracle9*i*) Oracle Context, Use Oracle Enterprise Security Manager to select the user search bases that contain users that require connection to databases in this context using password authentication. For user entries under these search bases, restrict access (to the password verifier) to authorized domains only—by checking the *Restrict Logon* checkbox.

   This step places a restrictive ACL on the selected user search base entry in the directory. That ACL specifies permitted access to the attribute that holds the Oracle database password verifier as follows:

   - Members of the *Password-Accessible Domains group* in the selected Oracle Context have *read access* to the attribute. All databases in the member domains can thus read the password verifier in order to authenticate enterprise users. Note that databases in domains excluded from membership in the *Password-Accessible Domains group* cannot accept password-authenticated connections.

   - Members of the *User Security Admins group* in the selected Oracle Context have *read and write access* to the attribute. User Security Administrators can thus create an initial database password for a newly created user, or reset a forgotten password. Note that a password verifier cannot be used to derive its original password.

- If the user search base is referenced by two different Oracle Contexts, and domains in both contexts require access from password-authenticated users, the password-assessable domains and `usersecurityadmins` groups from *both* contexts appear in the ACL.

- Users themselves require *read and write access* to the attribute. Accordingly, users can use Oracle Enterprise Login Assistant to change their respective database passwords.

- All other users are denied access to this attribute.

These ACLs restrict access to the Oracle database password verifier value only (the orcldbpassword attribute). No other attributes in the user entries are affected.

> **See Also:** *Oracle Directory Service Integration and Deployment Guide* for more information about the LDAP schema for Enterprise User Security

Enterprise User Security Elements

Figure 15–2 displays the Enterprise User Security elements—with SSL-authentication employed:

Figure 15–2   Enterprise User Security Elements (SSL-Authentication)
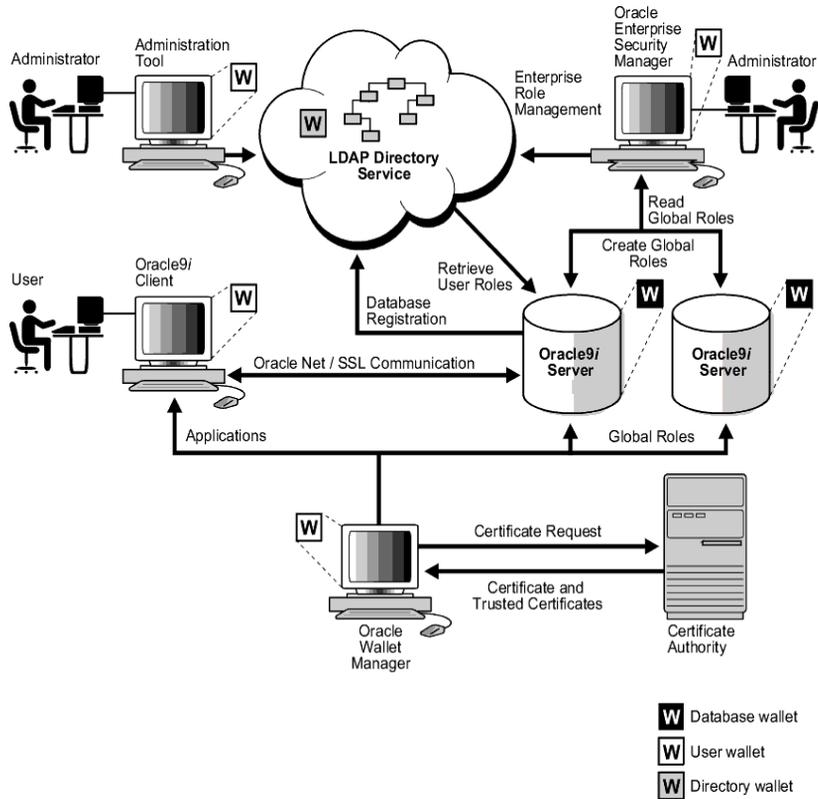
Figure 15–3 displays the Enterprise User Security elements—with password-authentication employed:

Figure 15–3   Enterprise User Security Elements (Password Authentication)
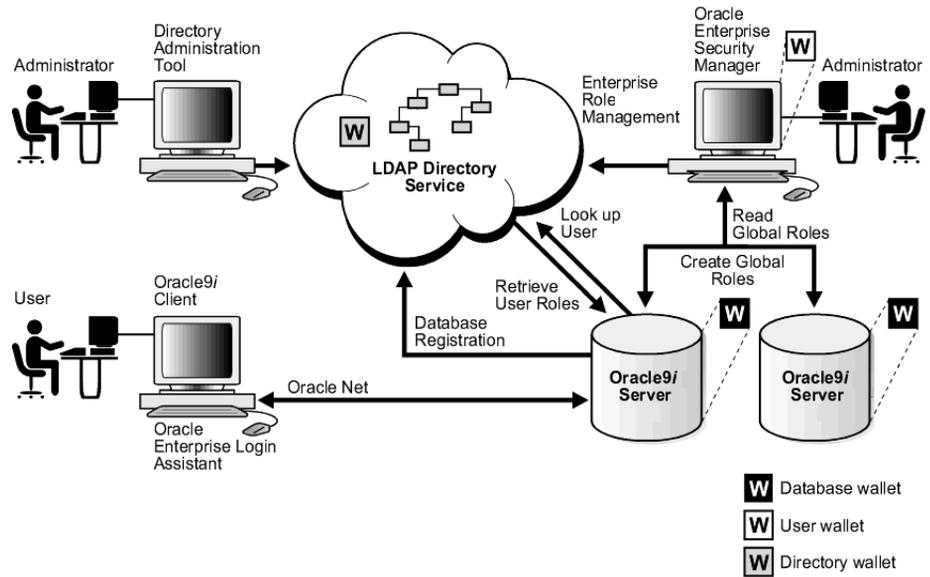
The Enterprise User Security Process with SSL

Figure 15–4 shows the operation of the Enterprise User Security process. For an SSL processing environment, the following assumptions apply:

- A wallet has been set up and configured for the user.

- The user is authenticated to the database through SSL.

Figure 15–4   How Enterprise User Security Works
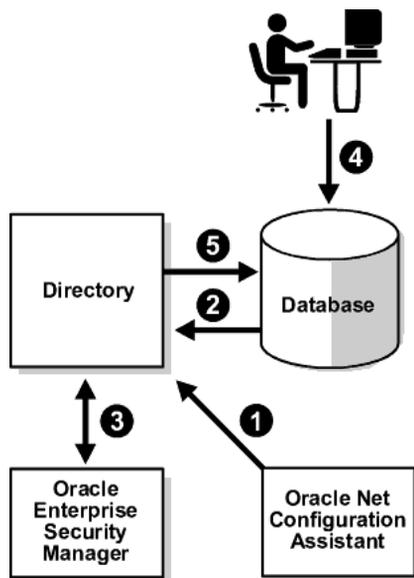
1. An administrator uses NetCA to (i) select the Oracle Context in the directory, or to (ii) create an Oracle Context as necessary.

2. A member of the OracleDBCreators group uses the Oracle Database Configuration Assistant to register the database with the directory.

3. An administrator uses Oracle Enterprise Security Manager to set up both enterprise users and enterprise roles in the directory and relevant domains.

4. A user initiates an SSL connection to the database (logs on), and the database uses SSL to authenticate the user; the database searches for the appropriate schema by accessing local tables.

5. If no appropriate user schema mapping is found locally, the database searches for one in the directory (2). If it finds one, the database retrieves the user's enterprise roles from the directory (5), and authorizes any associated global roles applicable to that database.

> **Note:** In a three-tier environment, enterprise users can authenticate to the database through any middle tier using **proxy authentication**.

The Enterprise User Security Process with Passwords

This section describes the operation of the Enterprise User Security process, with password-based authentication (See: Figure 15–4):

1. An administrator uses NetCA to (i) select the Oracle Context in the directory, or to (ii) create an Oracle Context as necessary.

2. A member of the OracleDBCreators group uses the Oracle Database Configuration Assistant to register the database with the directory.

3. An administrator uses Oracle Enterprise Security Manager to set up both enterprise users and enterprise roles in the directory and relevant domains, and to configure attributes in the context.

4. A user authenticates to the database (logs on) with a password.

5. As part of the authentication process, the database performs the following steps:

   - It looks up the username and retrieves the **distinguished name (DN)** and password verifier from the directory.

   - It authenticates the user based on the retrieved password verifier, and searches locally (on the database) for a schema exclusively owned by this user.

   - If it does not find an exclusive schema on the database, it searches for a shared schema in the directory.

   - It retrieves the user's enterprise roles, and authorizes any associated global roles applicable to that database.

> **Note:** In a three-tier environment, enterprise users can authenticate to the database through any middle tier using **proxy authentication**.

Shared Schemas

The following sections describe shared schemas, and how to set them up:

- Overview
- Configuring Shared Schemas
- Shared Schema Functionality and SSL
- Creating a Shared Schema
- Creating an Enterprise User in the Directory
- Mapping an Enterprise User to a Shared Schema

Overview

Users do not necessarily require individual accounts or schemas set up in each database. Alternatively, they can be granted access to common, **shared schemas** associated with target applications. For example, suppose that users *Tom*, *Dick*, and *Harriet* require access to the Payroll application on the Finance database. They do not need to create unique objects in the database, and therefore do not need their own schemas—they do need access to the Payroll schema.

Oracle9*i* Release 9.0.1 supports mapping multiple users stored in an enterprise directory to shared schema on an individual database. This separation of users from schemas reduces administration costs by reducing the number of user accounts on databases. It means that you do not need to create an account for each user—a user schema—in multiple databases, in addition to creating the user in the directory. Instead, you can create a user in one location, the enterprise directory, and map the user to a shared schema that other enterprise users can also be mapped to. For example, if Tom, Dick and Harriet all access both the Sales and the Finance databases, you do not need to create an account for each user on each of these databases. Instead, you can create a single shared schema on each database, such as SALES_APPLICATION and FINANCE_APPLICATION, respectively, that all three users can access. A typical environment might have some 5,000 enterprise users mapped to just one of three or four shared schemas. Alternatively, you can map multiple users to a single shared schema (a *guest* schema), and assign enterprise roles to each user.

Summary:

- Shared schemas eliminate the need to have a dedicated database schema on each database for each enterprise user.

- Each enterprise user can be mapped to a shared schema on each database the user needs to access. The user is granted access to each such shared schema when the user connects to a database.

- Shared schemas lower the cost of managing users in an enterprise.

Configuring Shared Schemas

To configure shared schemas, the local Database Administrator must create at least one database schema in a database. Enterprise users can be mapped to this schema.

In the following example, the administrator creates a shared schema and maps users to it:

- The administrator creates a global shared schema called EMPLOYEE and the global role HRMANAGER on the HR database.

- The administrator uses Oracle Enterprise Security Manager to create and manage enterprise users and roles in the directory. For example, the administrator creates enterprise user Harriet and an enterprise role named MANAGER. The administrator then assigns the global role HRMANAGER to the enterprise role MANAGER.

- The administrator assigns enterprise roles to enterprise users in the directory. For example, the administrator assigns the enterprise role MANAGER to Harriet.

- The administrator uses Oracle Enterprise Security Manager to map the user Harriet in the directory to the shared schema *EMPLOYEE* on the HR database.

When Harriet connects to the database, she is automatically connected to the EMPLOYEE schema and is given the global role HRMANAGER. Multiple enterprise users can be mapped to the same shared schema. For example, the enterprise security administrator can create another enterprise user *Scott* and map Scott to the EMPLOYEE schema. From that point on, both Harriet and Scott automatically use the EMPLOYEE schema when connecting to the HR database, but each can have different roles—and can be individually audited.

Shared Schema Functionality and SSL

Shared schema functionality relies on SSL or user passwords for authentication to the database. For a discussion of password-based authentication, See: Enterprise Users and Password Authentication on page 15-6.

A discussion of the SSL authentication process follows:

- Prior to connecting to a database, an enterprise user enables Autologin by providing a wallet password to Oracle Wallet Manager or Oracle Enterprise Login Assistant.

- When connecting (on the client side), Oracle Advanced Security performs an SSL handshake with the database, during which it passes the user's unique certificate to the server; this handshake authenticates the user to the server.

- The database extracts the user's DN from the user's certificate, and checks the database to see if a user with that DN owns an exclusive schema (not shared).

- If the database does *not* find the DN locally, it looks up the appropriate DN mapping in the directory. This DN mapping object in the directory associates a user with a database schema. The database may find the following DN mapping entries:

  - Full DN (entry-level) mapping

    This method associates the DN of a single directory user with a particular schema on a database. It results in one mapping entry for each user.

    When using full DN mapping, each enterprise user can be mapped either to a unique schema, or to a shared schema.

  - Partial DN (subtree-level) mapping

    This method lets multiple enterprise users share part of their DN to access the same shared schema. This method is useful if multiple enterprise users are already grouped under some common root in the directory tree. The subtree that these users share can be mapped to a shared schema on a database. For example, you can map all enterprise users in the subtree for the engineering division to one shared schema, BUG_APPLICATION, on the bug database. Note that the root of the subtree is not mapped to the specified schema.

  - No mapping at all

- If the database does *not* find either the DN locally or an appropriate DN mapping object in the directory, it refuses the user's connection to the database.

  If the database *does* find either the DN locally or the appropriate DN mapping object in the directory, the database lets the user log on.

- The database maps the user to the associated schema.

  For example, suppose that Harriet is trying to connect to the HR database, but the database does not find Harriet's DN (in the database). In this case, the HR database looks up the Harriet's DN in the directory. The directory has a

mapping of Harriet to the shared schema EMPLOYEE and returns this schema. The database logs Harriet in and connects her to the EMPLOYEE schema.

- The database retrieves this user's global roles for this database from the directory.

  The database also retrieves from its own tables any local roles and privileges associated with the database schema to which the user is mapped.

  The database uses both the global and the local roles to determine the information that the user can access.

  > **Note:** You can configure the database so that it uses *local roles only* and does not look up global roles in the directory. To do this, do not register the database in the directory with the Oracle Database Configuration Assistant.
  >
  > Alternatively, to temporarily disable directory lookup, you can comment-out the parameter RDBMS_SERVER_DN in the init.ora configuration file—and restart the database.
  >
  > If this configuration is set, the database uses only local roles to determine what the user can access. This lets customers use SSL for client authentication, without having to manage user privileges centrally. *This configuration does not work with mapped users and shared schemas—because the mapping information must be stored in the directory.*

  See Also:

  - Chapter 7, Configuring Secure Sockets Layer Authentication, for information about SSL
  - Chapter 16, Using Oracle Wallet Manager, for information about wallets and Oracle tools for managing them

Continuing this example, assume that the enterprise role MANAGER contains the global roles ANALYST on the HR database, and USER on the Payroll database. When Harriet, who has the enterprise role MANAGER, connects to the HR database, she uses the schema *EMPLOYEE* on that database.

- Her privileges on the HR database are determined by:
  - The global role ANALYST

- Any local roles and privileges associated with the *EMPLOYEE* schema on the HR database

- When Harriet connects to the Payroll database, her privileges are determined by:

  - The global role USER

  - Any local roles and privileges associated with the EMPLOYEE schema on the Payroll database

Creating a Shared Schema

The syntax for creating a shared schema is:

```
CREATE USER [shared schema name] IDENTIFIED GLOBALLY AS ''
```

For example, the administrator for the HR database creates a shared schema for the user SALES_APPLICATION as follows:

```
CREATE USER sales_application IDENTIFIED GLOBALLY AS ''
```

> **Note:** *There is no space* between the single quotation marks in the syntax for creating a shared schema.

Creating an Enterprise User in the Directory

You can use Oracle Enterprise Security Manager to create enterprise user entries one at a time. To load large numbers of entries, use other LDAP processes such as the Oracle Internet Directory bulk load tool.

> **See Also:** Your LDAP directory documentation (such as for Oracle Internet Directory or Microsoft Active Directory)

Mapping an Enterprise User to a Shared Schema

The mapping between enterprise users and a schema can be done in either the database or the directory.

The mapping is done in the directory by means of one or more mapping objects. A mapping object is used to map the **distinguished name (DN)** of a user to a database schema that the user will access. You create a mapping object by using Oracle Enterprise Security Manager. This mapping can be one of the following:

- A Full DN (entry-level) mapping

- Partial DN (subtree-level) mapping

  When determining the appropriate schema to which it is to connect the user, the database uses the following precedence rules:

  – It looks for that schema locally (in the database).

  – If it does not find it locally, it searches the directory. Within the directory, it looks under the *server* entry, first for a full DN mapping, then for a partial DN mapping.

  – If it does not find a mapping entry under the server entry, it looks under the *domain* entry, first for a full DN mapping, then for a partial DN mapping.

  – If it does not find a mapping entry in the domain entry, the database refuses the connection.

You can grant privileges to a specified group of users by granting roles and privileges to a database schema. Every user sharing such a schema gets these local roles and privileges in addition to personal enterprise roles. However, you should exercise caution when doing this, because every user who is mapped to this shared schema can exercise the privileges assigned to it. *Accordingly, Oracle does not recommend assigning roles and privileges to a shared schema.*

Current User Database Links

Oracle9*i* supports current user database links for both SSL-authenticated and password-authenticated enterprise users, which let you make a procedural connection to a second database as another user and with that user's privileges—though it does not require that the second user's credentials be stored in the database link definition. Such access is limited to the scope of the database link procedure.

For example, a current user database link lets Harriet, a user of the Finance database, procedurally access the Accounts Payable database by connecting as *Scott*, and using Scott's credentials.

For Harriet to access a current user database link to connect to the schema Scott, Scott must be a schema created as IDENTIFIED GLOBALLY in both databases. Harriet, however, can be a user identified in one of three ways:

- By a password

- GLOBALLY

- EXTERNALLY

To create Scott as a global user in both the Accounts Payable and Finance databases, you must enter the following command in each database:

```
CREATE USER Scott IDENTIFIED GLOBALLY as 'CN=Scott,O=nmt'
```

Note that the syntax for creating this kind of schema is slightly different from the syntax for creating a shared schema described in Creating a Shared Schema on page 15-23. In this case, the schema is Scott's alone. In order for the current user database link to work, the schema created for Scott cannot be shared with other users.

Current user database links operate only between trusted databases within a single enterprise domain—databases within the domain trust each other to authenticate users. You specify a *domain* as trusted by using Oracle Enterprise Security Manager. By default, if current user database links are enabled for a domain by using Enterprise Security Manager, they will work for all databases within that domain. To specify a database as untrusted that is part of a trusted enterprise domain, use the PL/SQL package DBMS_DISTRIBUTED_TRUST_ADMIN. To obtain a list of trusted servers, use the TRUSTED_SERVERS view.

See Also:

- *Oracle9i Heterogeneous Connectivity Administrator's Guide*, for additional information about current user database links

- *Oracle9i SQL Reference*, for more information about syntax

- *Oracle9i Supplied PL/SQL Packages and Types Reference*, for information about the PL/SQL package DBMS_DISTRIBUTED_TRUST_ADMIN

- *Oracle9i Database Reference*, for information about the TRUSTED_SERVERS view

Enterprise User Security Components

Enterprise User Security functionality uses the following administration tools:

- Oracle Enterprise Security Manager
- Oracle Enterprise Login Assistant
- Oracle Wallet Manager

Oracle Enterprise Security Manager

Oracle Enterprise Security Manager is an administration tool that provides a graphical user interface to help you manage enterprise users, enterprise domains, databases, and enterprise roles that are stored in a directory service. Use Oracle Enterprise Security Manager to:

- Administer enterprise domains.
- Administer both SSL-based and password-based user authentication and authorization.
- Administer database, security, and domain administrators.
- Create user-schema mappings.
- Associate enterprise users with Oracle Contexts.

> **See Also:** Chapter 18, Using Oracle Enterprise Security Manager

Oracle Enterprise Login Assistant

Use Oracle Enterprise Login Assistant to enable and disable Autologin, to upload wallets to, or download wallets from a directory, and to change a user wallet password. This tool lets enterprise users use SSL to connect to multiple services with a single sign-on. Oracle Enterprise Login Assistant masks the complexity of SSL, wallets, enterprise users, and the process of authenticating to multiple databases.

Oracle Enterprise Login Assistant also supports password authentication, letting users securely access multiple databases and applications using a single password, entered once for each session.

You can use ELA to change any of the following passwords:

- Directory password (Oracle Internet Directory only)
- Database password

- User wallet password (required for SSL connection only)

The directory password is the password used to bind to Oracle Internet Directory. The database password, on the other hand, is the single, global password that enterprise users enter into an application (such as SQL*Plus), in order to authenticate to multiple databases.

The user wallet password is used to access user wallets, stored locally (on the client) or in the directory. User wallets are required for SSL-based authentication, but they are not required for password-based authentication.

> **See Also:** Chapter 17, Using Oracle Enterprise Login Assistant

Oracle Wallet Manager

Oracle Wallet Manager is a standalone Java application that wallet owners and security administrators use to manage and edit the security credentials in their Oracle wallets. Wallet Manager tasks include:

- Generating a public/private key pair.
- Creating a certificate request for submission to a **certificate authority**.
- Installing a certificate for the entity.
- Configuring trusted certificates for the entity.
- Creating a wallet that can be used later for enabling Autologin, using Oracle Enterprise Login Assistant.
- Enabling Autologin to enable access to PKI-based services.
- Uploading a wallet to, and downloading a wallet from a directory.

> **Note:** Although password-authenticated enterprise users do not require client-side wallets, Oracle Wallet Manager is still required—to support secure connections between the databases and the directory (server-to-server connections require SSL and server-side wallets).

> **See Also:** Chapter 16, Using Oracle Wallet Manager, for detailed information about using this application

Deployment Considerations

Consider the following before deploying Oracle Enterprise Security Manager:

- Security Aspects of Centralizing Security Credentials
- Database Membership in Enterprise Domains

Security Aspects of Centralizing Security Credentials

Beyond the general benefits that flow from the centralization of enterprise users and their associated credentials, there are a number of security-related benefits and risks that should be reviewed.

One security benefit is that centralizing management makes it easier and faster to administer users, credentials, and roles, and to quickly revoke a user's privileges on all applications and databases across the enterprise. With centralized management, the administrator can delete a user in one place to revoke all global privileges, minimizing the risk of retaining unintended privileges.

Another security benefit is that it can be more secure to centrally control security information, because you can centralize the organization's security expertise. Specialized, security-aware administrators can manage all aspects of enterprise user security, including directory security, user roles and privileges, and database access. This is a substantial improvement over the traditional model, where Database Administrators are typically responsible for everything on the databases they manage, including security.

The downside is that, while Oracle Internet Directory is a secure repository, there is a security challenge—and inherent risk—in centralizing credentials in any publicly accessible repository. Although centralized credentials can be protected at least as securely as distributed credentials, the very nature of centralization increases the consequences of inadvertent credential exposure to unauthorized parties. It is therefore imperative to limit the privileges of administrators, to set restrictive Access Control Lists (ACLs) in the directory, and to implement good security practices in the protection of security credentials when they are temporarily outside of the directory.

Database Membership in Enterprise Domains

Consider the following criteria when defining the database membership of a domain:

- Current user **database links** operate only between databases within a single **enterprise domain**.

- Password authentication for enterprise users is defined at the domain level. Database membership in a domain should therefore be defined accordingly. If one or more databases are intended to only support SSL-based certificate authentication, they cannot be combined in the same domain with password-authenticated databases.

- Enterprise roles are defined at the domain level. To share an **enterprise role** across multiple databases, the databases must be members of the same domain.

Part II: Initial Configuration for SSL and Password Authentication

This part describes the initial Enterprise User Security configuration tasks—*for both SSL and password authentication.*

This part contains the following tasks:

- Task 1: Install or Identify a Certificate Service
- Task 2: Install and Configure a Directory Service
- Task 3: Install and Configure the Database
- Task 4: Configure the Database for SSL
- Task 5: Create the Wallet and Start the Listener
- Task 6: Verify Database Installation
- Task 7: Create Global Schemas and Roles

Task 1: Install or Identify a Certificate Service

Oracle Wallet Manager requires you to have a **certificate authority** (CA) in your environment. You can use a CA vendor's certificates, or you can use your own CA that can process PKCS#10 certificate requests in Base 64 format and return X509v3 certificates—also in Base 64 format.

> **See Also:** Chapter 16, Using Oracle Wallet Manager, for a description of certificate authorities and Oracle Wallet Manager

Task 2: Install and Configure a Directory Service

Conceptually, there are four major prerequisites for an Oracle RDBMS to communicate with the directory:

- The Oracle Schema must be installed in the directory, if one does not already exist. The Oracle Schema contains all the Oracle-specific object classes and attributes. The Oracle Schema is pre-installed in Oracle Internet Directory. If you are using an older version of OID, NetCA updates the schema. The new version of the Oracle Schema is backward compatible.

- An **Oracle Context** must be created in the directory, if one does not already exist. This is the subtree where all the Oracle-specific objects are stored (Figure 15–1). Oracle Internet Directory is shipped with a pre-installed Oracle Context at the directory root. You can create other Oracle Contexts elsewhere, but do not remove the root context.

- There must be a local file (ldap.ora) that tells the database how to connect to the directory, including host, port, and directory type— and the location in the directory of the Oracle Context.

- There must be an entry in the directory representing the database, so it can bind (log in) to the directory.

NetCA performs the first three steps (called *directory service access configuration),* and the Oracle Database Configuration Assistant (DBCA) performs the fourth. Note that you must create the Oracle Context at least once for each directory, but you must create an ldap.ora file for each ORACLE_HOME used to access the directory. If there is no context, NetCA prompts you to create one; if one already exists, it prompts you to choose one to use. If there is no Oracle schema installed in the directory (which is done automatically for Oracle Internet Directory), or it is not current, NetCA prompts you to install or update it.

If you run the recommended *custom* database install, NetCA and DBCA automatically complete the directory-related configuration.

If you install the Oracle9*i* database using a *typical* install, NetCA and DBCA do not complete the directory-related configuration. In this case, you must run both NetCA and DBCA in standalone mode.

To install and configure a directory service:

- Step 1: Install and Start an SSL-Enabled LDAP v3-Compliant Directory Service
- Step 2: Prepare the Directory for Enterprise User Support
- Step 3: Create Administrative Users

Step 1: Install and Start an SSL-Enabled LDAP v3-Compliant Directory Service

Oracle9*i* Release 9.0.1 supports the following internet directories:

- Oracle Internet Directory Release 2.1.1 or later
- Microsoft Active Directory

Setting up the directory for two-way SSL authentication requires the creation of an Oracle wallet for the directory. If you are using Oracle Internet Directory, use Oracle Wallet Manager and Oracle Internet Directory to create a wallet for the directory and to configure SSL. If you are using Microsoft Active Directory, see that product's documentation for instructions about enabling two-way SSL authentication.

Step 2: Prepare the Directory for Enterprise User Support

Ensure that the directory has an Oracle9*i* schema and an appropriate Oracle Context installed.

- Upgrading the Oracle Schema

  The Oracle Schema is pre-installed in Oracle Internet Directory. If you are using an older version of Oracle Internet Directory, you can upgrade the schema using NetCA. The Oracle9*i* version is backward compatible.

     See Also: Task 7: Create Global Schemas and Roles on page 15-49

- Upgrading the Oracle Context

  Oracle Internet Directory is shipped with a pre-installed Oracle Context at the directory root. You can use NetCA to create an Oracle Context.

  You must upgrade an Oracle8*i* Oracle Context before registering an Oracle9*i* database with that context in the directory. You can use this upgraded Oracle Context to register any Oracle8*i* databases that are created in the future. If you have a combination of Oracle8*i* and Oracle9*i* databases deployed, set the

VERSIONCOMPATIBILITY parameter to *8i* and *9i,* using Oracle Enterprise Security Manager. Oracle recommends that you set this to *9i* if you deploy only Oracle9*i* databases. This parameter determines if some of the database security attributes must be represented in the directory in two places to support both Oracle8*i* and Oracle9*i* databases. If you are deploying only Oracle8*i* databases, there is no requirement to upgrade the Oracle Context—and no requirement to set this parameter.

Note:

- If Oracle Enterprise User Security has been set up with an Oracle8*i* Oracle Context, the associated security information (enterprise users, roles, privileges, domains...) cannot be upgraded when the Oracle8*i* context is upgraded. *In this case, you must create a fresh Oracle9i Oracle Context and re-create the security information.*

- If you have never used Oracle Enterprise User Security, you can upgrade the Oracle8*i* Oracle Context to an Oracle9*i* context and proceed with setup and configuration.

See Also: *Oracle Directory Service Integration and Deployment Guide*

Step 3: Create Administrative Users

If they do not already exist, create enterprise users in the directory who are authorized to perform the following functions:

- Register databases.

- Administer database security.

- Create and manage enterprise domains.

> **Note:**   It is possible to perform all Enterprise User Security administrative functions as a single administrator. If you choose this approach, you must only create one administrative user entry.
>
> *Note, however, that this creates a condition where there is no separation of security functions—not considered good security practice.*
>
> Oracle Enterprise User Security provides the capability to create many different kinds of administrators, to achieve separation of security functions, and a more secure enterprise environment.

Task 3: Install and Configure the Database

To install and configure one or more databases:

- Step 1: Install Oracle9i Release 9.0.1 Database Software
- Step 2: Set Up Directory Access for ORACLE_HOME
- Step 3: Authorize Users for Administrative Functions
- Step 4: Use Oracle Database Configuration Assistant to Register the Database in the Directory

Step 1: Install Oracle9*i* Release 9.0.1 Database Software

Use the Oracle Universal Installer to install Oracle9*i* databases. Before installation, obtain the following from the directory administrator:

- A directory login name and password. The login name must belong to someone in the Database Installation Administrators Group (cn=OracleDBCreators). The user who created the Oracle Context is automatically a member of this group.
- The location of an Oracle Context that should contain the database's future entry.

During installation, select Oracle9*i* Enterprise Edition and the *Custom installation type* with Oracle Advanced Security.

> **See Also:**   Oracle9*i* Release 9.0.1 installation documentation for your platform, for detailed instructions about how to install and create a database

NetCA runs automatically at the end of the Oracle9*i* installation process; see Step 2 for related instructions (if you did not choose the custom install, NetCA must be started manually).

Step 2: Set Up Directory Access for ORACLE_HOME

To configure directory service access configuration, use NetCA:

1. Run NetCA:

   - **Windows NT**: Select `Start->Programs->Oracle-<ORACLE_HOME_ NAME>->Network Administration->Oracle Net Configuration Assistant`

   - **UNIX:** Type `NetCA` at the command line.

2. Select `Directory Usage Configuration`; choose `Next`.

3. Choose `Select an already configured directory server to use`; choose `Next`.

4. In the Directory Type window, select your Directory Type (Oracle Internet Directory, for example).

5. In the Directory Location window:

   - Enter the name of your directory host system in the Hostname field.

   - Enter the non-SSL port number in the Port field; the default is `389`.

   - Enter the SSL port number in the SSL Port field; the default is `636`.

   - Choose `Next`.

6. Select the appropriate Oracle Context.

7. Choose `Finish` to exit NetCA.

8. Verify that an `ldap.ora` file is located in the following directory path:

   UNIX and Windows NT: `<ORACLE_HOME>/network/admin`

Figure 15–5   Example: The ldap.ora File
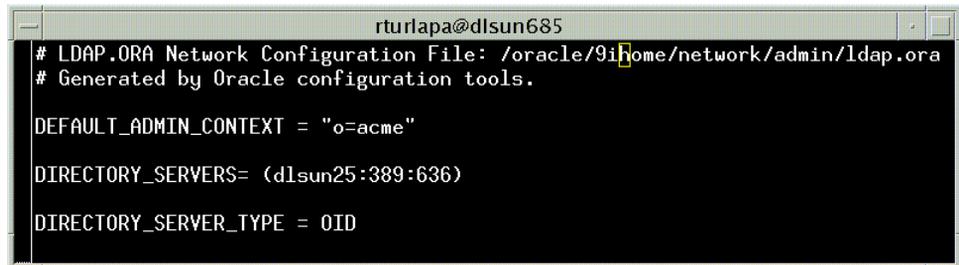
```
rturlapa@dlsun685
# LDAP.ORA Network Configuration File: /oracle/9ihome/network/admin/ldap.ora
# Generated by Oracle configuration tools.

DEFAULT_ADMIN_CONTEXT = "o=acme"

DIRECTORY_SERVERS= (dlsun25:389:636)

DIRECTORY_SERVER_TYPE = OID
```

> **See Also:**   *Oracle Net Services Administrator's Guide*, for instructions about setting up directory access for the database (*Configuring Naming Methods*)

Step 3: Authorize Users for Administrative Functions

To register your database in the directory (using DBCA), you must provide directory credentials for (i) a user in the Database Installation Administrator's group, or (ii) a user in the Context Administrator's group, or (iii) the directory superuser. If you choose the first or second approach, you may need to add appropriate users to that group *before* running DBCA. Use Oracle Enterprise Security Manager to put the appropriate directory users into the *Database Installation Admin group*—so they can register the database in the directory.

> **Note:**   This group is called the Database Installation Admin group by Oracle Enterprise Security Manager, but the actual group name in the directory is `OracleDBCreators.`

> **See Also:**   Chapter 18, **Using Oracle Enterprise Security Manager**

To authorize users for administrative functions:

1. Start Oracle Enterprise Security Manager and log in:

2. Using Oracle Enterprise Security Manager, select the appropriate context and choose the *Admins* tab; add appropriate users to the following groups:

   - Database Installation Administrators

   - Context Administrators

Step 4: Use Oracle Database Configuration Assistant to Register the Database in the Directory

1. If you performed a *typical* database installation, start DBCA in standalone mode as follows:

    -   **Windows NT**: Select `Start->Programs->Oracle-<ORACLE_HOME_ NAME>->Database Administration->Database Configuration Assistant`

    -   **UNIX:** Enter `dbca` at the command line.

2. If you performed a *custom* database installation, Oracle Database Configuration Assistant prompts you to register the database in the directory. Choose `Yes`.

3. If you are running DBCA in a standalone mode, select `Configure database options in a database` and choose `Next`.

    -   Select a database and choose `Next` (this process typically takes about a minute to complete); the final Oracle Database Configuration Assistant window appears (Figure 15–6):

Figure 15–6   Oracle Database Configuration Assistant Window (Finish)

    -   Choose `Yes, Register the Database,` and enter the directory credentials for a user in the Database Installation Administrators group.

- Select database features as desired; choose Next.

- Accept the correct server mode.

- Choose `Finish`; the Locate Initialization File window appears. DBCA Locate Initialization File Window

- Select the appropriate initialization file and choose OK. Oracle Database Configuration Assistant creates a new database service object underneath your chosen Oracle Context in the directory. The credentialed user is the one with the correct access permissions to create the new database entry in the directory. This user obtained these permissions by being placed into the Database Installation Administrators group by Oracle Enterprise Security Manager. When DBCA registers the database in the directory, it also adds the `RDBMS_SERVER_DN` initialization parameter to the `init.ora` file for the database. This parameter represents the **distinguished name (DN)** of the database as registered in the directory.

Notes:

- Do not modify the `RDBMS_SERVER_DN` parameter manually by editing the `init.ora` file; this can make the locally stored DN for the database inconsistent with the actual DN in the directory. If this happens, change the `init.ora` parameter to match the DN for the database in the directory.

- Use your directory administration tool to verify that a database entry and subtree exist under your Oracle Context in the directory.

- If you get an error message, use Oracle Enterprise Security Manager to check that the user you are providing credentials for is in the Database Installation Administrators group.

Task 4: Configure the Database for SSL

To configure the database for SSL:

- Step 1: Configure Oracle Net for Listener and Database SSL Support

- Step 2: Configure SSL Service Name

- Step 3: Configure the Listener

- Step 4: Review the .ORA Files

Step 1: Configure Oracle Net for Listener and Database SSL Support

Oracle Net must be configured for SSL on both the **listener** and the database. The listener must have a listening endpoint that is configured for the TCP/IP with SSL protocol, and the location of the database wallet must be specified. Use Oracle Net Manager to do this (See: Enabling SSL on page 7-14):

1. With `ORACLE_HOME` set to the databases' home directory, run Oracle Net Manager:

   - **Windows NT**: Select `Start->Programs->Oracle-<ORACLE_HOME_NAME>->Network Administration->Oracle Net Manager`.

   - **UNIX:** Enter `netmgr` at the command line.

2. Configure Profile:

   - Select `Profile`.

   - Select `Oracle Advanced Security` from the drop-down list at the top of the right window region.

   - Choose the `SSL` tab.

   - Choose the `Server` button.

   - Add the database name to the end of the wallet location.

   - Select `File—>Save the network configuration`; your `sqlnet.ora` file is updated.

   Default Wallet Locations for a database:

   - **Windows NT**:
     `C:\WINNT\Profiles\DATABASES\database_name`

   - **UNIX:** `/etc/ORACLE/WALLETS/DATABASES/database_name`

Step 2: Configure SSL Service Name

To configure the SSL service name:

1. In the Oracle Net Manager Service Naming window, select the Service Naming icon and choose the Create icon; this is necessary for SSL-authenticated enterprise users only.

2. In the Oracle Net Service Name Wizard window (Welcome), enter your chosen Net Service Name; this is the name you use to access your database as an enterprise user. Choose `Next`.

3. In the Oracle Net Service Name Wizard window (page 2 of 5: Protocol), select `TCP/IP with SSL`; choose `Next`.

4. In the Oracle Net Service Name Wizard window (page 3 of 5: Protocol Settings), enter your database Host Name and Port Number—that you will use for the SSL connection.

5. In the Oracle Net Service Name Wizard window (page 4 of 5: Service), enter the Oracle9*i* Service Name; choose `Next`.

6. Do not test this connection when asked. It will fail because (i) you have not set up the listener to listen for SSL connections, and (ii) you have not set up the database wallet; choose `Finish`.

7. In the Oracle Net Manager window, select `File—>Save the network configuration`; your `TNSNAMES.ORA` file is updated, and can be reviewed later.

Step 3: Configure the Listener

To configure the **listener**:

1. In the Oracle Net Manager window, expand `Listeners` and select the appropriate `Listener` (in the expandable tree menu on the left side of the window).

2. Select `Listening Locations` from the drop-down menu at the top right side of the window.

3. Choose the `Add Address` button at the bottom right side of the window.

4. Using the Protocol drop-down list, select SSL; enter your host name and your chosen SSL port number.

5. Select `File—>Save Network Configuration`; your `listener.ora` file is updated. Exit Oracle Net Manager.

Notes:

- Do not attempt to start the listener—until you have set up a wallet for SSL connections.

- Do not modify the value of `SSL_CLIENT_AUTHENTICATION` in `listener.ora`, which should be `FALSE` *(the listener is not doing the authentication—the database uses SSL to authenticate the client).*

---

6. For Windows NT only: you must manually edit `tnsnames.ora` for the client; edit `$ORACLE_HOME/network/admin/tnsnames.ora` by adding the following to your SSL service name:

```
SECURITY = (AUTHENTICATION_SERVICE = TCPS)
```

Step 4: Review the .ORA Files

To facilitate review of your .ora files, some Windows NT examples follow:

Example: The SQLNET.ORA File

```
NAMES.DEFAULT_DOMAIN = WORLD
WALLET_LOCATION =
    (SOURCE =
        (METHOD = FILE)
        (METHOD_DATA =
            (DIRECTORY = C:\WINNT\Profiles\DATABASES\oe)
        )
    )

SQLNET_AUTHENTICATION_SERVICES = (TCPS,NTS)
SSL_CLIENT_AUTHENTICATION = TRUE

SSL_VERSION = 0

SQLNET.CRYPTO_SEED = 4fhfquweotcadsfdsafjkdsfqp5f201p45mxskdlfdasf
```

> **Note:** The wallet location matches the one you entered in Oracle
> Net Manager for the database.

Example: The TNSNAMES.ORA File:

```
OESSL.WORLD =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCPS) (HOST = host1) (PORT = 5000)
        )
        (CONNECT_DATA =
            (SERVICE_NAME = finance)
        )
        (SECURITY = (AUTHENTICATION_SERVICE = TCPS)
            (SSL_SERVER_CERT_DN="cn=finance,cn=OracleContext,o=Oracle,c=us")
        )
    )

OE.WORLD =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = host1) (PORT = 1521)
        )
        (CONNECT_DATA =
            (SERVICE_NAME = oe.world)
        )
    )
```

Example: The LISTENER.ORA File:

```
WALLET_LOCATION =
    (SOURCE =
        (METHOD = FILE)
        (METHOD_DATA =
            (DIRECTORY = C:\WINNT\Profiles\DATABASES\oe)
        )
    )


LISTENER =
    (DESCRIPTION_LIST =
        (DESCRIPTION =
            (ADDRESS = (PROTOCOL = TCP) (host = HOST1) (port = 1521)
        )
```

```
            (DESCRIPTION =
                (ADDRESS = (PROTOCOL = TCPS) (HOST = host1) (PORT = 5000))
            )
        )

SID_LIST_LISTENER =
    (SID_LIST =
        (SID_DESC =
            (GLOBAL_DBNAME = oe.world)
            (ORACLE_HOME = D:\Oracle\Ora81)
            (SID_NAME = oe)
        )
    )


SSL_CLIENT_AUTHENTICATION = FALSE
```

Task 5: Create the Wallet and Start the Listener

To create and configure the database wallet:

- Step 1: Create a Database Wallet

- Step 2: Enable Autologin

- Step 3: Start the Listener

- Step 4: Perform Database Logout for Security (optional)

Step 1: Create a Database Wallet

To create a database wallet:

1. Create a directory for the wallet for the location you specified in Step 1 on page 15-40.

2. Run Oracle Wallet Manager to create a new wallet for the database:

   - **Windows NT**: Select
     `Start—>Programs—>Oracle-OracleHome81—>Network`
     `Administration—>Wallet Manager`

   - **UNIX:** Enter `owm` at the command line.

3. Select `New` from the wallet menu. Do not create a new default directory when asked—this is for user wallets. During certificate request creation, type the **distinguished name (DN)** of the database *exactly*:

   `cn=database_name,cn=OracleContext,<location of Oraclecontext>`
   It is found in the initialization parameter file, in the parameter

   `RDBMS_SERVER_DN.`

   Note: The Distinguished Name is case-sensitive.

   For example:

   If the global database name chosen during installation is `sales.us.nmt.com`, and the location selected within NetCA for the Oracle Context is `o=nmt`, the complete DN of the database that you enter into Oracle Wallet Manager is:

   `cn=sales,cn=OracleContext,o=nmt`

   Note: `cn=OracleContext` must be included in the DN immediately after the simple database name.
   ---

4. Send the certificate request to your **certificate authority** (CA).

5. Add the CA *trusted certificate* to the database wallet. The CA trusted certificate is sometimes called a **root key certificate**.

6. Add database certificate to the database wallet.

Step 2: Enable Autologin

For users to access the database using SSL two-way mutual authentication with an LDAP compliant directory server, Autologin must be enabled for the database wallet, and the listener must be running. To enable database Autologin and run the listener:

1. Using Oracle Wallet Manager, select the Autologin check box under the Wallet menu to enable Autologin and to be able to start the listener on the database.

2. Save the wallet to the directory you set up when you completed Step 1 on page 15-40. For verification, check that there is a cwallet.sso file in the wallet directory.

3. Stop the listener. The listener must read the database's open wallet, so the database must log on before the listener can be started.

   To stop the listener, enter the following at the command line:

   - **Windows NT**: `lsnrctl stop`

   - **UNIX:** `lsnrctl stop`

     Note for NT: You can alternatively stop Oracle Listener Services in the Services Control Panel. See: Notes on page 15-48.

     Notes:

     - Only user wallets can be stored in the directory; database wallets cannot be stored in the directory.

     - End users never have to use Oracle Wallet Manager, because Oracle Enterprise Login Assistant can be used to enable and disable Autologin.

     ---

4. **Change Oracle Services Login (Windows NT only).** Because the database and the listener services are running as system (with few privileges in NT), and the wallets are opened under your user name, the database and the listener are not able to read the wallet. In order for them to read their wallet, they must be changed to log on as the user who enabled Autologin for the database wallet.

   To change the Oracle Services login:

   - Shut down the database by opening the Services control panel and selecting `OracleService <database name>;` choose the `Stop` button; choose `Yes` to confirm.

   - Choose the `Startup` button.

   - In the `Log On As` region of the Service Window (Figure 15–7):

Figure 15–7   The Oracle Service Window
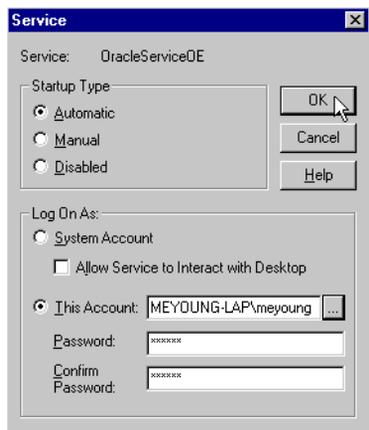
Choose This Account and enter <domain>\< NT user login> for the user who enabled Autologin for the database wallet; alternatively, you can choose the browse button (...) to select from a list; enter your password in the Password and Confirm Password fields; choose OK.

- Choose Start to start up the Oracle database.

- Repeat these steps (starting with *Choose the Startup Button*) for Oracle-<ORACLE_HOME_NAME>->TNSListener; do *not* start the listener service.

- Close the Services window.

Step 3: Start the Listener

To start the listener, enter the following at the command line:

- **Windows NT**: lsnrctl start

- **UNIX:** lsnrctl start

Notes:

- If the Listener starts correctly, it confirms that it is listening on TCPS, on the command line.

- If there are errors when you attempt to start the Listener, you may not have selected Autologin, or the wallet may be in the wrong location.

- Under Windows NT, you can use the OracleListener Service under the services control panel to start and stop the listener—but without the command line response that confirms listener activity.

The database wallet is now open, and the database is able to participate in authenticated communications using SSL; on Windows NT, the `OracleTNSListener` service is also started.

> **See Also:** Chapter 16, Using Oracle Wallet Manager, for detailed instructions about creating a wallet

Step 4: Perform Database Logout for Security (optional)

Important: If the database is to be shut down for an extended period of time, disable use of the database wallet for security purposes.

To logout from the database:

1. Stop the listener.

2. Using Oracle Wallet Manager, disable Autologin by clearing the Autologin check box.

3. Restart the listener.

Task 6: Verify Database Installation

To verify that the database has been successfully configured:

1. Verify that there is a `cwallet.sso` file located in the database wallet directory. If not, Autologin was not successfully enabled. If this happens, go back to the

Oracle Wallet Manager, open the wallet, select the Autologin check box, and save the wallet.

2. Verify that there is an `ldap.ora` file located in

   `$ORACLE_HOME/network/admin`

   If there is no `ldap.ora` file, NetCA failed to configure directory access. Verify that the `ORACLE_HOME` is set and rerun NetCA.

3. Use the directory administration tool to verify that a database entry and subtree exist under the Oracle Context you specified when you ran NetCA. If you do not find the database entry, verify that the directory is running, the Oracle Context is set up, and the `ldap.ora` file exists and is correct. Then register the database again, using DBCA.

Task 7: Create Global Schemas and Roles

Although this task can be completed using Oracle Enterprise Manager, the following examples use SQL*Plus directly.

To create global schemas and roles:

- Step 1: Create a Global Schema
- Step 2: Grant a Create Session Privilege
- Step 3: Create Global Roles
- Step 4: Associate Privileges

Step 1: Create a Global Schema

Using SQL*Plus, create a shared schema (called Guest, for example) for enterprise users by entering:

```
CREATE USER guest IDENTIFIED GLOBALLY AS ''
```

Note the two single quotation marks with no space between them at the end of the line. If you enter a specific **distinguished name (DN)** between the quotation marks, only that user is able to connect to that schema, and it is not shared.

Step 2: Grant a Create Session Privilege

Users connecting to this schema require a `CREATE SESSION` privilege. You can grant the `CREATE SESSION` privilege either to the global schema, or to a **global role** which you grant to specific users though an enterprise role.

Step 3: Create Global Roles

Create global roles for the database to hold relevant privileges. These roles are associated with enterprise roles to be created later. Enterprise roles are allocated to users.

For example:

```
CREATE ROLE emprole IDENTIFIED GLOBALLY;
CREATE ROLE custrole IDENTIFIED GLOBALLY;
```

Step 4: Associate Privileges

Associate privileges with the new global roles.

For example:

```
GRANT select ON products TO custrole, emprole;
```

> **Note:** Oracle Advanced Security can be configured to authenticate enterprise users using either SSL or password authentication.
>
> - To configure SSL authentication, continue with Part III: Final Configuration for SSL Authentication, which follows.
>
> - To configure password authentication, go to Part IV: Final Configuration for Password Authentication on page 15-59.

Part III: Final Configuration for SSL Authentication

This section describes the final steps to complete the installation and configuration of Enterprise User Security for SSL authentication. The required tasks (numbered in sequence from Part II) follow:

- Task 8: Configure Database Clients
- Task 9: Configure an Enterprise Domain
- Task 10: Configure Enterprise Users
- Task 11: Log In as an Enterprise User

Note: The configuration tasks for Enterprise User Security are numbered consecutively, and continue in sequence from Part II. This part includes Task 8 through Task 11.

- For Tasks 1 through 7, See: Part II: Initial Configuration for SSL and Password Authentication on page 15-31

- For Tasks 12 through 16, See: Part IV: Final Configuration for Password Authentication on page 15-59.

Task 8: Configure Database Clients

Once you have installed Oracle9*i* clients, configure Oracle Net on the clients by using Oracle Net Manager. You may complete this step during or after installation of Oracle9*i* Release 9.0.1.

Because you will be using an LDAP directory service for enterprise security, you may also want to use Oracle Net directory naming. Oracle Net directory naming lets the client connect to the database using the database entry registered with the directory by Oracle Database Configuration Assistant. Alternatively, you can use one of the other Oracle Net naming methods, such as *local naming* (`tnsnames.ora` file), to configure a net service name for the database.

To configure database clients:

1. Use Oracle Net Manager to configure SSL on UNIX; configure an SSL net service name, as described by Step 2 on page 15-40.

2. Configure the client profile. Do not enter a wallet location when configuring a client profile. The lack of a specific wallet location indicates that SSL should find the default wallet for the current operating system user. In this way, the `sqlnet.ora` file can be shared by enterprise users, providing easier administration and deployment. Each user whose wallet is in a non-default wallet location must have a separate `sqlnet.ora` file that contains that user's wallet location.

> **Note:** If you do not install clients, and `ORACLE_HOME` is set to a database server `ORACLE_HOME`, and that `ORACLE_HOME` has a `sqlnet.ora` file with a wallet location, you must create at least one new `TNS_ADMIN` directory with a `sqlnet.ora` file—with *no* wallet location. This ensures that SSL uses the default location of the wallet for the operating system user.

Default Wallet Directories for the User Wallets:

- **Windows NT:**

  `c:\winnt\profiles\<user>\ORACLE\WALLETS`

- **UNIX:**

  `/etc/ORACLE/WALLETS/<OS username>`

> **Note:** Wallets for specific users are set up when you create enterprise users. See Chapter 18, Using Oracle Enterprise Security Manager, for instructions about creating enterprise users.

See Also:

- *Oracle Net Services Administrator's Guide*
- Chapter 7, Configuring Secure Sockets Layer Authentication, for information about configuring SSL

Task 9: Configure an Enterprise Domain

Oracle Enterprise Security Manager is installed automatically as part of the Oracle9*i* installation, and is used to configure an enterprise domain. Note that the Oracle default domain is created by default when the Oracle Context is created in the directory, and databases are automatically added as members of that domain when they are registered by DBCA. Table 15–3 lists the steps required to set up an enterprise domain, and cross-references related instructions. If you are using the Oracle default domain, you can skip steps 1 and 6.

Table 15–3   Setting up an Enterprise Domain

Step	Related Instructions
1. Create an **enterprise domain**.	Administering Enterprise Domains on page 18-30.
2. (Optional) Add domain administrators for the domain.	Chapter 18, Using Oracle Enterprise Security Manager.
3. Enable or disable enterprise user current user database links between member databases.	Administering Enterprise Domains on page 18-30.
4. Select authentication type for the domain. Must be (i) SSL only, or (ii) both password and SSL.	
5. Configure user-schema mappings for the domain; alternatively, you can configure database-specific user-schema mappings.	**Chapter 18, Using Oracle Enterprise Security Manager.**
6. Create enterprise roles in the domain.	Administering Enterprise Roles on page 18-36.

Table 15–3   Setting up an Enterprise Domain

Step	Related Instructions
7. Enroll the database as a member of the desired enterprise domain.	Defining Database Membership of an Enterprise Domain on page 18-32
8. Create global roles on the databases. The SQL*Plus command is: `CREATE ROLE rolename IDENTIFIED GLOBALLY`	■ *Oracle9i SQL Reference* ■ See Also: Step 3: Create Global Roles on page 15-50
9. Assign global roles to each enterprise role.	Administering Enterprise Roles on page 18-36

Task 10: Configure Enterprise Users

To create a new enterprise user:

- Step 1: Add a New Enterprise User to the Directory
- Step 2: Create a User Wallet
- Step 3: Authorize the User
- Step 4: Map the User to a Schema

Step 1: Add a New Enterprise User to the Directory

Any directory user can be an enterprise user. You can add users to the directory by using one of the following tools:

- Oracle Enterprise Security Manager
- The administration tool for your directory service
- The standard LDAP command line tools

If you elect to populate the directory with users before using Oracle Enterprise Security Manager, note that user entries in the directory must have the `orcluser objectclass`.

If Oracle Enterprise Security Manager is used to prepare existing user entries for Oracle use (provision), the `orcluser objectclass` is added to the existing entry.

See Also:

- Creating New Enterprise Users on page 18-7 for instructions about adding new enterprise users to the directory by using Oracle Enterprise Security Manager

- Documentation for your directory service for information about using the directory administration tools

Step 2: Create a User Wallet

To create a user wallet, See: Chapter 16, Using Oracle Wallet Manager.

> **Note:** Store the user wallet in the default user wallet location, or in the directory (if it is stored only in the directory, it must be downloaded to the client before use):
>
> - **Windows NT**: `x:\winnt\profiles\<os user>\ORACLE\WALLETS`
>
> - **UNIX:** `/etc/ORACLE/WALLETS/<os user>`

Step 3: Authorize the User

You can do either or both of the following:

- Local Oracle role authorization:

  Use Oracle Enterprise Manager or SQL*Plus to grant local roles and privileges to the database user or schema; *this step is optional.*

  > **Note:** If the schemas are **shared schemas**, all roles granted to the shared schema are enabled for all users connecting to that schema. Accordingly, *you should avoid granting any local roles or privileges to a shared schema.*

- Enterprise role authorization:

  Use Oracle Enterprise Security Manager to grant enterprise roles to the enterprise user in the directory.

Step 4: Map the User to a Schema

If you are using a shared schema, use Oracle Enterprise Security Manager to map the user to a schema. You can choose either of the following mapping options:

- Database:

  Applies to one database.

- Domain:

  Applies to all databases in the domain.

For example:

If you are creating a domain mapping from three users to a shared schema called guest, and you have more than one database in the domain, each database must have a shared schema (called guest) that all three users can access. These three users cannot connect to any database in the domain that does not have a shared schema called guest.

Alternatively, you can create a mapping under a particular database. If you do it this way, the mapping applies only to that database, and not to all databases in the domain. If you have mappings in both places, the database mapping takes precedence.

> **See Also:**
> - Task 9: Configure an Enterprise Domain, for information about setting up an enterprise domain
> - Administering Enterprise Users on page 18-7
> - Mapping an Enterprise User to a Shared Schema on page 15-23

Task 11: Log In as an Enterprise User

To log in as an Enterprise User:

- Step 1: Download the User Wallet
- Step 2: Enable Autologin
- Step 3: Connect to the Database

Step 1: Download the User Wallet

To download a user wallet:

1. Log in to the operating system as the appropriate user.
2. Start Oracle Enterprise Login Assistant.
3. Download the wallet.

Step 2: Enable Autologin

The enterprise user must enable Autologin for the user wallet (created in Task 10) in order to log in to the database. Enabling Autologin generates a single sign-on file and enables authentication to the SSL adapter.

To enable Autologin, use Oracle Enterprise Login Assistant to select the Autologin box.

> **See Also:** Chapter 17, Using Oracle Enterprise Login Assistant, for the following:
>
> - To download a user wallet.
> - To enable Autologin.

Step 3: Connect to the Database

1. Set `ORACLE_HOME`.

   If the `ORACLE_HOME` is set to a server `ORACLE_HOME`, you must set the `TNS_ADMIN` environment variable to address the directory where you placed the `sqlnet.ora` file—that you created in Task 8: Configure Database Clients on page 15-52.

   If you have a separate client `ORACLE_HOME`, you do not need to set the `TNS_ADMIN` environment variable.

2. Launch SQL*Plus and enter:

```
sqlplus/@connect_identifier
```

where `connect_identifier` is the net service name you set up in Task 8: Configure Database Clients on page 15-52.

If you are successful, the system responds `Connected to:`...; this is the principal confirmation of a successful connection and setup. If an error message is displayed, see: Part IV: Final Configuration for Password Authentication on page 15-59.

If you do connect successfully, check that the appropriate global roles were retrieved from the directory by entering:

```
select * from session_roles
```

In the Oracle Enterprise Login Assistant, select `Autologin > Logout` to disable authentication with the SSL adapter.

> **See Also:** Chapter 17, Using Oracle Enterprise Login Assistant, for instructions about using Oracle Enterprise Login Assistant

> **Note:** You have competed the configuration of Enterprise User Security for SSL authentication; *do not proceed to Section IV, which describes the configuration of password authentication only.*

Part IV: Final Configuration for Password Authentication

This section describes how to set up password authentication for enterprise users. The required tasks (numbered in sequence from Part III) follow:

- Task 12: Complete Initial Setup Steps
- Task 13: Configure the Enterprise Domain
- Task 14: Configure Oracle Context
- Task 15: Configure Enterprise Users
- Task 16: Connect as Password Authenticated Enterprise User

Note: The configuration tasks for Enterprise User Security are numbered consecutively, and continue in sequence from Part III. This part includes Task 12 through Task 16.

- For Tasks 1 through 7, See: Part II: Initial Configuration for SSL and Password Authentication on page 15-31

- For Tasks 8 through 11, See: Part III: Final Configuration for SSL Authentication on page 15-51.

- *Note further that the configuration tasks described by Part III and Part IV are mutually exclusive.* To configure SSL authentication for enterprise users, complete the tasks in *Part II and Part III only.* To configure password authentication for enterprise users, complete the tasks in *Part II and Part IV only.*

Task 12: Complete Initial Setup Steps

In this task you complete the initial Enterprise User Security setup steps, including the following:

- Install or identify a **certificate** service.

- Install and configure a directory service.

- Install and configure the database.

- Configure the database for SSL.

- Create and configure the database wallet.

- Create global schemas and roles.

> **See Also:** The first six tasks described under *Part II: Initial Configuration for SSL and Password Authentication on page 15-31*

Task 13: Configure the Enterprise Domain

Configure the enterprise domain for password authentication.

Oracle Enterprise Security Manager launches from the Oracle Enterprise Manager console. Oracle Advanced Security uses it for managing enterprise domains, users, roles, and databases. It can be used to configure an enterprise domain. Note that the Oracle default domain is created by default when the Oracle Context is created in the directory, and databases are automatically added as members of that domain when they are registered by DBCA (Figure 15–8).
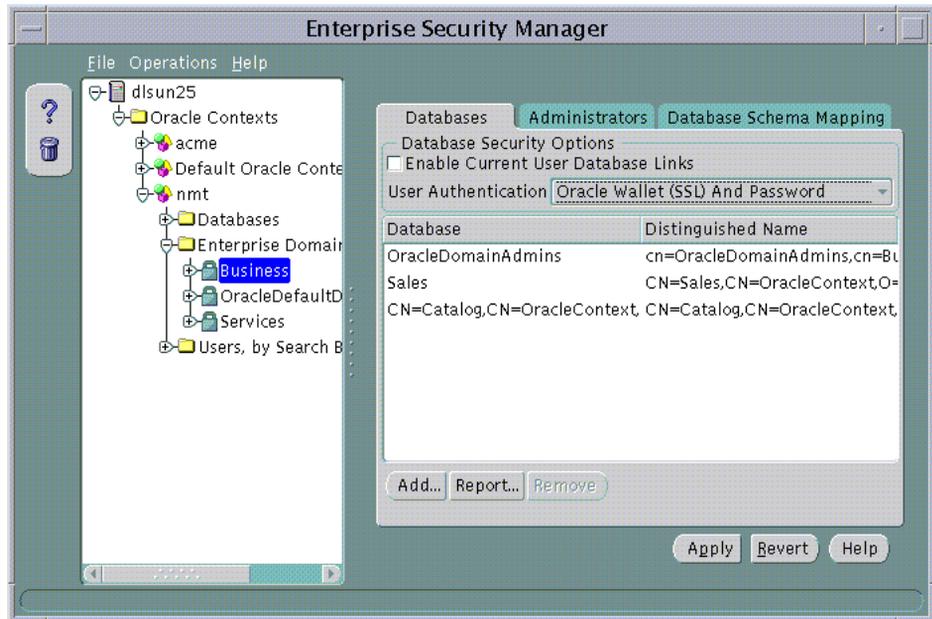
Figure 15–8   Enterprise Security Manager: Oracle Domain Properties Window

Table 15–4 lists the steps required to set up an enterprise domain, and cross-references related instructions. If you are using the Oracle default domain, you can skip step 1 and step 4.

Table 15–4   Setting up an Enterprise Domain

Step	Related Instructions
1. Create an enterprise domain.	Administering Enterprise Domains on page 18-30.
2. (Optional) Add domain administrators for the domain.	Chapter 18, Using Oracle Enterprise Security Manager.
3. Enable or disable current user database links between member databases.	Administering Enterprise Domains on page 18-30.
4. Configure user schema mappings for the domain or database.	User-Schema Mappings on page 15-8.
5. Create enterprise roles in the domain.	Administering Enterprise Roles on page 18-36.
6. Use Oracle Enterprise Security Manager to make the database a member of the desired enterprise domain.	Defining Database Membership of an Enterprise Domain on page 18-32.
7. Create global roles on the databases. The SQL*Plus command is: `CREATE ROLE rolename IDENTIFIED GLOBALLY`	■ *Oracle9i SQL Reference* ■ See Also: Step 3: Create Global Roles on page 15-50
8. Assign global role(s) to each enterprise role.	Administering Enterprise Roles on page 18-36.
9. Choose the Enterprise Domain Administration tab; select *Oracle Wallet (SSL) And Password, or Password Only* from the Enterprise User Authentication drop-down menu.	See: Figure 15–8.

Task 14: Configure Oracle Context

- Step 1: Configure User Search Bases

- Step 2: Configure UserID Attribute

- Step 3: Configure Administrators

- Step 4: Configure Password-Accessible Domains

Figure 15–9   Oracle Context Properties Window

Note: The reference to *Default Oracle Context* in Figure 15–9 should read *Root Oracle Context*; all references to *Default Oracle Context* will be changed to *Root Oracle Context* in the production release of Oracle Advanced Security.

Step 1: Configure User Search Bases

Choose the General tab of the Oracle Context Properties Window (Figure 15–9). In the Common User Search Bases region, enter the user search bases under which the

databases are to search for user entries. A user search base is the root of a subtree under which you have stored your enterprise user entries in the directory.

Step 2: Configure UserID Attribute

Choose the General tab (Figure 15–9). In the Context Attribute Settings region, enter the name of the user entry attribute that holds the `UserID`—that uniquely identifies each enterprise user. The default `UserID` attribute is the `common name` (`cn`) attribute defined in the LDAP directory, but can be changed by the Security Administrator.

Example:

- Assume that all enterprise users in your organization can be uniquely identified by their `employeeid`.

- If `employeeid` values are stored in an attribute called `eid`, you enter `eid` in the `UserID` field.

Step 3: Configure Administrators

Choose the Administrators tab (Figure 15–9). Set up all necessary administrators for this Oracle Context, if you haven't already done so.

- A Context Administrator has full privileges for the Oracle Context.

- A Database Security Administrator can create and delete enterprise domains and roles, and assign databases to domains.

- A User Security Administrator manages security for the user entries in the directory, by setting passwords, viewing password hints, and other similar actions.

Step 4: Configure Password-Accessible Domains

In order to accept password-authenticated connections, a database must belong to a domain in the Password Accessible Domains group—and the database access permissions on the user search base must be enabled. This enables the database to read the user's login credentials in the directory.

In a selected Oracle9*i* Oracle Context, add the domain to the Password-Accessible Domains group. Choose Add and select one of the current enterprise domains from the resulting dialog. To remove an enterprise domain from the group, select it in the Accessible Domains window and choose Remove.

See Also:

- Step 5: Enable Database Access on page 15-69
- Security of User Database Login Information on page 15-11

Task 15: Configure Enterprise Users

- Step 1: Create Enterprise Users
- Step 2: Authorize Users
- Step 3: Create Enterprise User Ids
- Step 4: Create Enterprise User Passwords
- Step 5: Enable Database Access

Step 1: Create Enterprise Users

Any directory user can be an enterprise user. You can add users to the directory by using one of the following tools:

- Oracle Enterprise Security Manager (Figure 15–10)
- The administration tool for your directory service
- The standard LDAP command line tools

Figure 15–10   Enterprise User Security: Create User Window

If you elect to populate the directory with users before using Oracle Enterprise Security Manager, note that user entries must have the `orcluser objectclass`.

> **See Also:**
>
> - Administering Enterprise Users on page 18-7 for instructions about adding new enterprise users to the directory by using Oracle Enterprise Security Manager
>
> - Documentation for your directory service for information about using the directory administration tools

Step 2: Authorize Users

You can do any of the following:

- Grant local Oracle roles.

  Use Oracle Enterprise Manager or SQL*Plus to grant local roles and privileges to the database user or schema; this step is optional.

- Grant enterprise roles.

Use Oracle Enterprise Security Manager to grant enterprise roles to the enterprise user in the directory. User authorizations are the aggregate of both local database roles and enterprise roles.

- Map the users to a schema.

If you are using a shared schema, use Oracle Enterprise Security Manager to map the user to a schema. You can choose either of the following mapping options:

- Database:

  Applies to one database.

- Domain:

  Applies to all databases in the domain.

  For example:

If you are creating a domain mapping from three users to a shared schema called guest, and you have more than one database in the domain, each database must have a shared schema (called guest) that all three users can access. These three users cannot connect to any database in the domain that does not have a shared schema called guest.

Alternatively, you can create a mapping under a particular database. If you do it this way, the mapping applies only to that database, and not to all databases in the domain. If you have mappings in both places, the database mapping takes precedence.

> **See Also:**
>
> - Task 9: Configure an Enterprise Domain on page 15-53 for information about setting up an enterprise domain
>
> - Administering Enterprise Users on page 18-7
>
> - Mapping an Enterprise User to a Shared Schema on page 15-23

Step 3: Create Enterprise User Ids

For each user, define a UserID that is unique across the enterprise (Figure 15–10). The default UserID is the value in the common name (cn) attribute defined in the LDAP directory.

Choose a UserID that conforms to the following:

- It is *unique across the directory,* or at least across the subtrees under the user search bases entered under Step 1. For example, if you choose cn to be the

UserID attribute, and Scott's `cn` attribute value is `Scott.us`, there can be no other `cn=Scott.us` defined in any of the users specified in the user search base field.

- *It is short and easy to enter.* A UserID is intended to be an easy-to-use abbreviation of the **distinguished name (DN)**.

For example, in Figure 15–10 the `UserID` is `hscortea`.

> **See Also:**
> - Figure 15–2, Enterprise User Security Elements (SSL-Authentication)
> - Figure 15–9, Oracle Context Properties Window

Step 4: Create Enterprise User Passwords

Choose the Password tab of the Create User Window (Figure 15–10) to create a password for each enterprise user. Oracle Enterprise Security Manager automatically creates the associated password verifiers and stores them in the `orclPassword` attribute of the user entry (Figure 15–11).
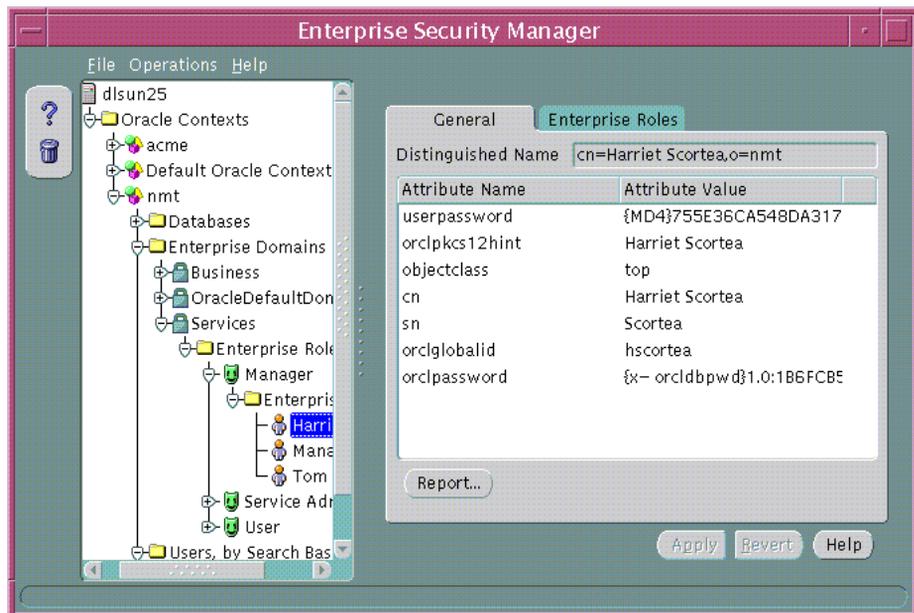
> **Note:** You can change use Oracle Enterprise Login Assistant to change passwords at any time.

> **See Also:** Defining a New Enterprise User Password on page 18-10

Figure 15–11   Enterprise Security Manager: User Attributes Window

Step 5: Enable Database Access

The user entry must reside in a directory subtree of users that has been enabled for Oracle database access. You can set Oracle Database Access permissions for a selected subtree—to let databases within a domain in the Password-Accessible Domains group read the user's login credentials.

To enable database access:

On a selected subtree of directory users, set Oracle Database Access permissions to permit databases in the Password-Accessible Domains group to access the user's database login credentials:

- Select the target user subtree under Users, by Search Base.

- Select Database Access Restriction for that subtree.

> **See Also:** Security of User Database Login Information on
> page 15-11

Task 16: Connect as Password Authenticated Enterprise User

For an enterprise user whose `UserID` is `hscortea` (Figure 15–11) and whose
password is `welcome`, enter the following to connect using `sqlplus`:

```
SQL>connect hscortea/welcome@<TNS Service Name>
```

The database authenticates the enterprise user (`hscortea`) by verifying the
username/password combination against the directory entry associated with this
user. If successful, the connection to the database is established.

> **Note:** You have completed the configuration of Enterprise User
> Security for password authentication.

Part V: TroubleShooting Enterprise User Login

This section describes potential problems and associated corrective actions.

No Global Roles

The following tips help you verify that the user has been allocated the correct global roles upon database login and, if necessary, help determine the cause of failure:

1. Check for the existence of global roles. Enter the following, including the semi-colon (;):

    ```
    SELECT * FROM session_roles;
    ```

2. If there are no roles, one of the following applies:

    ■ The roles were not allocated to an enterprise role in Oracle Enterprise Security Manager.

    ■ The enterprise role was not assigned to the user in Oracle Enterprise Security Manager.

    ■ The database and Oracle Enterprise Security Manager have different values for the database domain; shut down and restart the database to update the database internal value.

    ■ Your database does not have proper permissions in the directory to see the roles. These permissions are created automatically, so it is possible that the **distinguished name (DN)** in the database certificate does not match the distinguished name registered for the database. In this case, the directory does not recognize the database as the proper entity, and denies access.

    Do an LDAP search to display the appropriate roles by entering the following:

    ```
    ldapsearch –h <directory hostname> P <SSL directory port number> –U 3
    –W "file:<walletpath>" –P <database wallet password>
    –b "cn=oracleDBSecurity, cn=Products, cn=OracleContext, <admin context>"
    "objectclass=orclDBenterpriseRole"
    ```

    If you do not see the roles, the database is not in the correct domain—or there is an incorrect distinguished name (DN) in the database wallet certificate. If the database appears to be receiving information from the wrong domain, try restarting the database to update its internal domain membership information.

TNS Lost Connection

This error may indicate that you attempted to configure a domestic cipher suite. Run Oracle Net Manager again, and be sure that you choose the Show Domestic Cipher Suites button.

ORA-1004: Default username feature not supported

This error indicates that the connection was not over SSL. Look at the `tnsnames.ora` file to verify the protocol value of the net service name that you are using. The value must be `TCPS` and not `TCP`.

ORA-1017: Invalid username/password

The distinguished name that the wallet uses to connect does not match the DN in the `CREATE USER` statement for any schema in the database, and it does not match the DN in any relevant mapping.

1. Check the DN of the user in the mapping created using Oracle Enterprise Security Manager.

2. Also check that your directory is actually listening properly for incoming SSL connections. From a command prompt, enter:

```
ldapbind -h <directory hostname> -p <directory SSL port number> -U 3 -W
"file:[database wallet path]" -P [database wallet password]
```

`Bind successful` should be displayed. If the bind fails, try restarting the SSL instance of your directory.

Then try the bind again.

If it still doesn't work, carefully check the wallet location in the configuration set via Oracle Directory Manager. Make sure that it is set to the proper path name.

> **Important:** You *must* get this `ldapbind` to work. If it does not work, *do not continue.*

3. If the prior steps do not work, circumvent the user-schema mapping step by altering the user `guest` (for example) to be a non-shared schema.

In `sqlplus` as `system/manager@database_name`, enter:

```
alter user guest identified globally as <user DN>;
```

and then try the `connect /@connect_identifier` again. If this succeeds, the problem is associated with the mapping of the user to the schema; use Oracle Enterprise Security Manager to check that mapping in the directory.

If this still fails, the user wallet DN does not match the DN you specified in the `alter user` statement. Check the user wallet.

Alter this user back to a shared schema by entering:

```
alter user guest identified globally as '';
```

ORA-12560: Protocol adapter error

This error usually means that something is wrong with the wallet. Look in the `sqlnet.log` file in the current operating system directory for more information. Also, on Windows NT, this can mean that the Oracle service has stopped; check the Services control panel.

Decryption of Encrypted Private Key Fails

Applies to Window NT only.

This error occurs when you attempt to open a wallet that you are not permitted to open.

For Example:

- You are logged into the system as `user-x`, but you do not have a local `sqlnet.ora` file that identifies `c:\winnt\profiles\user-x\oracle\wallets` as your wallet location.

- SSL uses the `sqlnet.ora` file in the default location to find the wallet location, and then tries to open the database wallet to get your login credentials.

- This attempt *fails*, because `user-x` does not have permission to open the database wallet.

ORA-28030

This is a catch-all Oracle9*i* error that indicates something unanticipated went wrong with the RDBMS to directory LDAP query. It is possible that SSL has failed on the directory—the database and directory wallets may not share a **trusted certificate**. Try to bind to the directory over SSL using the database wallet.

> **See Also:** Error ORA-1017: Invalid username/password on page 15-73

Tracing

You can use tracing to help debug. This is appropriate if the `ldapbind` (See: **ORA-1017: Invalid username/password**) fails, indicating that the directory's SSL instance is not working properly.

Oracle Internet Directory

If you are using Oracle Internet Directory as your ldap directory, use the following tracing procedure:

1. Turn on debugging flags in Oracle Internet Directory.

   > **Note:** See Also: *Oracle Internet Directory Administrator's Guide*

2. Start up the SSL Oracle Internet Directory instance in full debug mode. Log files will be written to `$ORACLE_HOME\ldap\log`. Look at the file with your SSL directory instance number and an `s` in its filename. The log files without the `s` are for the monitor process (`oidmon`) and the dispatcher. Look at the end of the log file immediately after you have tried your `connect /@connect_ identifier`. One thing to look for is the string `Distinguished Name` to ensure that it matches the DN of your user.

3. Turn off Oracle Internet Directory tracing.

16

Using Oracle Wallet Manager

Security administrators use Oracle Wallet Manager to manage public-key security credentials on Oracle clients and servers. The wallets it creates are opened by using either Oracle Enterprise Login Assistant or Oracle Wallet Manager.

This chapter describes Oracle Wallet Manager, and contains the following topics:

- Overview
- PKCS #12 Support
- Multiple Certificate Support
- LDAP Directory Support
- Managing Wallets
- Managing Certificates

> **See Also:** Chapter 17, Using Oracle Enterprise Login Assistant, for information about how to open and close wallets for secure SSL communications using Oracle Enterprise Login Assistant

Overview

Traditional private-key or symmetric-key cryptography requires that entities desiring to establish secure communications possess a single secret key known only to them. *Harriet* and *Dick*, for example, could agree to shift each letter in their private messages by two character positions (A becomes C, B becomes E, and so on) to encrypt the message text. Using this method, a *HELLO* message from Harriet to Dick would read *JGNNP.* The actual encryption methods in current use are much more complex and significantly more secure, but an underlying problem remains—sending messages encrypted with a single key requires prior, *secure* distribution of the key to each participating party. Otherwise, a malicious third party might obtain the key, intercept communications, and compromise security. Public-key cryptography addresses this problem, by providing a secure method for key distribution.

Public-key cryptography requires a party to possess a **public/private key pair**. The **private key** is kept secret and is known only to that party. The **public key**, as the name implies, is freely available. To send a secret message to this party requires that a third party sender encrypt the message with the public key. Such a message can only be decrypted by a party holding the associated private key.

For example, when Dick wants to send a secure message to Harriet, he first asks Harriet for her public key (or obtains it from another, public source). Harriet gives Dick the public key, but Tom, a malicious eavesdropper, also obtains the public key. Nevertheless, when Dick sends Harriet a message encrypted with her public key, Tom cannot decrypt it; the message can only be decrypted with Harriet's private key.

Public-key algorithms thus guarantee the secrecy of a message, but they don't guarantee *secure communications* because they don't verify the identities of the communicating parties. In order to establish secure communications, it is important to verify that the public key used to encrypt a message does in fact belong to the target recipient. Otherwise, a third party can potentially eavesdrop on the communication and intercept public key requests, substituting its public key for a legitimate key.

If Tom, for example, is able to substitute his public key for Harriet's public key and send it to Dick, Dick might then send a message to Harriet encrypted with Tom's public key—believing he was using Harriet's public key. Tom could then decrypt a subsequent intercepted message from Dick using his private key, re-encrypt it with Harriet's public key and re-transmit it to Harriet. Harriet could then decrypt the incoming message using her private key, and never know that it had been intercepted by Tom—the **man-in-the-middle**.

In order to avoid such a man-in-the-middle attack, it is necessary to verify the owner of the public key, a process called **authentication**. Authentication can be accomplished through a **certificate authority** (CA).

A CA is a third party that is trusted by both of the parties attempting secure communication. The CA issues public key certificates that contain an entity's name, public key, and certain other security credentials. Such credentials typically include the CA name, the CA signature, and the certificate effective dates (From Date, To Date).

The CA uses its private key to encrypt a message, while the public key is used to decrypt it, thus verifying that the message was encrypted by the CA. The CA public key is well known, and does not have to be authenticated each time it is accessed. Such CA public keys are stored in an Oracle **wallet**.

Wallet Password Management

Oracle Wallet Manager includes an enhanced wallet password management module that enforces Password Management Policy guidelines, including the following:

- Minimum password length (8 characters)

- Maximum password length unlimited

- Alphanumeric character mix required

Strong Wallet Encryption

Oracle Wallet Manager stores private keys associated with X.509 certificates, requiring strong encryption. Accordingly, Release 9.0.1 replaces DES encryption with 3-key Triple-DES—a substantially stronger encryption algorithm.

Microsoft Windows Registry

Oracle Wallet Manager lets you optionally store multiple Oracle wallets in the user profile area of the Microsoft Windows System Registry (for Windows 95/98/ME/NT 4.0/2000), or in a Windows file management system. Storing your wallets in the registry provides the following benefits:

- **Better Access Control.** Wallets stored in the user profile area of the registry are only accessible by the associated user. User access controls for the system thus become, by extension, access controls for the wallets. In addition, when a user logs out of a system, access to that user's wallets is effectively precluded.

- **Easier Administration.** Since wallets are associated with specific user profiles, no permissions need to be managed, and the wallets stored in the profile are

automatically deleted when the user profile is deleted. Oracle Wallet Manager can be used to create and manage the wallets in the registry, and the wallets are accessible by Oracle Enterprise Login Assistant as well.

- **Improved Security.** Because the wallets are imbedded in the registry, the wallets associated with a particular user profile are transparent to all other users. Viewed in combination with *better access control* and *easier administration,* this amounts to an additional security layer for Oracle wallets.

Options Supported:

- Open wallet from the Registry
- Save wallet to the Registry
- Save As to a different Registry location
- Delete wallet from the Registry
- Open wallet from the file system and save it to the Registry
- Open wallet from the Registry and save it to the file system

> **See Also:**
> - *Oracle9i Database Administrator's Guide for Windows*
> - *Oracle9i Network, Directory, and Security Guide for Windows*

Oracle Wallet Functions

Oracle Wallet Manager is a stand-alone Java application that wallet owners use to manage and edit the security credentials in their Oracle wallets. These tasks include the following:

- Generating a public/private key pair and creating a certificate request for submission to a CA.
- Installing a certificate for the entity.
- Configuring **trusted certificate**s for the entity.
- Opening a wallet to enable access to PKI-based services.
- Creating a wallet that can be accessed by using either Oracle Enterprise Login Assistant or Oracle Wallet Manager.
- Uploading a wallet to an LDAP directory.
- Downloading a wallet from an LDAP directory.

- Importing wallets.

- Exporting wallets.

Backward Compatibility

Oracle Wallet Manager is backward-compatible to Release 8.1.5.

PKCS #12 Support

Oracle Wallet Manager stores X.509 certificates and **private key**s in industry-standard, PKCS #12 format. This makes the Oracle wallet structure interoperable with supported third party PKI applications, and provides wallet portability across operating systems.

> **Note:** Although Oracle Advanced Security and Oracle Wallet Manager fully comply with PKCS #12, there may be some compatibility issues using third-party products—such as Netscape Communicator and Microsoft Internet Explorer.

Importing Third-Party Wallets

Oracle Wallet Manager can import and support the following PKCS #12-format wallets, subject to product-specific procedures and limitations:

- Netscape Communicator 4.x

- Microsoft Internet Explorer 5.x

- OpenSSL

To import a third-party wallet:

1. Follow the product-specific procedure to export the wallet.

2. Save the exported wallet to a platform-specific file name in a directory expected by Oracle Advanced Security.

   For UNIX and Windows NT, the file name is `ewallet.p12`.

   For other platforms, see the platform-specific documentation.

   See Also: Importing a Trusted Certificate on page 16-24.

Notes:

- You must copy the third-party PKCS #12 wallet file name to a directory expected by Oracle Wallet Manager and change the name; the UNIX/NT wallet file name is `ewallet.p12`.

- Since browsers typically do not export **trusted certificate**s under PKCS #12 (other than the signer's own certificate), you may need to add trust points to authenticate the other party in the SSL connection. You can use Oracle Wallet Manager to do this.

Exporting Oracle Wallets

Oracle Wallet Manager can export its own wallets to third party environments. To export a wallet:

1. Use Oracle Wallet Manager to save the wallet file.

2. Follow the third-party product-specific import procedure to import a platform-specific PKCS #12 wallet file created by Oracle Wallet Manager (called `ewallet.p12` on UNIX and NT platforms).

   Note:

   - Oracle Wallet Manager supports multiple certificates for each wallet. However, current browsers typically support import of single-certificate wallets only. Accordingly, for these browsers, you must export an Oracle Wallet containing a single key-pair.

   - *Wallet export is only supported to (i) Netscape Communicator Domestic Version, and to (ii) OpenSSL.*

Multiple Certificate Support

Oracle wallet tools (Oracle Wallet Manager, Enterprise Login Assistant) support multiple **certificates** for each wallet, supporting the following **Oracle PKI certificate usages**:

- SSL
- S/MIME signature
- S/MIME encryption
- Code-Signing
- CA Certificate Signing

Oracle Wallet Manager supports multiple certificates for a *single digital entity,* where each certificate can be used for a set of Oracle PKI certificate usages—but the same certificate cannot be used for all such usages (See: Tables 16–2 and 16–3 for legal usage combinations). There must be a one-to-one mapping between certificate requests and certificates. The same certificate request can be used to obtain multiple certificates. More than one certificate cannot be installed in the same wallet at the same time.

Oracle Wallet Manager uses X.509 V3 extension `KeyUsage` to define Oracle PKI certificate usages (Table 16–1):

Table 16–1   KeyUsage Values

Value	Usage
0	digitalSignature
1	nonRepudiation
2	keyEncipherment
3	dataEncipherment
4	keyAgreement
5	keyCertSign
6	cRLSign
7	encipherOnly
8	decipherOnly

When installing a certificate (user certificate, **trusted certificate**), Oracle Wallet Manager uses Tables 16–2 and 16–3 to map the KeyUsage extension values to Oracle PKI certificate usages:

Table 16–2   OWM Import of User Certificate to an Oracle Wallet

KeyUsage Value	Critical?[1]	Usage
none	na	Certificate is importable for SSL or S/MIME encryption use.
0 alone, or any combination including 0 but excluding 5 and 2	na	Accept certificate for S/MIME signature or code-signing use.
1 alone	Yes	Not importable.
	No	Accept certificate for S/MIME signature or code-signing use.
2 alone, or 2 + any combination excluding 5	na	Accept certificate for SSL or S/MIME encryption use.
5 alone, or any combination including 5	na	Accept certificate for CA certificate signing use.
Any settings not listed above	Yes	Not importable.
	No	Certificate is importable for SSL or S/MIME encryption use.

[1]   If the KeyUsage extension is *critical*, the certificate cannot be used for other purposes.

Table 16–3   OWM Import of Trusted Certificates to an Oracle Wallet

KeyUsage Value	Critical?[1]	Usage
none	na	Importable.
Any combination excluding 5	Yes	Not importable.
	No	Importable.
5 alone, or any combination including 5	na	Importable.

[1]   If the KeyUsage extension is *critical*, the certificate cannot be used for other purposes.

You should obtain certificates from the certificate authority with the correct `KeyUsage` value for the required Oracle PKI certificate usage. A single wallet can contain multiple **key pairs** for the same usage. Each certificate can support multiple Oracle PKI certificate usages, as indicated by Tables 16–2 and 16–3. Oracle PKI applications use the first certificate containing the required PKI certificate usage.

For example: For SSL usage, the first certificate containing the SSL Oracle PKI certificate usage is used.

> **Note:** *SSL Oracle PKI Certificate Usage* is the only usage supported by Oracle PKI applications.

LDAP Directory Support

Oracle Wallet Manager can upload wallets to—and retrieve them from—an LDAP-compliant directory.

Storing wallets in a centralized LDAP-compliant directory lets users access them from multiple locations or devices, ensuring consistent and reliable user authentication—while providing centralized wallet management throughout the wallet life cycle. To prevent accidental over-write of functional wallets, only wallets containing an installed certificate can be uploaded.

> **See Also:**
> - Uploading a Wallet to an LDAP Directory on page 16-14.
> - Downloading a Wallet from an LDAP Directory on page 16-15

Oracle Wallet Manager requires that enterprise users are already defined and configured in the LDAP directory, to be able to upload or download wallets. If a directory contains Oracle8*i* (or prior) users, they are automatically upgraded to use the wallet upload/download feature—upon first use.

> **See Also:** Task 15: Configure Enterprise Users on page 15-65.

Oracle Wallet Manager downloads a user wallet using a simple password based connection to the LDAP directory. However, for uploads it uses an SSL connection if the open wallet contains a certificate with SSL Oracle PKI certificate usage.

> **See Also:** Multiple Certificate Support on page 16-8, for more information about Oracle PKI certificate user.

If an SSL certificate is not present in the wallet, password-based authentication is used.

> **Note:** The directory password and the wallet password are independent, and can be different. Oracle recommends that these passwords are maintained to be consistently different, where neither one can logically be derived from the other.

Managing Wallets

This section describes how to create a new wallet and perform associated wallet management tasks, such as generating certificate requests, exporting certificate requests, and importing certificates into wallets, in the following subsections:

- Starting Oracle Wallet Manager
- Creating a New Wallet
- Opening an Existing Wallet
- Closing a Wallet
- Uploading a Wallet to an LDAP Directory
- Downloading a Wallet from an LDAP Directory
- Saving Changes
- Saving the Open Wallet to a New Location
- Saving in System Default
- Deleting the Wallet
- Changing the Password
- Using Auto Login

Starting Oracle Wallet Manager

To start Oracle Wallet Manager:

- **Windows NT:** Select `Start—>Programs—>Oracle-<ORACLE_HOME_ NAME>—>Network Administration—>Wallet Manager`
- **UNIX:** Enter `owm` at the command line.

Creating a New Wallet

Create a new wallet as follows:

1. Choose `Wallet > New` from the menu bar; the New Wallet dialog box appears.

2. Follow the required guidelines for creating a password and enter a password in the Wallet Password field.

Because an Oracle wallet contains user credentials that can be used to authenticate the user to multiple databases, it is especially important to choose a strong wallet password. A malicious user who guesses the wallet password can access all the databases to which the wallet owner has access.

Oracle Wallet Manager requires that you choose a password that is at least eight characters long, and contains a mix of alphabetic and numeric or special characters.

Example: `gol8fer*`

It is also a prudent security practice for users to change their passwords periodically, such as once a month or once a quarter.

> **See Also:** Wallet Password Management on page 16-3.

3. Re-enter that password in the Confirm Password field.

4. Choose OK to continue.

5. If the entered password does not conform to the required guidelines, the following message appears:

   Password must have a minimum length of eight characters, and contain alphabetic characters combined with numbers or special characters. Do you want to try again?

6. An Alert is displayed, and informs you that a new empty wallet has been created. It prompts you to decide whether you want to create a certificate request. See: **Adding a Certificate Request** on page 16-20.

   If you choose Cancel, you are returned to the Oracle Wallet Manager main window. The new wallet you just created appears in the left window pane. The certificate has a status of `Empty`, and the wallet displays its default trusted certificates.

7. Select `Wallet > Save In System Default` to save the new wallet.

   If you do not have permission to save the wallet in the system default, you can save it to another location.

   A message at the bottom of the window informs you that the wallet was successfully saved.

Opening an Existing Wallet

Open a wallet that already exists in the file system directory as follows:

1. Choose Wallet > Open from the menu bar; the Select Directory dialog box appears.

2. Navigate to the directory location in which the wallet is located, and select the directory.

3. Choose OK; the Open Wallet dialog box appears.

4. Enter the wallet password in the Wallet Password field.

5. Choose OK.

6. The message `Wallet opened successfully` appears at the bottom of the window, and you are returned to the Oracle Wallet Manager main window. The wallet's certificate and its trusted certificates are displayed in the left window pane.

Closing a Wallet

To close an open wallet in the currently selected directory:

- Choose `Wallet > Close`.

- The message `Wallet closed successfully` appears at the bottom of the window, to confirm that the wallet is closed.

Uploading a Wallet to an LDAP Directory

To upload a wallet to an LDAP directory, Oracle Wallet Manager uses SSL if a SSL certificate is contained in the target wallet. Otherwise, it lets you enter the directory password. Note that both Oracle Wallet Manager and Enterprise Login Assistant can upload and download wallets interchangeably.

To prevent accidental destruction of your wallet, Oracle Wallet Manager will not permit you to execute the Upload option, unless the target wallet is currently open and contains at least one user certificate.

To upload a wallet:

1. Choose `Wallet>Upload into the Directory Service`. If the currently open wallet has not been saved, a dialog box appears with the following message:

   Wallet needs to be saved before uploading.

   Choose Yes to proceed.

2. Wallet certificates are checked for key usage `SSL`. If at least one certificate has `SSL` key usage, a dialog box prompts for the server and the port. Enter the server and port information associated with the LDAP directory and choose OK. Oracle Wallet Manager attempts connection to the LDAP directory server using SSL.

3. If upload fails, the following message appears:

   Upload wallet failed

   Otherwise, the following message appears:

   Wallet uploaded successfully.

4. If the target wallet does not contain any certificates with key usage `SSL`, a dialog box prompts for the user **distinguished name (DN)** and the LDAP server and port information. Enter this information and choose OK. Oracle Wallet Manager attempts connection to the LDAP directory server using Simple Password Authentication mode, assuming that the wallet password is the same as the directory password.

5. If the prior step fails, a dialog box prompts for the directory password. Oracle Wallet Manager attempts connection to the LDAP directory server using this password and displays a warning message if the attempt fails. Otherwise, Oracle Wallet Manager displays a successful status message at the bottom of the window.

Downloading a Wallet from an LDAP Directory

When a wallet is downloaded from an LDAP directory, it is resident in working memory; *it is not saved to the file system unless you expressly save it—using any of the Save options described in the following sections.*

> **See Also:**
> - Saving Changes
> - Saving the Open Wallet to a New Location
> - Saving in System Default

To download a wallet from an LDAP directory:

1. Choose `Wallet>Download from the Directory Service.`

2. A dialog box prompts for the user distinguished name, and the directory password, server and port information associated with the source LDAP

directory. Oracle Wallet Manager uses *simple password authentication* to connect to the LDAP directory.

3. If the download operation fails, the following warning message is displayed:

Download wallet failed

4. If the download is successful and there is an existing open wallet, the following message is displayed:

An opened wallet already exists in memory. Do you wish to overwrite it with the downloaded wallet?

Choose OK to open the downloaded wallet.

5. Oracle Wallet Manager attempts to open that wallet using the directory password.

6. If the operation fails (using the directory password), a dialog box prompts for the wallet password.

7. If Oracle Wallet Manager cannot open the target wallet using the wallet password, the following message is displayed:

Open downloaded wallet failed

Otherwise the status:

Wallet downloaded successfully

is displayed at the bottom of the window.

Saving Changes

To save your changes to the current open wallet:

- Choose `Wallet > Save`.

- A message at the bottom of the window confirms that the wallet changes were successfully saved to the wallet in the selected directory location.

Saving the Open Wallet to a New Location

Use the Save As option to save the current open wallet to a new directory location:

1. Choose Wallet > Save As; the select directory dialog box appears.

2. Select a directory location to save the wallet.

3. Choose OK.

The following message appears if a wallet already exists in the selected directory:

```
A wallet already exists in the selected path. Do you
want to overwrite it?.
```

Choose Yes to overwrite the existing wallet, or No to save the wallet to another directory.

A message at the bottom of the window confirms that the wallet was successfully saved to the selected directory location.

Saving in System Default

Use the `Save in System Default` menu option to save the current open wallet to the system default directory location.

- Choose `Wallet > Save in System Default`.

- A message at the bottom of the window confirms that the wallet was successfully saved in the system default wallet location.

> **Note:** Certain Oracle applications are not able to use the wallet if it is not in the system default location.

Deleting the Wallet

To delete the current open wallet:

1. Choose `Wallet > Delete`; the `Delete Wallet` dialog box appears.

2. Review the displayed wallet location to verify you are deleting the correct wallet.

3. Enter the wallet password.

4. Choose OK; a dialog panel appears to inform you that the wallet was successfully deleted.

> **Note:** Any open wallet in application memory will remain in memory until the application exits. Therefore, deleting a wallet that is currently in use does not immediately affect system operation.

Changing the Password

A password change is effective immediately. The wallet is saved to the currently selected directory, with the new encrypted password.To change the password for the current open wallet:

1. Choose `Wallet > Change Password;` the `Change Wallet Password` dialog box appears.

2. Enter the existing wallet password.

3. Enter the new password.

   See Also: Wallet Password Management on page 16-3, for password policy restrictions.

4. Re-enter the new password.

5. Choose OK.

A message at the bottom of the window confirms that the password was successfully changed.

Using Auto Login

The Oracle Wallet Manager Auto Login feature opens a copy of the wallet and enables PKI-based access to secure services—as long as the wallet in the specified directory remains open in memory.

You must enable Auto Login if you want single sign-on access to multiple Oracle databases (disabled by default).

Enabling Auto Login

To enable Auto Login:

1. Choose Wallet from the menu bar.

2. Choose the check box next to the Auto Login menu item; a message at the bottom of the window displays Autologin enabled.

Disabling Auto Login

To disable Auto Login:

1. Choose Wallet from the menu bar.

2. Choose the check box next to the Auto Login menu item; a message at the bottom of the window displays Autologin disabled.

Managing Certificates

Oracle Wallet Manager uses two kinds of certificates: user certificates and trusted certificates. This section describes how to manage both certificate types, in the following subsections:

- Managing User Certificates
- Managing Trusted Certificates

> **Note:** You must first install a trusted certificate from the certificate authority before you can install a user certificate issued by that authority. Several trusted certificates are installed by default when you create a new wallet.

Managing User Certificates

Managing user certificates involves the following tasks:

- Adding a Certificate Request
- Importing the User Certificate into the Wallet
- Removing a User Certificate from a Wallet
- Removing a Certificate Request
- Exporting a User Certificate
- Exporting a User Certificate Request

Adding a Certificate Request

You can use this task to add multiple certificate requests. Note that when creating multiple requests, Oracle Wallet Manager automatically populates each subsequent request dialog box with the content of the initial request—which you can then edit.

The actual certificate request becomes part of the wallet. You can reuse any certificate request to obtain a new certificate. However, you cannot edit an existing certificate request; store only a correctly filled out certificate request in a wallet.

To create a PKCS #10 certificate request:

1. Choose `Operations > Add Certificate Request`; the `Add Certificate Request` dialog box appears.

2. Enter the following information (Table 16–4):

Table 16–4   Certificate Request: Fields and Descriptions

Field Name	Description
Common Name	Mandatory. Enter the name of the user's or service's identity. Enter a user's name in first name /last name format.
Organizational Unit	Optional. Enter the name of the identity's organizational unit. Example: Finance.
Organization	Optional.Enter the name of the identity's organization. Example: XYZ Corp.
Locality/City	Optional. Enter the name of the locality or city in which the identity resides.
State/Province	Optional. Enter the full name of the state or province in which the identity resides. Enter the full state name, because some certificate authorities do not accept two–letter abbreviations.
Country	Mandatory. Choose the drop-down list to view a list of country abbreviations. Select the country in which the organization is located.
Key Size	Mandatory. Choose the drop-down box to view a list of key sizes to use when creating the public/private key pair. See Table 16–5 to evaluate key size.
Advanced	Optional. Choose Advanced to view the Advanced Certificate Request dialog panel. Use this field to edit or customize the identity's distinguished name (DN). For example, you can edit the full state name and locality.

Table 16–5   Available Key Sizes

Key Size	Relative Security Level
512	Not regarded as secure.
768	Provides some security.
1024	Secure.

3. Choose OK. An Oracle Wallet Manager dialog box informs you that a certificate request was successfully created. You can either copy the certificate request text from the body of this dialog panel and paste it into an e-mail message to send to a certificate authority, or you can export the certificate request to a file.

See Also: Exporting a User Certificate Request

4. Choose OK. You are returned to the Oracle Wallet Manager main window; the status of the certificate is changed to `Requested`.

Importing the User Certificate into the Wallet

You will receive an e-mail notification from the certificate authority informing you that your certificate request has been fulfilled. Import the certificate into a wallet in either of two ways: copy and paste the certificate from the e-mail you receive from the certificate authority, or import the user certificate from a file.

Pasting the Certificate

To paste the certificate:

1. Copy the certificate text from the e-mail message or file you receive from the certificate authority. Include the lines `Begin Certificate` and `End Certificate`.

2. Choose `Operations > Import User Certificate` from the menu bar; the Import Certificate dialog box appears.

3. Choose the `Paste the Certificate` button, and choose OK; an Import Certificate dialog box appears with the following message:

   `Please provide a base64 format certificate and paste it below.`

4. Paste the certificate into the dialog box, and choose OK. A message at the bottom of the window confirms that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the status of the corresponding entry in the left panel subtree changes to `Ready`.

Selecting a File that Contains the Certificate

To select the file:

1. Choose `Operations > Import User Certificate` from the menu bar.

2. Choose the `Select a file...` certificate button, and choose OK; the Import Certificate dialog box appears.

3. Enter the path or folder name of the certificate location.

4. Select the name of the certificate file (for example, `cert.txt`).

5. Choose OK. A message at the bottom of the window appears, to inform you that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the status of the corresponding entry in the left panel subtree changes to `Ready`.

Removing a User Certificate from a Wallet

1. In the left panel subtree, select the certificate that you want to delete.

2. Choose `Operations > Remove User Certificate`; a dialog panel appears and prompts you to verify that you want to remove the user certificate from the wallet.

3. Choose Yes; you are returned to the Oracle Wallet Manager main panel, and the certificate displays a status of `Requested`.

Removing a Certificate Request

To remove a certificate request:

1. In the left panel subtree, select the certificate request that you want to delete.

2. Choose `Operations Menu.`

3. Select menu item `Remove Certificate Request.`

Note: You must remove a certificate before removing its associated request.

Exporting a User Certificate

Save the certificate in a file system directory when you elect to export a certificate:

1. In the left panel subtree, select the certificate that you want to export.

2. Choose `Operations > Export User Certificate` from the menu bar; the Export Certificate dialog box appears.

3. Enter the file system directory to save your certificate in, or navigate to the directory structure under Folders.

4. Enter a file name to save your certificate, in the Enter File Name field.

5. Choose OK. A message at the bottom of the window confirms that the certificate was successfully exported to the file. You are returned to the Oracle Wallet Manager main window.

Exporting a User Certificate Request

Save the certificate request in a file system directory when you elect to export a certificate request:

1. In the left panel subtree, select the certificate request that you want to export.

2. Choose `Operations > Export Certificate Request` from the menu bar; the Export Certificate Request dialog box appears.

3. Enter the file system directory in which you want to save your certificate request, or navigate to the directory structure under Folders.

4. Enter a file name to save your certificate request, in the Enter File Name field.

5. Choose OK. A message at the bottom of the window confirms that the certificate request was successfully exported to the file. You are returned to the Oracle Wallet Manager main window.

Managing Trusted Certificates

Managing trusted certificates includes the following tasks:

- Importing a Trusted Certificate
- Removing a Trusted Certificate
- Exporting a Trusted Certificate
- Exporting All Trusted Certificates
- Exporting a Wallet

Importing a Trusted Certificate

You can import a trusted certificate into a wallet in either of two ways: paste the trusted certificate from an e-mail that you receive from the certificate authority, or import the trusted certificate from a file.

Oracle Wallet Manager automatically installs trusted certificates from VeriSign, RSA, Entrust, and GTE CyberTrust when you create a new wallet.

Pasting the Trusted Certificate To paste the trusted certificate:

1. Choose `Operations > Import Trusted Certificate` from the menu bar; the Import Trusted Certificate dialog panel appears.

2. Choose the `Paste the Certificate` button, and choose OK. An Import Trusted Certificate dialog panel appears with the following message:

   ```
   Please provide a base64 format certificate and paste it
   below.
   ```

3. Copy the trusted certificate from the body of the e-mail message you received that contained the user certificate. Include the lines `Begin Certificate` and `End Certificate`.

4. Paste the certificate into the window, and Choose OK. A message at the bottom of the window informs you that the trusted certificate was successfully installed.

5. Choose OK; you are returned to the Oracle Wallet Manager main panel, and the trusted certificate appears at the bottom of the Trusted Certificates tree.

Selecting a File that Contains the Trusted Certificate

To select the file:

1. Choose `Operations > Import Trusted Certificate` from the menu bar. The Import Trusted Certificate dialog panel appears.

2. Enter the path or folder name of the trusted certificate location.

3. Select the name of the trusted certificate file (for example, `cert.txt)`.

4. Choose OK. A message at the bottom of the window informs you that the trusted certificate was successfully imported into the wallet.

5. Choose OK to exit the dialog panel; you are returned to the Oracle Wallet Manager main panel, and the trusted certificate appears at the bottom of the Trusted Certificates tree.

Removing a Trusted Certificate

To remove a trusted certificate from a wallet:

1. Select the trusted certificate listed in the Trusted Certificates tree.

2. Choose `Operations > Remove Trusted Certificate` from the menu bar.

A dialog panel warns you that your user certificate will no longer be verifiable by its recipients if you remove the trusted certificate that was used to sign it.

3. Choose Yes; the selected trusted certificate is removed from the Trusted Certificates tree.

Note: A certificate that is signed by a trusted certificate is no longer verifiable when you remove it from your wallet.

Also, you cannot remove a trusted certificate if it has been used to sign a user certificate that is still present in the wallet. To remove such a trusted certificate, you must first remove the certificates that it has signed.

Exporting a Trusted Certificate

To export a trusted certificate to another file system location:

1. In the left panel subtree, select the trusted certificate that you want to export.

2. Select `Operations > Export Trusted Certificate`; the Export Trusted Certificate dialog box appears.

3. Enter a file system directory in which you want to save your trusted certificate, or navigate to the directory structure under Folders.

4. Enter a file name to save your trusted certificate.

5. Choose OK; you are returned to the Oracle Wallet Manager main window.

Exporting All Trusted Certificates

To export all of your trusted certificates to another file system location:

1. Choose `Operations > Export All Trusted Certificates`. The Export Trusted Certificate dialog box appears.

2. Enter a file system directory in which you want to save your trusted certificate, or navigate to the directory structure under Folders.

3. Enter a file name to save your trusted certificates.

4. Choose OK; you are returned to the Oracle Wallet Manager main window.

Exporting a Wallet

You can export a wallet to text-based PKI formats. Individual components are formatted according to the following standards (Table 16–6). Within the wallet, only those certificates with key usage SSL are exported with the wallet.

Table 16–6   PKI Wallet Encoding Standards

Component	Encoding Standard
Certificate chains	X509v3
Trusted certificates	X509v3
Private keys	PKCS #8

17

Using Oracle Enterprise Login Assistant

Use Oracle Enterprise Login Assistant to open and close wallets, to update centrally managed wallets and passwords in an LDAP directory, and to enable or disable secure SSL connections.

This chapter describes Oracle Enterprise Login Assistant, and contains the following topics:

- About Oracle Enterprise Login Assistant
- Starting Oracle Enterprise Login Assistant
- Opening Existing Wallet on Local System
- Connecting to LDAP Directory and Downloading New Wallet
- Changing Wallet Passwords
- Uploading Wallet to LDAP Directory
- Logging Out and Disabling SSL Connection

> **See Also:** Chapter 16, Using Oracle Wallet Manager, for information about managing wallets with Oracle Wallet Manager.

About Oracle Enterprise Login Assistant

Oracle Enterprise Login Assistant is a client-side tool used to authenticate users to the enterprise. It can authenticate users to an LDAP directory service, and download an Oracle wallet from the directory. It can also decrypt the wallet and let users establish a seamless SSL connection to all PKI-enabled applications and databases within the enterprise—without requiring additional passwords. Enterprise Login Assistant can update the directory password (OID only) and other related passwords stored in the directory, and it can also upload an Oracle Wallet to the directory.

Starting Oracle Enterprise Login Assistant

Refer to your platform-specific documentation for instructions about how to start
Oracle Enterprise Login Assistant.

Opening Existing Wallet on Local System

Upon startup, Oracle Enterprise Login Assistant searches for an installed wallet in the default system location—defined in your platform-specific documentation. If it finds an installed wallet, the main login window appears; in this example, no local wallet was found (Figure 17–1):

Figure 17–1   Enterprise Login Assistant Login Window
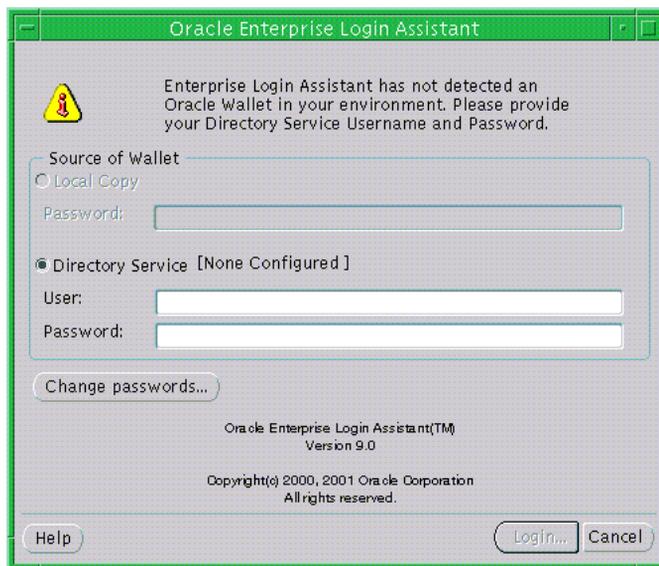
To establish a secure SSL connection using the default wallet:

1. Choose the Local Copy button.

2. Enter the wallet password.

3. Choose the Login button.

    Enterprise Login Assistant creates an obfuscated copy of the wallet in the local file system, and you are returned to the *logged-in* state; the Logged-In Window appears (Figure 17–2). This confirms that the wallet was opened successfully, and that a successful connection has been established.

Figure 17–2   Enterprise Login Assistant Logged-In Window

> **Note:**   If Enterprise Login Assistant finds an obfuscated wallet upon startup, it assumes that you are already logged in—automatically changing to *logged-in* state.

Connecting to LDAP Directory and Downloading New Wallet

Enterprise Login Assistant can download a new wallet from an LDAP directory to your local system. This is necessary for your first connection to an Oracle enterprise.

To connect to an LDAP directory and download a new wallet:

1. Choose the Directory Service button (Figure 17–3).

Figure 17–3   Enterprise Login Assistant Directory Login Window
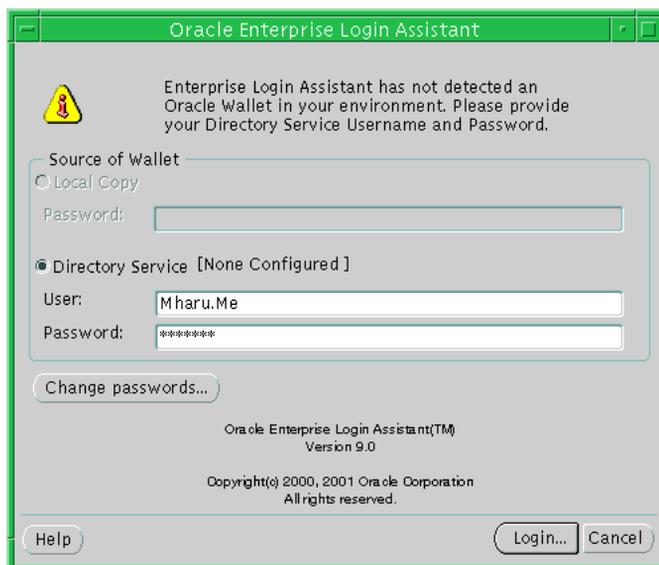
2. Enter your directory UserID and password.

3. Choose the Login button.

   Enterprise Login Assistant attempts to connect to the directory and download a wallet. If there is no default directory service configured, it prompts for the hostname and port of an alternative directory service (contact your System Administrator for further details).

4. Enterprise Login Assistant stores the wallet in the default location on the local system and attempts to decrypt it using the directory password. If the directory password is different from the wallet password, it prompts you for the wallet password.

5. Enterprise Login Assistant creates an obfuscated copy of the wallet in the local file system, and you are returned to the logged-in state; the Logged-In Window appears (Figure 17–2). This confirms that the wallet was successfully copied to the local system and opened.

Changing Wallet Passwords

You can use Enterprise Login Assistant to change any or all of the following passwords:

- The wallet password
- The directory password
- The database password

To change a password:

1. Choose the Change Password button from the Logged In Window (Figure 17–2); the Change Password Window appears (Figure 17–4):

Figure 17–4   Enterprise Login Assistant Change Password Window
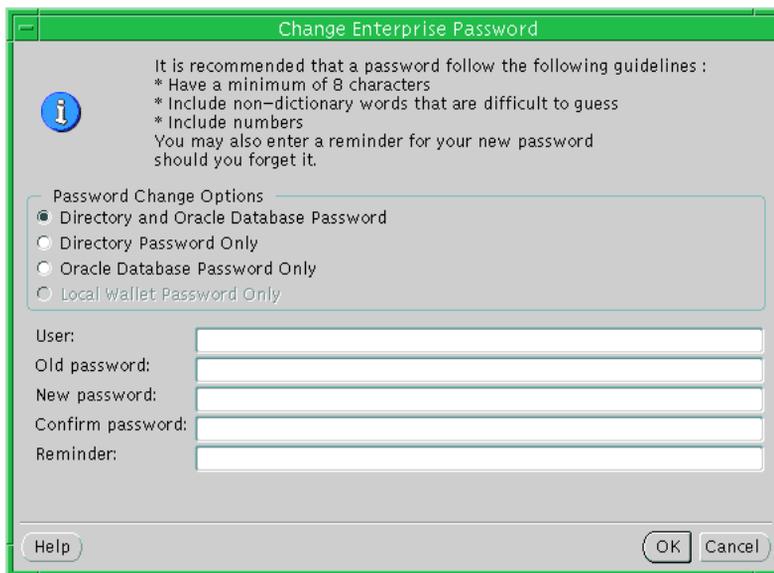
2. Choose one of the following password change options:

- All Passwords Below
- Wallet Password Only
- Directory Password Only

- Oracle Database Password Only

3. Enter your directory UserID.

4. Enter your *existing* password in the Old password field.

5. Enter your *new* password, in accordance with your password policy, and confirm it by entering it again.

6. Enter an optional password hint in the Reminder field.

7. Choose the OK button.

If the Old Password you entered matches the existing password(s), Enterprise Login Assistant updates the selected passwords with the new password and optional hint. Enterprise Login Assistant displays the following message to confirm successful update of the new password(s):

```
Password changed successfully.
```

Choose the OK button to exit the dialog box.

> **Notes:**
>
> - If you choose Wallet Password Only, a UserID is not required, and you cannot enter a Reminder.
>
> - Enterprise Login Assistant provides users with a broad range of control over their own passwords and credentials. However, your enterprise installation may have special security requirements that limit the applicability of this tool. Security Administrators can adjust the enterprise-wide security schema in the LDAP directory to inhibit users from updating certain passwords, or to force users to make all passwords identical—which would disable the individual password selection options in the Change Password Window (Figure 17–4).

Uploading Wallet to LDAP Directory

To upload a wallet to an LDAP directory:

1. Choose the Upload Wallet button in the Logged-In Window (Figure 17–2).

2. If you have already authenticated to the LDAP directory service in the current session, the local obfuscated wallet is in a closed state (encrypted); a copy of the wallet is uploaded to the directory, replacing the existing wallet.

3. If you have not yet authenticated to the LDAP directory service in the current session, Enterprise Login Assistant prompts you for your directory UserID and password to connect you to the directory before Step 2 is performed.

Logging Out and Disabling SSL Connection

Use Enterprise Login Assistant to disable single sign-on communications from server-side applications.

To log out and disable the SSL connection:

1. Choose the Logout button from the Logged-In Window (Figure 17–2).

    Enterprise Login Assistant displays the following warning:

    ```
    If you log out, your applications will no longer use the
    security credentials of your wallet.
    ```

2. Choose the Yes button to continue; you are returned to the Login Window (Figure 17–1).

18

Using Oracle Enterprise Security Manager

This chapter describes how to use Oracle Enterprise Security Manager to administer Enterprise User Security in Oracle9*i* databases. This chapter contains the following topics:

- Introduction
- Installing and Configuring Oracle Enterprise Security Manager
- Administering a Directory for Enterprise User Security
- Administering Enterprise Users
- Administering Oracle Contexts

> **See Also:** *Oracle Internet Directory Administrator's Guide*

Introduction

Oracle Enterprise Security Manager, a component of Oracle Enterprise Manager, is an administration tool employed by Oracle Advanced Security to manage **enterprise users**, **enterprise domains**, databases, and **enterprise role**s that are held in an LDAP-compliant directory service.

The directory service is used as a central repository to define user and server access information for a network. It stores naming information, global password definitions, PKI credentials, and application access authorizations for the users that it defines. Such centralized storage of enterprise users and their access privileges supports single sign-on capability, and provides secure, scalable user administration.

Installing and Configuring Oracle Enterprise Security Manager

The following tasks describe how to use Oracle Enterprise Security Manager to install Oracle Management Server and Oracle Enterprise Manager:

- Task 1: Configure an Oracle Internet Directory
- Task 2: Install Oracle Enterprise Manager
- Task 3: Configure Oracle Enterprise Manager for Enterprise User Security
- Task 4: Start Oracle Enterprise Security Manager
- Task 5: Log On to the Directory

Task 1: Configure an Oracle Internet Directory

Oracle9*i* Enterprise User Security is based on an LDAP-compliant directory. The directory server must be properly installed and configured before Oracle Enterprise Manager can be used to manage Enterprise User Security. The following elements of directory configuration must be completed before proceeding:

- A compatible LDAP-compliant directory must be installed, running, and accessible over both standard LDAP and Secure Sockets Layer LDAP (LDAP/SSL). Oracle Advanced Security Release 9.0.1 is compatible with the following LDAP-compliant directories:

  - Oracle Internet Directory (8*i*, 9*i*)

  - Microsoft Active Directory

    See Also: *Oracle Internet Directory Administrator's Guide*

- Oracle Internet Directory must be configured to support Oracle9*i* directory schema objects and must include an **Oracle Context**. You can use Oracle Net Configuration Assistant to configure both of these on the directory server.

    See Also: *Oracle Net Services Administrator's Guide*

Task 2: Install Oracle Enterprise Manager

Oracle Enterprise Manager is automatically installed by the Oracle9*i* Enterprise Edition server installation process, and includes all necessary functionality to support Enterprise User Security. Oracle Enterprise Manager is also installed by default with the Oracle9*i* infrastructure installation at the same time as Oracle

Internet Directory. Oracle Enterprise Manager can also be installed separately in its own ORACLE_HOME, using the custom install option.

See Also:

- *Oracle Enterprise Manager Installation*
- *Oracle Enterprise Manager Administrator's Guide*

Task 3: Configure Oracle Enterprise Manager for Enterprise User Security

You can use Oracle Enterprise Manager to manage Enterprise User Security in two modes of operation:

- You can use Oracle Enterprise Manager Console to connect to the Oracle 9i Management Server (OMS) and discover a Directory Server to manage.
- Alternatively, you can launch Oracle Enterprise Security Manager from the same ORACLE_HOME as Oracle Enterprise Manager connect directly to the directory server.

The functionality is identical in either mode of operation. Only the latter mode, Oracle Enterprise Security Manager, is described in this chapter.

> **Note:**   Oracle Enterprise Security Manager does not require special configuration for it to run. However, all Oracle databases in the enterprise that use Oracle Enterprise Security Manager must be accessible over Oracle Net from the Oracle Enterprise Manager ORACLE_HOME.

Task 4: Start Oracle Enterprise Security Manager

To launch Oracle Enterprise Security Manager from the Enterprise Manager ORACLE_HOME, enter the following at the command line:

```
oemapp esm
```

The directory login box appears (Figure 18–1):

Figure 18–1   Directory Server Login Window
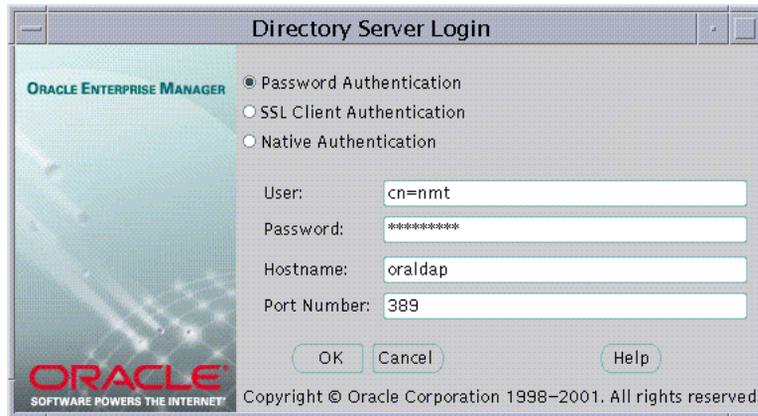
Task 5: Log On to the Directory

Oracle Enterprise Security Manager provides three ways to connect to a directory server, summarized by Table 18–1:

Table 18–1   ESM Authentication Methods

Authentication Method	Description
Password Authentication	Uses simple authentication requiring a **distinguished name (DN)** or a known directory UserID and a password (i.e., user name and password).
SSL Client Authentication	Uses two-way SSL authentication in which both the client and server use Oracle Wallets containing digital certificates (i.e., user name and certificate). The subsequent connection is encrypted.
Native Authentication	Applies to Microsoft Windows NT and Windows 2000 only; uses operating system-level authentication to log on to a Microsoft Active Directory.

To select an authentication method, choose the appropriate option in the Directory Server Login Window (Figure 18–1).

Administering a Directory for Enterprise User Security

Oracle Enterprise Security Manager displays the following window after the initial connection (Figure 18–2):

Figure 18–2   ESM: Main Window (Directory Tab)

Oracle Enterprise Security Manager manages one directory server, identified at the top of the main application tree, followed by a series of menu operations that apply to this server.

You use Enterprise Security Manager to manage users in the directory. The application shows the directory to which it is connected and lets you delete and browse users in that directory. Oracle Enterprise Security Manager can also be used to manage **Oracle Contexts** in the directory. An Oracle Context is a subtree in a directory recognizable to Oracle products. It provides an administrative hierarchy for management of Oracle data—including installed Oracle products that access the directory.

Administering Enterprise Users

This section describes how to use Oracle Enterprise Security Manager to administer enterprise users. It contains the following topics:

- Creating New Enterprise Users

- Defining a Directory Base

- Defining a New Enterprise User Password

- Defining an Initial Enterprise Role Assignment

- Viewing an Oracle Wallet

- Browsing Users in the Directory

- Enabling Database Access

Creating New Enterprise Users

Use Oracle Enterprise Security Manager to create users in the directory.

To create new users, select Create Enterprise User... from the Operations menu (Figure 18–3):

Figure 18–3   ESM: Operations Menu

The Create User window appears (Figure 18–4).

Figure 18–4   ESM: Create User Window (User Naming Tab)

Referring to Table 18–2, enter the appropriate user information required by the User Naming tabbed window; choose OK to create a new enterprise user.

Table 18–2   Enterprise User Fields

Field Name	Mandatory?	Description
base	Yes	The entry in the directory under which the new user is created.
First Name	Yes	Given name.
Surname	Yes	Surname (last name).
UserID	Yes	The user name (Logon Identifier) that the user can use to connect to the network, databases, and applications.
Apply Suffix	No	The current value of any common UserID suffix that is appended to the UserID. For example: `<userID>.us.acme.com`
Email Address	No	The new user's email address.

Table 18–2   Enterprise User Fields

Field Name	Mandatory?	Description
cn=	No	The Common Name component (cn=) of the Distinguished Name (DN) of the new user in the directory. By default it is set to the full name of the new user. However, you can override this value to force a particular value for the cn portion of the DN.

Defining a Directory Base

An enterprise user entry can reside at any **base** within the directory. The base can be any existing directory entry, such as *country entry* (c=us), or an *organization entry* (o=acme,c=us). Multiple users typically share the same directory base. This base associates all the users contained under it with the same high level organization in the hierarchy.

You can enter the base in the base field of the Create User window (Figure 18–4). Alternatively, you can browse the entire directory to select a suitable base by choosing the Browse... button (in the same window); the Browse Directory Window appears (Figure 18–5):

Figure 18–5   ESM: Browse Directory Window
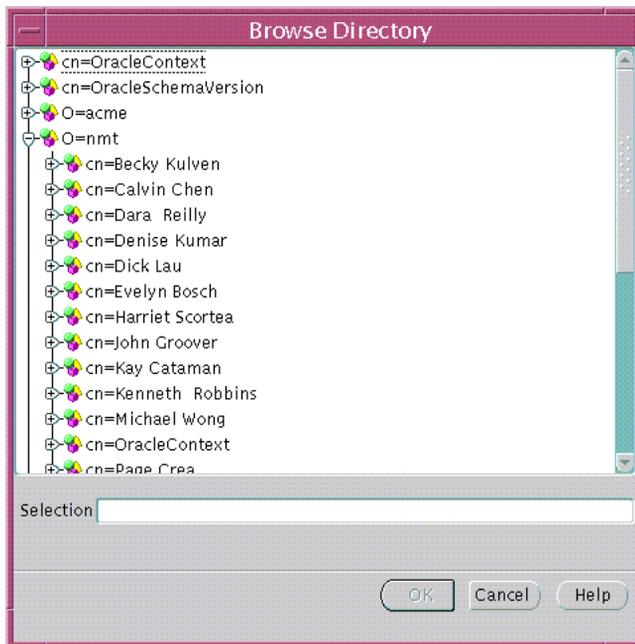
The Browse Directory window lets you navigate the directory by drilling down into each entry from the top of the directory tree. When a directory **entry** is selected its **distinguished name (DN)** is placed in the Selection field. To accept the selected Distinguished Name choose the OK button. This value is returned as the selected base for a new directory user, and is preserved for all subsequent operations that create or search for users in the directory—although you can change it from time to time.

Defining a New Enterprise User Password

The Password tab of the Create User Window (Figure 18–6) lets you define and maintain the enterprise user password:

Figure 18–6   ESM: Create User Window (Password Tab)
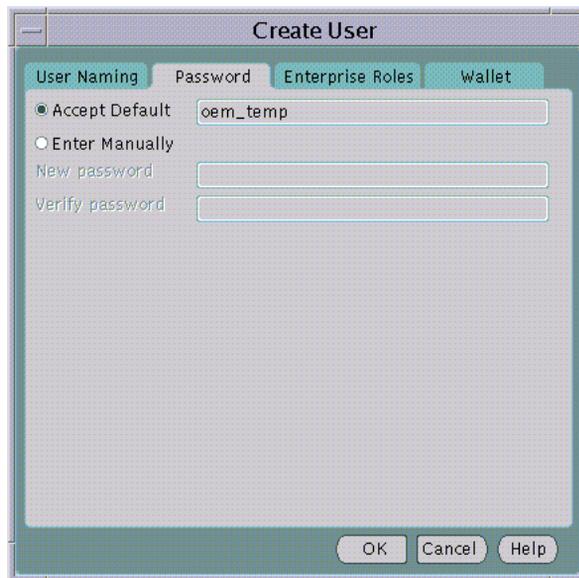
The enterprise user password is used for:

- Directory logon.

- Database logon, to databases that support password authentication for global users.

- A new Oracle Wallet, if created for the new user at this time.

When creating a new password, you can accept a default password or manually enter and confirm a new password. *In either case, the new user must change the password immediately after its first use.*

> **See Also:**   Chapter 17, Using Oracle Enterprise Login Assistant

Defining an Initial Enterprise Role Assignment

When you create a new enterprise user, you can grant any previously configured enterprise roles to a new user.

> **See Also:** Administering Enterprise Roles on page 18-36

To select one or more enterprise roles to grant to a new user, choose the Add... button on the Enterprise Roles tab of the Create User window (Figure 18–7):

Figure 18–7   ESM: Create User Window (Enterprise Roles Tab)

The Add Enterprise Roles window appears (Figure 18–8):

Figure 18–8   ESM: Add Enterprise Roles Window
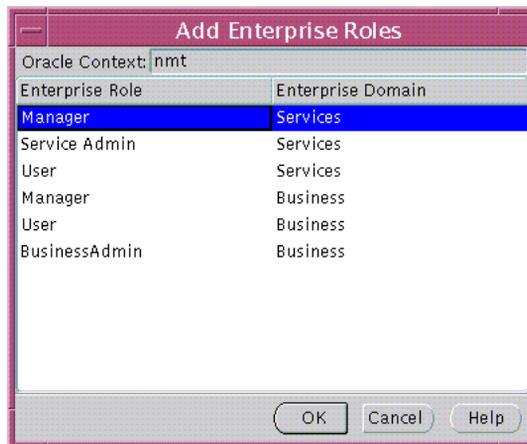
Select any enterprise roles in your Oracle Context to assign to the new user; choose
OK.

Viewing an Oracle Wallet

You can use Oracle Enterprise Security Manager to view a user **wallet**, stored in the
directory as part of the directory entry for the user.

You can use Oracle Wallet Manager to create new user wallets, and to upload and
download wallets from the directory.

> **See Also:**   Chapter 16, Using Oracle Wallet Manager

Browsing Users in the Directory

Oracle Enterprise Security Manager lets you browse the directory for all users
currently stored.

To browse enterprise users, choose the All Users tab in the main window (Figures
18–2, 18–9):

Figure 18–9   ESM: Main Window (All Users Tab)
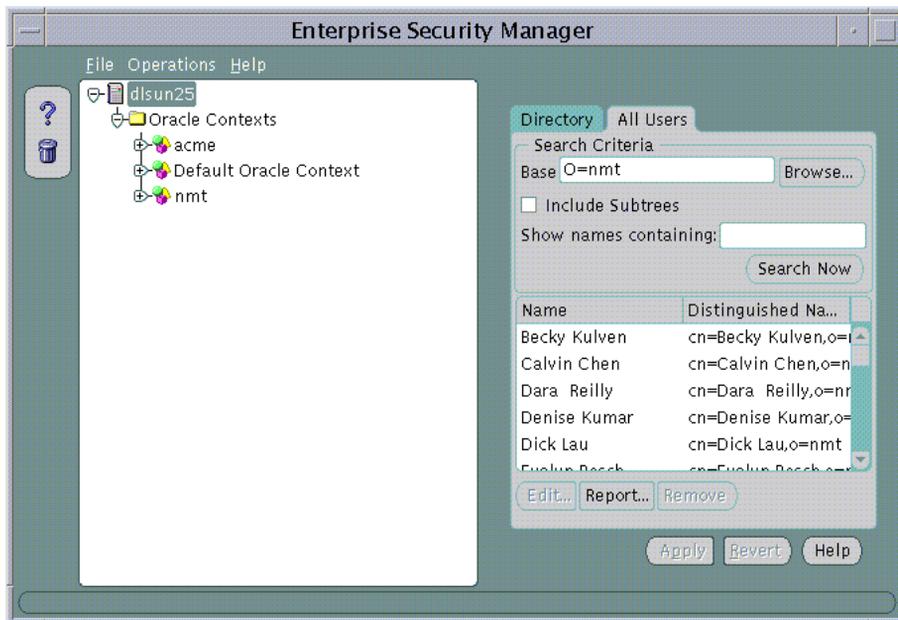
To search for users in the directory, define the search criteria and choose the Search Now button. The window displays the results of the search. Table 18–3 summarizes the search criteria and their respective effects on the search results:

Table 18–3   Directory Search Criteria

Search Criteria	Effect on the Search
Base	This is the base entry point in the directory where the search is performed. Only users under this base are returned by the search.
Include Substrees	This determines whether to show *all* users found in the entire subtree under the selected base, or to only show only those users that exist directly under that base location (one level only).
Show names containing	This *limits the search* to those users whose directory entries have a common name that starts with the characters you specify. This is useful if you do not know the exact name or base of the target users.

Example 1:

Searching an Oracle directory for a user named Richard (Figure 18–10):

Figure 18–10   ESM: Searching Directory for User Richard
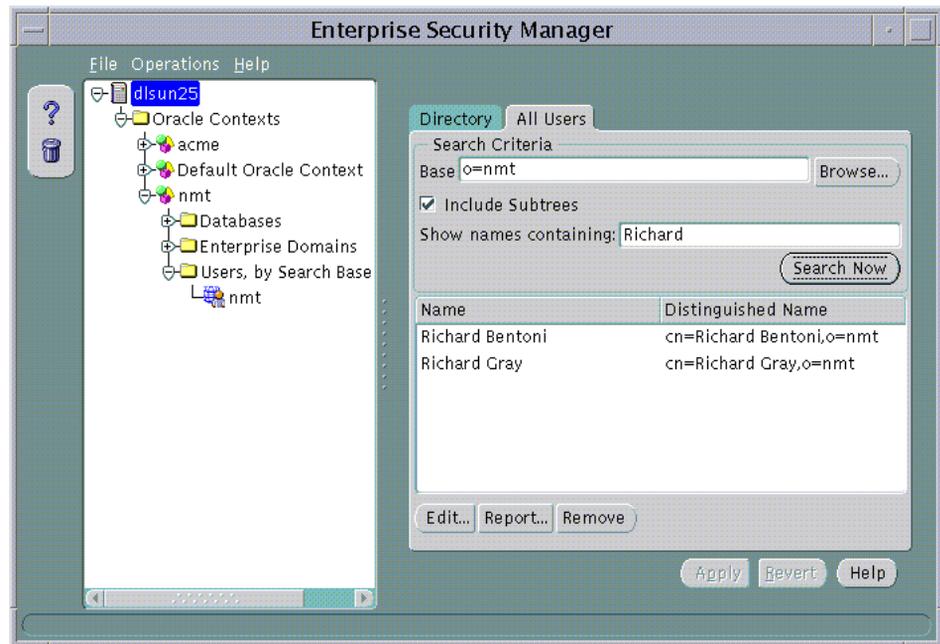
Example 2:

Selecting a user from the search results for editing.

To edit one of the returned user names, select the target user name and choose the Edit... button—or just double-click the target user name in the list (Figure 18–11):

Figure 18–11   ESM: Edit User Window
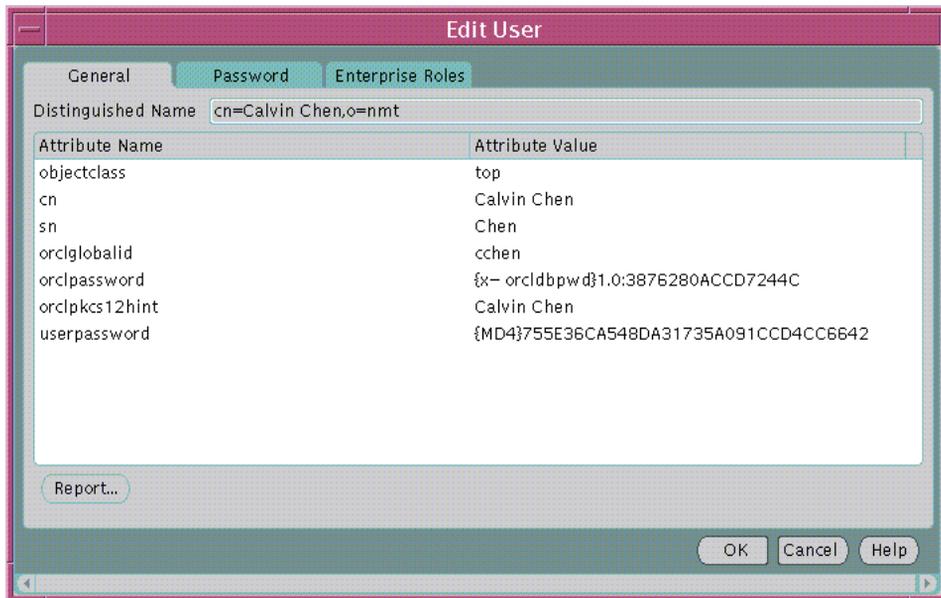
When you select a directory user for edit, you can change the password and enterprise role assignments—and you can modify the user wallet in the same manner as during its initial creation.

> **See Also:**
>
> - Creating New Enterprise Users on page 18-7
> - Viewing an Oracle Wallet on page 18-13

Enabling Database Access

The user entry must reside in a directory subtree of users that has been enabled for Oracle database access. You can set Oracle Database Access permissions for a selected subtree—to let databases within a domain in the Password-Accessible Domains group read the user's login credentials.

To enable database access:

On a selected subtree of directory users, set Oracle Database Access permissions to permit databases in the Password-Accessible Domains group to access the user's database login credentials:

- Select the target user subtree under Users, by Search Base.
- Select Database Access Restriction for that subtree.

Administering Oracle Contexts

An **Oracle Context** is a subtree in a directory that contains the data used by any installed Oracle product that uses the directory. Oracle Enterprise Security Manager is one such product. It lets you manage database and security-related information in the directory, in an Oracle Context.

> **Note:** It is not necessary to create users within an Oracle Context, though it is acceptable to do so (the directory can define its users for a wide variety of purposes).

> **See Also:** Chapter 15, Managing Enterprise User Security

Oracle Context Versions

Oracle Enterprise Security Manager can support multiple Oracle Contexts in a directory, including Oracle8*i* and Oracle9*i* versions. However, *Oracle9i Enterprise User Security can only be managed using an Oracle9i Oracle Context.* Enterprise manager for oracle 9i may be used to manage version 9i oracle contexts as well version 8i oracle contexts in the directory.

Oracle Enterprise Security Manager displays *all* existing Oracle Contexts in its main application tree—including both Oracle8*i* and Oracle9*i* versions. In the following example (Figure 18–12), Oracle Enterprise Security Manager is connected to an Oracle directory that has been configured to support the Oracle9*i* directory schema and an Oracle9*i* root Oracle Context.

Defining Properties of an Oracle Context

An Oracle Context has a number of properties that can be viewed and managed in the Enterprise Security Manager window (Figure 18–12, Table 18–4):

Figure 18–12   ESM: General Tab
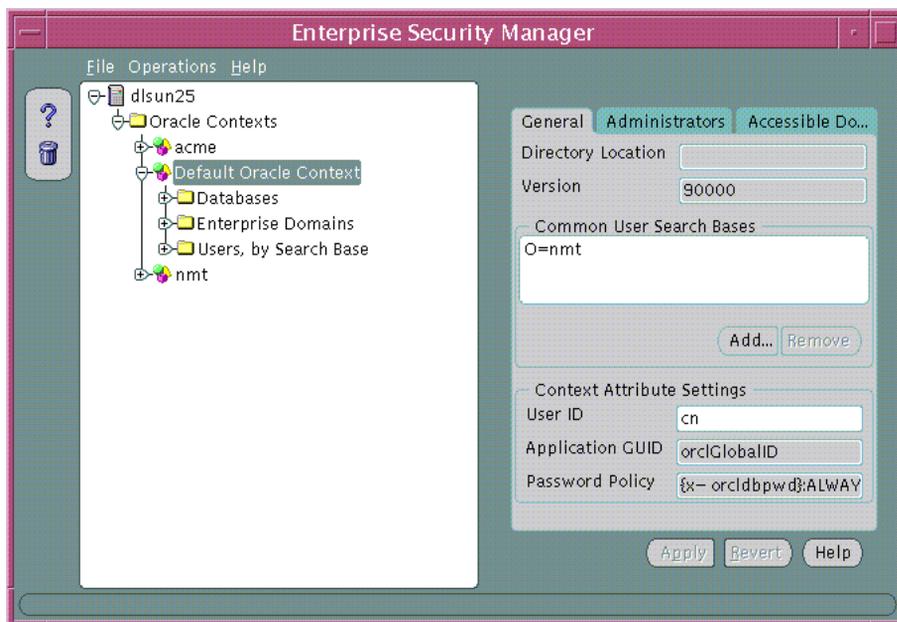
> **Note:**   The reference to *Default Oracle Context* in Figure 18–12
> should read *Root Oracle Context*; all references to *Default Oracle
> Context* will be changed to *Root Oracle Context* in the production
> release of Oracle Advanced Security.

To define or edit properties of an Oracle Context, refer to Table 18–4:

Table 18–4   Oracle Context Properties

Property	Description
Directory Location	The parent of the Oracle Context. In the case of the root Oracle context this value is empty, as the context is at the root of the directory tree.
Version	This defines the Oracle Context Version: Oracle8*i* or Oracle9*i*.
Versioncompatibility	This defines whether the Oracle Context supports Oracle8*i*, Oracle9*i*, or both of them.

Table 18–4   Oracle Context Properties

Property	Description
Common User Search Bases	The list of base locations in the directory at which users may commonly exist. Identifying a list of user search bases lets you quickly browse the users at those directory locations, and also indicates to Oracle9*i* databases in the Oracle Context where they can find directory users that connect to them.
UserID	The UserID attribute uniquely identifies users in the enterprise; a globally unique identifier for each user. Users use the value in the UserID attribute to authenticate to Oracle9*i* databases, directory servers, or directory enabled applications. The default value is cn, the common name of the directory user.
Application GUID	The name of the attribute in a user entry in which unique application GUID values exist. It cannot be modified in this release.
Password Policy	The password policy syntax used by Oracle9*i* databases when authenticating password authenticated global users. *It cannot be modified in this release.*

Defining User Search Bases

Common user search bases can be added to or removed from an Oracle9*i* Oracle Context using the General tabbed window (Figure 18–12).

> **Note:**   This functionality is not available for Oracle8*i* Oracle Contexts.

To remove a user search base from an Oracle Context:

1. Using the Oracle Enterprise Security Manager General tabbed window (Figure 18–12), select a search base from the *Common User Search Bases* list, and choose the Remove... button.

2. Choose the Apply button; the user search base is removed from the Oracle Context in the directory.

To add a new user search base to an Oracle Context:

1. Using the Oracle Enterprise Security Manager General tabbed window
    (Figure 18–12), choose the Add... button; the Browse Directory window appears
    (Figure 18–13):

Figure 18–13   ESM: Browse Directory (User Search Bases)
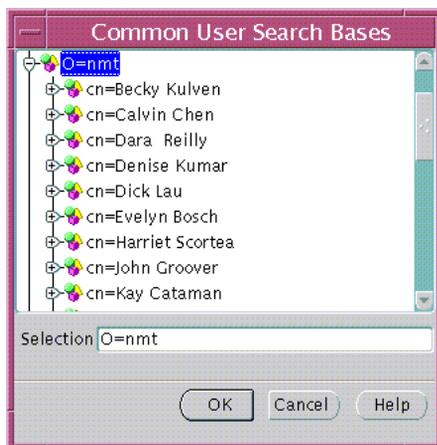
2. Navigate the directory tree and select an entry for a user search base.
    Alternatively, you can edit the contents of the Selection field in this window to
    manually define the user search base.

3. Choose OK; the selected entry is added to the list of user search bases in the
    General tabbed window (Figure 18–12).

4. Choose Apply (Figure 18–12); the user search base is added to the Oracle
    Context in the directory.

Defining Oracle Context Administrators

An Oracle Context contains administrative groups that have varying levels of
privileges for operations within an Oracle Context. Some administrative groups are
only available to Oracle9*i* Oracle Contexts and some are available to both Oracle8*i*
and Oracle9*i* Oracle Contexts. The administrative groups for an Oracle Context are
defined by Table 18–5:

Table 18–5   Oracle Context Administrators

Administrative Group	Definition	Oracle9*i* Version	Oracle8*i* Version
Full Context Management	All possible Administrator privileges for all product areas in the Oracle Context.	Yes	No
Directory User Management	Can view directory user password reminders and update passwords.	Yes	No
Database Security Management	Can manage all enterprise domains and roles in the Oracle Context.	Yes	Yes
Database Registration	Can register a new database in the Oracle Context.	Yes	Yes
Oracle Net Management	Can manage Oracle Net objects in the Oracle Context.	Yes	Yes

Use the *Administrators* tab of the Oracle Enterprise Security Manager main window to manage Oracle Context Administrators(Table 18–14):

Figure 18–14   ESM Administrator's Tab

To remove a user from a list of Oracle Context Administrators:

1. Choose the Administrator Category (Table 18–5); a list of administrators within this category is displayed.

2. Select a user name from the list.

3. Choose the Remove button; the selected user is removed from the list.

4. Choose the Apply button; the selected user is removed as an Oracle Context Administrator from the selected Administrator Category.

Figure 18–15   ESM: Add Users Window
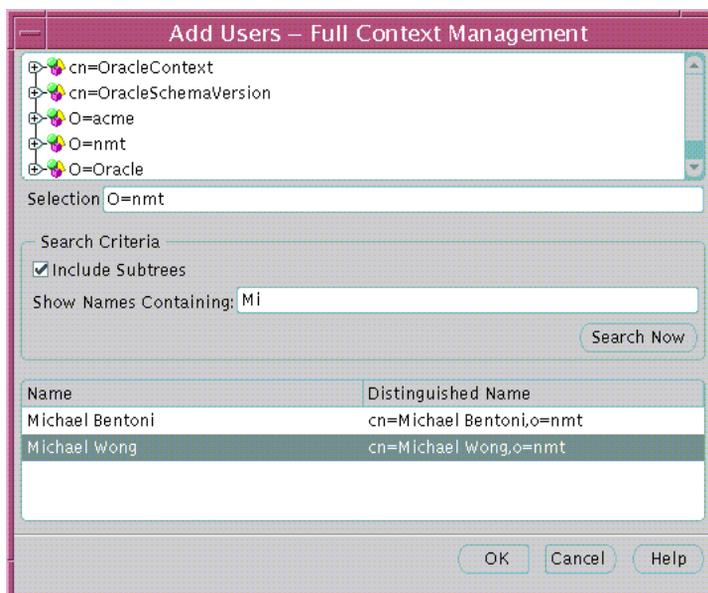
To add a new user to the list of Oracle Context Administrators:

1. Choose the Add... button in Figure 18–14; the Add Users screen appears (Figure 18–15).

   Use this window to locate and select users in the directory. There are three panels in the window:

   - **Top panel:** The directory search tree.

- ■ **Middle panel:** Search criteria that determine the users returned by the search.

- ■ **Bottom panel:** Search results—users found in the directory that match the search criteria.

2. Navigate the Directory (in the top panel) to select a directory entry as a user search base. You can edit the contents of the selection field in this window to manually define the user search base.

3. Check the *Include Subtrees* option in the middle panel (Search Criteria). This selection option searches for all users within the search base, including substrees.

4. Enter any known User Name in the *Show Names Containing* field to which user names returned by the search must conform. This limits the search to users in the directory who have a common name value that is or starts with the specified text.

5. Choose the Search Now button (middle panel). If there are any users in the directory at the base you have selected that match your search criteria they are listed in the window.

6. Select the desired user name either by selecting it from the list and choosing OK, or by double-clicking it. Multiple users can be selected from the list by selecting a range of users and choosing OK. The new users appear in the list of Administrators under the category you have selected.

> **Note:** This window is commonly used throughout Oracle Enterprise Security Manager where it is necessary to select users from the directory.

7. In order to accept password-authenticated connections, a database must belong to a domain in the Password Accessible Domains group—and the database access permissions on the user search base must be enabled. This enables the database to read the user's login credentials in the directory.

   In a selected Oracle9*i* Oracle Context, add the domain to the *Password-Accessible Domains* group. Choose Add and select one of the current enterprise domains from the resulting dialog. To remove an enterprise domain from the group, select it in the Accessible Domains window and choose Remove.

See Also:

- Step 5: Enable Database Access on page 15-69

- Security of User Database Login Information on page 15-11

Managing Password Accessible Domains

There are three requirements for a database to accept a connection from a password-authenticated user:

- The database must be a member of a domain configured to accept (i) *Password and SSL,* or (ii) *Password Only user authentication* (See: Table 18–7).

- The domain must be a member of a *password-accessible* domains group, called the **Password-Accessible Domains List**, added by an Oracle Context Administrator or a Database Security Administrator. Domain members of this list can read the user's password verifier in the directory, while those excluded from this list cannot. *The domain must be part of an Oracle9i Oracle Context.*

- The user entry must be in a directory subtree of users that has been enabled for Oracle database access. You can set Oracle Database Access permissions for a selected subtree that lets databases in the Password-Accessible Domains List read the users' database login information.

To configure password accessibility:

1. Add the target database to an enterprise domain that has been configured to accept (i) *Password and SSL,* or (ii) *Password Only* user authentication.

   See Also:

   - Defining Database Membership of an Enterprise Domain on page 18-32

   - Managing Database Security Options for an Enterprise Domain on page 18-34

2. In a selected Oracle9*i* Oracle Context, add the domain to the Password-Accessible Domains List. Choose Add and select one of the current **enterprise domains** from the resulting dialog. To remove an enterprise domain from the list, select it in the Accessible Domains window and choose Remove.

3. On a selected subtree of directory users, set Oracle database access permissions to permit databases in the Password-Accessible Domains List to access the users' database login information:

- Select the target user subtree under *Users, by Search Base.*

- Select *Database Access Restriction* for that subtree.

    See Also: Security of User Database Login Information on page 15-11

Note: Password accessible domains require an Oracle9*i* Oracle Context.

Managing Database Security

The directory can be used as a central repository that controls user authentication and authorization on multiple databases. Oracle Enterprise Security Manager lets you to manage an Oracle Context in the directory for database security.

Both Oracle8*i* and Oracle9*i* databases are published to the directory within an Oracle Context using the Oracle Database Configuration Assistant. Once databases are published to the directory, you can use Oracle Enterprise Security Manager to manage user access to those databases. This is achieved using the following objects in the Oracle Context (Table 18–6):

Table 18–6   ESM: Oracle Context Objects

Object	Description
Database	A directory entry representing a published database.
Enterprise Domain	A grouping of databases published in the directory, upon which a common user access model for database security can be implemented
Enterprise Role	An Authorization that spans multiple databases within an **enterprise domain**. It is an **enterprise role** to which individual roles can be granted on each of the databases in an enterprise domain.
Mapping	A mapping object is used to map the **distinguished name (DN)** of a user to a database schema that the user will access.

See Also:

- *Oracle9i Database Administrator's Guide*
- Chapter 15, **Using Oracle Enterprise Security Manager**

Administering Databases

After a database has been published to an Oracle Context in the directory, Oracle Enterprise Security Manager can be used to view and modify security characteristics of that database.

Managing Database Administrators

A **Database Administrator** is a directory user that has privileges to modify the database and its subtree in the Oracle Context. Database Administrators may be managed using the Administrators tabbed window when a database is selected under an Oracle Context in the main application tree (Figure 18–14).

To remove a user from the list of Database Administrators:

1. Select a user from the list of administrators.
2. Choose Remove; the selected user is removed from the list.
3. Choose Apply; the user is removed as a Database Administrator for that database in the Oracle Context.

To add a new user to the list of **Enterprise Domain Administrators**:

1. Choose Add; the Add Users window appears (Figure 18–15). Use this window to locate and select users in the directory.
2. Select a user or users from the directory to be added as a Database Administrator; the new user(s) is displayed in the Administrators tabbed window (Figure 18–14).
3. Choose Apply; the new Administrator(s) is added to the database in the Oracle Context.

See Also:

- Creating New Enterprise Users on page 18-7

- Browsing Users in the Directory on page 18-13

Managing Database Schema Mappings

Database **schema mappings** let databases that are registered in the directory accept connections from users without requiring any dedicated database **schemas** for them. For example, when user *Scott* connects to a database, a database schema called *Scott* must exist—for that logon to be successful. This can be difficult to maintain if there are thousands of users and perhaps hundreds of databases in a very large enterprise.

Users that are defined in an LDAP-compliant directory do not require dedicated schemas on every Oracle8*i* or Oracle9*i* database to which they might connect.

A database can use a schema mapping to share one database schema between multiple directory users. The schema mapping is a pair of values: the base in the directory at which users exist, and the name of the database schema they will use.

You can use the Database Schema Mappings tabbed window to manage database schema mappings—when a database is selected under an Oracle Context in the main application tree. This window contains a list of database schema names and Directory Base pairs (Figure 18–16):

Figure 18–16   ESM: Database Schema Mappings Tab

To remove a mapping from the list of database schema mappings in an enterprise domain:

1.   Select a mapping by selecting from the Database Schema Mapping tabbed window.

2.   Choose Remove. The selected Mapping is removed from the list.

3.   Choose Apply; the mapping is removed from the enterprise domain.

To add a new mapping to the list of database schema mappings in the enterprise domain:

1.   Choose Add...; the Add Database Schema Mappings window appears (Figure 18–17):

Figure 18–17   ESM: Add Database Schema Mappings Window
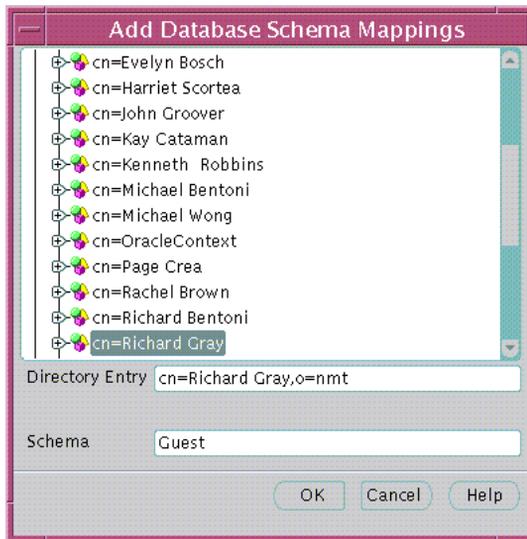
Use this window to locate and select a base in the directory and pair it with a database schema name, to make a database schema mapping. There are two components to the window: there is a directory search tree from which to select a base, and a field in which to enter a schema name.

2. Navigate the directory to select a desired entry as a base for the database schema mapping. This can be any directory entry but should be located above the subtree of users to be mapped. You can also edit the contents of the *Directory Entry* field in this window to manually define the base.

3. Enter the name of the database schema for which this Mapping will be made into the Schema field, and choose OK. This must be a valid name, for a schema that already exists on that database.The new database schema mapping appears in the database schema mappings window (Figure 18–16).

4. Choose Apply; the new database schema mapping is added to the selected database in the Oracle Context.

Administering Enterprise Domains

An Oracle Context contains at one enterprise domain called `OracleDefaultDomain`. The `OracleDefaultDomain` is part of the Oracle Context when it is first created in the directory. When a new database is registered

into an Oracle Context it automatically becomes a member of the
OracleDefaultDomain in that Oracle Context. You can create and remove your
own enterprise domains but you cannot remove the OracleDefaultDomain from
an Oracle Context.

To create a new enterprise domain in an Oracle Context, use either of the following
methods:

- Select Create Enterprise Domain from the Operations menu (Figure 18–16).
- Select an Oracle Context from the main application tree with a right
  mouse-click.

  The Create Enterprise Domain window appears (Figure 18–18):

Figure 18–18   ESM: Create Enterprise Domain Window

To create the new enterprise domain:

1. Select the appropriate Oracle Context from the drop-down list (Figure 18–18).

> **Note:** If you invoked the Create Enterprise Domain window by
> right-clicking the Oracle Context in the main application tree, the
> name of that Oracle Context is already selected.

2. Enter the name of the new enterprise domain, in the Domain Name field.

3. Choose OK; the new enterprise domain is created in the Oracle Context, and
appears on the main application tree.

To remove an enterprise domain:

1. Select the target enterprise domain from the main application tree
(Figure 18–16).

2. Use either of the following methods:

   ■ Select Remove Enterprise Domain from the Operations menu.

   ■ Select an enterprise domain from the main application tree with a right mouse-click.

3. Oracle Enterprise Security Manager asks you to confirm removal of the enterprise domain from the Oracle Context; choose OK to remove it.

Note: You cannot remove an enterprise domain from an Oracle Context if that enterprise domain still contains any enterprise roles.

Defining Database Membership of an Enterprise Domain

Use the application tree of the main Oracle Enterprise Security Manager window to select a target enterprise domain. You can then use the Databases tab to manage database membership of an enterprise domain in an Oracle Context (Figure 18–19):

Figure 18–19   ESM: Databases Tab (Database Membership)
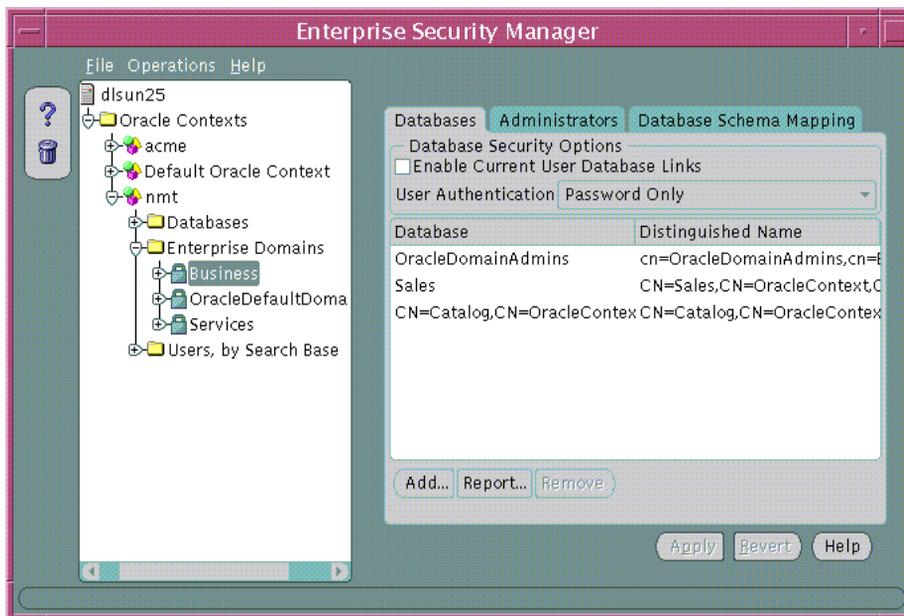
To remove a database from an enterprise domain:

1. Select a target database for removal, and choose Remove...; the database is removed from the list.

2. Choose Apply; the database is removed from the enterprise domain in the Oracle Context.

To add a database to an enterprise domain:

> **Note:** You can only add a database to an enterprise domain if both the database and the enterprise domain exist *in the same Oracle Context.* It follows, therefore, that:
>
> - An enterprise domain cannot contain a database published in a different Oracle Context.
>
> - A database in an Oracle Context cannot be added as a member of two different enterprise domains.

1. Choose Add... (Figure 18–19); the Add Databases window appears. This window lists all the databases associated with the Oracle Context (Figure 18–20):

Figure 18–20 ESM: Add Databases Window
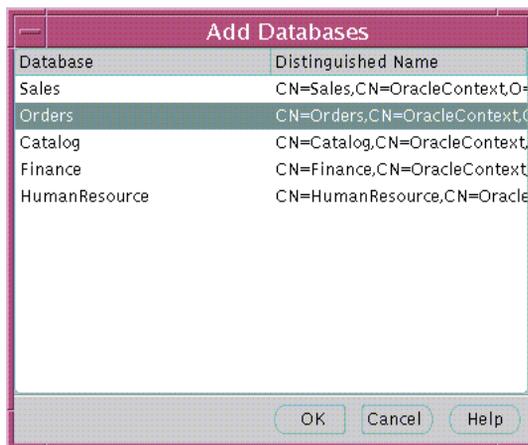
2. Select a new target database to be added to the enterprise domain.

3. Choose OK; the selected database is added to the list of databases in the Databases tabbed window (Figure 18–19).

4. Choose Apply (Figure 18–19); the new database is added to the enterprise domain in the Oracle Context.

Managing Database Security Options for an Enterprise Domain

Use the Databases tabbed window (Figure 18–19) to manage database security options applicable to all databases that are members of the enterprise domain.

Database security options are summarized by Table 18–7:

Table 18–7   ESM Database Security Options

Database Security Option	Description
Enable current user database links	Any database pair can only permit use of *Current User Database Links* if both databases exist in an enterprise domain in which this setting is enabled.
User authentication	All databases in an enterprise domain must enforce one of the following types of authentication for its clients: ■ Password Authentication only. ■ Oracle Net SSL Authentication only using Oracle Wallets. ■ Either Password or Oracle Net SSL Authentication (default).

Managing Enterprise Domain Administrators

An **Enterprise Domain Administrator** is a directory user in an enterprise domain that has privileges to modify the content of that domain. You can use the Administrators tabbed window (Figure 18–14) to manage Enterprise Domain Administrators when an enterprise domain is selected under an Oracle Context in the main application tree.

To remove a user from the list of Enterprise Domain Administrators:

1. Select a user from the list of Administrators.

2. Choose Remove; the selected user is removed from the list.

3. Choose Apply; the user is removed as an Enterprise Domain Administrator for that domain in the Oracle Context.

To add a new user to the list of Enterprise Domain Administrators:

1. Choose Add... (Figure 18–14); the Add Users window appears (Figure 18–15). Use this window to locate and select target users for designation as Enterprise Domain Administrators. The new users appear in the Administrators tabbed window (Figure 18–14).

2. Choose Apply (Figure 18–14); the new Administrators are added to the enterprise domain in the Oracle Context.

Managing Enterprise Domain Database Schema Mappings

As previously discussed, database schema mappings can be managed for each database in an Oracle Context. Schema mappings can also be defined for each enterprise domain in an Oracle Context, using the database schema mappings tabbed window with an enterprise domain selected in the main application tree. These mappings apply to all databases that are members of the enterprise domain. Therefore, each database in the enterprise domain must have a schema of the same name used in the mapping for that mapping to be effective on that database.

Figure 18–21   ESM: Database Schema Mappings Tab
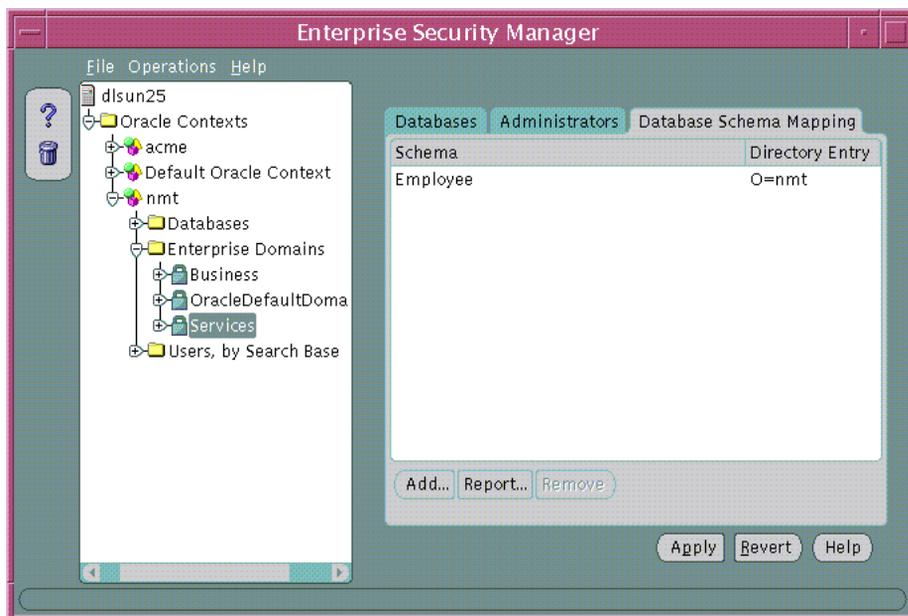
To remove a mapping from the list of database schema mappings in the enterprise domain (Figure 18–21):

1. Select a mapping from the Database Schema Mappings list.

2. Choose Remove; the selected mapping is removed from the list.

3. Choose Apply; the mapping is removed from the enterprise domain.

To add a new mapping to the list of database schema mappings in the enterprise domain (Figure 18–21):

1. Choose Add...; the Add Database Schema Mappings window appears. Use this window to locate and select a base in the directory for the new mapping, as discussed previously.

2. Enter a new database schema mapping to the enterprise domain.

3. Choose Apply; the new database schema mapping is added to the enterprise domain selected in the Oracle Context.

> **See Also:**
> - Managing Database Schema Mappings on page 18-28
> - Defining a Directory Base on page 18-9

Administering Enterprise Roles

An **enterprise domain** within an **Oracle Context** can contain multiple **enterprise roles**. An enterprise role is a set of Oracle role-based **authorizations** across one or more databases in an enterprise domain.

To create a new enterprise role:

You can create an enterprise role in an enterprise domain either from the Operations menu on the Oracle Enterprise Security Manager main window (Figure 18–21), or by right-clicking an enterprise domain in the main application tree. In either case, the Create Enterprise Role window appears (Figure 18–22):

Figure 18–22   ESM: Create Enterprise Role Window
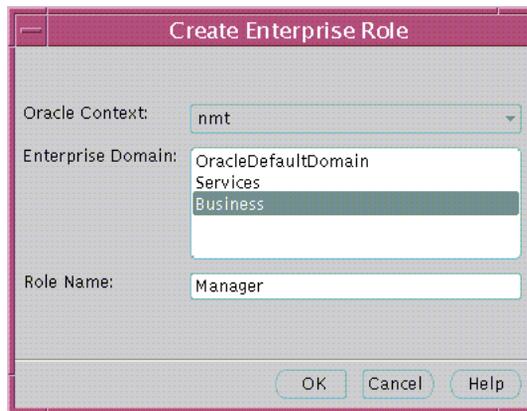
1. Choose the target Oracle Context from the Oracle Context drop-down list; this is the Oracle Context containing the target enterprise domain—to hold the new enterprise role.

    > **Note:**   If you invoked the Create Enterprise Role window by right-clicking an enterprise domain, the name of the Oracle Context is already selected.

2. Select the appropriate enterprise domain for the new enterprise role, from the Enterprise Domain list.

    > **Note:**   If you invoked the Create Enterprise Role window by right-clicking an enterprise domain, the name of the enterprise domain is already selected.

3. Enter the name of the new enterprise role in the Role Name field.

4. Choose OK; the new enterprise role is created in the enterprise domain, and appears on the main application tree.

To remove an enterprise role:

1. Select the target enterprise role from the main application tree (Figure 18–21).

2. Choose Remove Enterprise Role, either from the Operations menu or by right-clicking the enterprise domain in the main application tree.

3. Oracle Enterprise Security Manager asks you to confirm the removal of the enterprise role; choose Yes.

Assigning Database Global Role Membership to an Enterprise Role

Use the Database Global Roles tabbed window (Figure 18–23) of the Oracle Enterprise Security Manager main window to manage database global role membership in an enterprise role. This window lists the names of each **global role** that belongs to the enterprise role, along with the name of the database on which that global role exists.

Figure 18–23   ESM: Database Global Roles Tab

When populating an enterprise role with different database roles it is only possible to reference roles on databases that are configured to be *global roles* on those databases. A global role on a database is identical to a normal role, except that the **Database Administrator** has defined it to be authorized only via the directory. A Database Administrator cannot locally grant and revoke global roles to users of the database.

To remove a database global role from an enterprise role:

1. Select a global role from the list in the main application tree, and choose Remove...; the global role is removed from the list.

2. Choose Apply; the global role is removed from the enterprise role in the enterprise domain.

To add a global role to an enterprise role:

1. Choose Add... (Figure 18–23); the Add Global Database Roles window appears. This window lists all of the databases in the enterprise domain—from which global roles can be selected to add to an enterprise role.

2. Select a database from which to obtain global roles. A window appears and prompts you for logon details to authenticate to the database (and fetch global roles). Typically, this is a DBA logon to that database.

Figure 18–24   ESM: Database Authentication Required Window
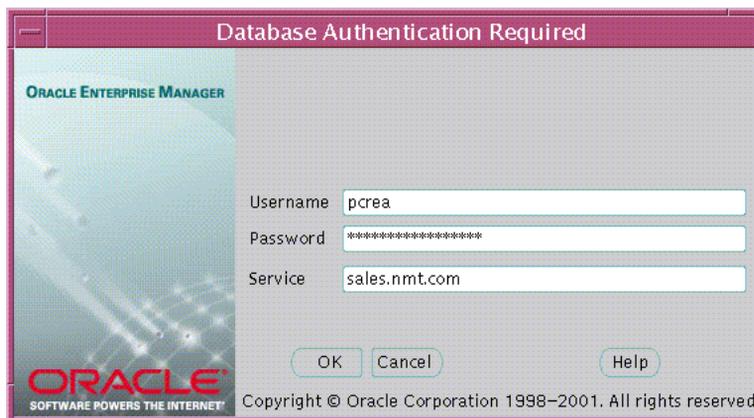
> **Note:** The name of the database appears in the Service field by default. You can use this name to connect to the database if your ORACLE_HOME has LDAP enabled as its Oracle Net naming method, or if this name appears as a TNS alias in your local Oracle Net configuration. Otherwise, you can overwrite the content of the Service field with any other TNS alias configured for that database, or by a connect string in the format:
>
> `<host>:<port>:<oracle sid>`
>
> **Example:** `cartman:1521:broncos`

3. Choose OK; Oracle Enterprise Security Manager connects you to the given database and fetches the list of global roles supported on that database. The list of values, if any, is displayed in the Add Global Database Roles window.

4. Select one or more global roles from the list of returned values and choose **OK**; these global roles appear in the Database Global Roles tabbed window (Figure 18–23).

5. Choose Apply; the new global roles are added to the enterprise role in the enterprise domain.

Managing Enterprise Role Grantees

An enterprise role grantee is a directory user granted an enterprise role, including all database global roles contained within that enterprise role. You can use the Enterprise Users tabbed window (Figure 18–25) to manage enterprise role grantees, when an enterprise role is selected under an enterprise domain in the main application tree.

To remove a user from the list of enterprise role grantees (Figure 18–25):

1. Select a user from the list of grantees.

2. Choose Remove; the selected user is removed from the list.

3. Choose Apply; the user is removed as a grantee for that enterprise role in the enterprise domain.

To add a new user to the list of enterprise role grantees:

1. Choose Add...; the Add Users window appears (Figure 18–15). Use this window to locate and select one or more directory users to add as enterprise role grantees. The new users appear in the Enterprise Users Page (Figure 18–25):

Figure 18–25   ESM: Enterprise Users Tab

2. Choose Apply; the new grantees are added to the enterprise role in the enterprise domain.

You can assign enterprise roles to this newly created enterprise user by selecting the user and choosing the Enterprise Role tab.

> **See Also:**   Defining an Initial Enterprise Role Assignment on page 18-11

Part VI

Appendixes

This part contains the following reference appendixes:

- Appendix A, Data Encryption and Integrity Parameters
- Appendix B, Authentication Parameters
- Appendix C, Integrating Authentication Devices Using RADIUS
- Appendix D, Oracle Advanced Security FIPS 140-1 Settings
- Appendix E, Oracle Implementation of Java SSL
- Appendix F, Abbreviations and Acronyms

A

Data Encryption and Integrity Parameters

This appendix describes **encryption** and data **integrity** parameters supported by Oracle Advanced Security. It also includes an example of a `sqlnet.ora` file generated by performing the network configuration described in Chapter 2, Configuring Data Encryption and Integrity, and Chapter 7, Configuring Secure Sockets Layer Authentication.

This appendix contains the following topics:

- Sample sqlnet.ora File
- Data Encryption and Integrity Parameters

Sample sqlnet.ora File

This section contains a sample `sqlnet.ora` configuration file for a set of clients with similar characteristics and a set of servers with similar characteristics. The file includes examples of Oracle Advanced Security encryption and data integrity parameters.

Trace File Setup

```
#Trace file setup
trace_level_server=16
trace_level_client=16
trace_directory_server=/orant/network/trace
trace_directory_client=/orant/network/trace
trace_file_client=cli
trace_file_server=srv
trace_unique_client=true
```

Oracle Advanced Security Encryption

```
#ASO Encryption
sqlnet.encryption_server=accepted
sqlnet.encryption_client=requested
sqlnet.encryption_types_server=(RC4_40)
sqlnet.encryption_types_client=(RC4_40)
sqlnet.crypto_seed = "-kdje83kkep39487dvmlqEPTbxxe70273"
```

Oracle Advanced Security Integrity

```
#ASO Checksum
sqlnet.crypto_checksum_server=requested
sqlnet.crypto_checksum_client=requested
sqlnet.crypto_checksum_types_server = (MD5)
sqlnet.crypto_checksum_types_client = (MD5)
```

SSL

```
#SSL
WALLET_LOCATIOn = (SOURCE=
                       (METHOD = FILE)
                       (METHOD_DATA =
                        DIRECTORY=/wallet)

SSL_CIPHER_SUITES=(SSL_DH_anon_WITH_RC4_128_MD5)
SSL_VERSION= 3
SSL_CLIENT_AUTHENTICATION=FALSE
```

Common

```
#Common
automatic_ipc = off
sqlnet.authentication_services = (beq)
names.directory_path = (TNSNAMES)
```

Kerberos

```
#Kerberos
sqlnet.authentication_services = (beq, kerberos5)
sqlnet.authentication_kerberos5_service = oracle
sqlnet.kerberos5_conf= /krb5/krb.conf
sqlnet.kerberos5_keytab= /krb5/v5srvtab
sqlnet.kerberos5_realms= /krb5/krb.realm
sqlnet.kerberos5_cc_name = /krb5/krb5.cc
sqlnet.kerberos5_clockskew=900
sqlnet.kerberos5_conf_mit=false
```

CyberSafe

```
#CyberSafe
sqlnet.authentication_services = (beq, cybersafe)
sqlnet.authentication_gssapi_service = oracle/cybersaf.us.oracle.com
sqlnet.authentication_kerberos5_service = oracle
sqlnet.kerberos5_conf= /krb5/krb.conf
sqlnet.kerberos5_keytab= /krb5/v5srvtab
sqlnet.kerberos5_realms= /krb5/krb.realm
sqlnet.kerberos5_cc_name = /krb5/krb5.cc
sqlnet.kerberos5_clockskew=900
```

RADIUS

```
#Radius
sqlnet.authentication_services = (beq, RADIUS )
sqlnet.radius_authentication_timeout = (10)
sqlnet.radius_authentication_retries = (2)
sqlnet.radius_authentication_port = (1645)
sqlnet.radius_send_accounting = OFF
sqlnet.radius_secret = /orant/network/admin/radius.key
sqlnet.radius_authentication = radius.us.oracle.com
sqlnet.radius_challenge_response = OFF
sqlnet.radius_challenge_keyword = challenge
sqlnet.radius_challenge_interface =
oracle/net/radius/DefaultRadiusInterface
sqlnet.radius_classpath = /jre1.1/
```

Data Encryption and Integrity Parameters

If you do not specify any values for Server Encryption, Client Encryption, Server Checksum, or Client Checksum, the corresponding configuration parameters do not appear in the `sqlnet.ora` file. However, Oracle Advanced Security defaults to ACCEPTED.

For both data encryption and integrity algorithms, the server selects the first algorithm listed in its `sqlnet.ora` file that matches an algorithm listed in the client `sqlnet.ora` file, or in the client installed list—if the client lists no algorithms in its `sqlnet.ora` file. If there are no entries in the server `sqlnet.ora` file, the server sequentially searches its installed list to match an item on the client side—either in the client `sqlnet.ora` file or in the client installed list. *If no match can be made and one side of the connection REQUIRED the algorithm type (data encryption or integrity), the connection fails.* Otherwise, the connection succeeds with the algorithm type inactive.

Data encryption and integrity algorithms are selected independently of each other; encryption can be activated without integrity, and integrity can be activated without encryption, as shown by Table A–1:

Table A–1   Algorithm Type Selection

Encryption Selected?	Integrity Selected?
Yes	No
Yes	Yes
No	Yes
No	No

There are three classes of parameters required to enable data encryption and integrity:

- Encryption and Integrity Level Settings

- Encryption and Integrity Selected Lists

- Seeding the Random Key Generator

    See Also:

    - Chapter 2, Configuring Data Encryption and Integrity

    - Activating Encryption and Integrity on page 2-6

Encryption and Integrity Level Settings

Table A–2 summarizes data encryption and integrity level settings:

Table A–2   Encryption and Integrity Level Settings

Algorithm Type	Platform	Item	Description
Encryption	Server	Purpose	This parameter specifies the desired encryption behavior when a client or a server acting as a client connects to this server. The behavior of the server partially depends on the SQLNET.ENCRYPTION_CLIENT setting at the other end.
		Syntax	`SQLNET.ENCRYPTION_SERVER = valid_value`
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	**ACCEPTED**
	Client	Purpose	This parameter specifies the desired encryption behavior when this client or server acting as a client connects to a server. The behavior of the client partially depends on the value set for SQLNET.ENCRYPTION_SERVER at the other end of the connection.
		Syntax	`SQLNET.ENCRYPTION_CLIENT = valid_value`
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	**ACCEPTED**

Table A–2   Encryption and Integrity Level Settings

Algorithm Type	Platform	Item	Description
Integrity	Server	Purpose	This parameter specifies the desired data integrity behavior when a client or another server acting as a client connects to this server. The behavior partially depends on the SQLNET.CRYPTO_CHECKSUM_CLIENT setting at the other end.
		Syntax	`SQLNET.CRYPTO_CHECKSUM_SERVER = valid_value`
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	**ACCEPTED**
	Client	Purpose	This parameter specifies the desired data integrity behavior when this client or server acting as a client connects to a server. The behavior partially depends on the SQLNET.CRYPTO_CHECKSUM_SERVER setting at the other end of the connection.
		Syntax	`SQLNET.CRYPTO_CHECKSUM_CLIENT = valid_value`
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	**ACCEPTED**

Encryption and Integrity Selected Lists

Table A–3   Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description
Encryption	Server	Purpose	This parameter specifies a list of encryption algorithms used by this server, in the order of intended use. This list is used to negotiate a mutually acceptable algorithm with the client end of the connection. Each algorithm is checked against the list of available client algorithm types until a match is found. If an algorithm that is not installed is specified on this side, the connection terminates with error message ORA-12650.
		Syntax	SQLNET.ENCRYPTION_TYPES_SERVER = (*valid_ encryption_algorithm* [,v*alid_encryption_algorithm*])
		Values	■ RC4_256: RSA RC4 (256-bit key size). ■ 3DES168: 3-key Triple-DES (168-bit effective key size). ■ RC4_128: RSA RC4 (128-bit key size). ■ 3DES112: 2-key Triple-DES (112-bit effective key size). ■ RC4_56: RSA RC4 (56-bit key size). ■ DES: Standard DES (56-bit key size). ■ RC4_40: RSA RC4 (40-bit key size). ■ DES40: DES40 (40-bit key size).
		Default	If no algorithms are defined in the local `sqlnet.ora` file, all installed algorithms are used in a negotiation—in the above sequence.

Table A–3   Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description
Encryption	Server	Usage Notes	You can specify multiple encryption algorithms—either a single value or a list of algorithm names. For example, either of the following encryption parameters is acceptable: SQLNET.ENCRYPTION_TYPES_SERVER=(RC4_40) SQLNET.ENCRYPTION_TYPES_SERVER=(DES,RC4_56,RC4_128,DES40)
	Client	Purpose	This parameter specifies a list of encryption algorithms used by this client or server acting as a client. This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. If an algorithm that is not installed is specified on this side, the connection terminates with error message ORA-12650.
		Syntax	`SQLNET.ENCRYPTION_TYPES_CLIENT = (valid_ encryption_algorithm [,valid_encryption_ algorithm])`
		Values	■ RC4_256: RSA RC4 (256-bit key size). ■ 3DES168: 3-key Triple-DES (168-bit effective key size). ■ RC4_128: RSA RC4 (128-bit key size). ■ 3DES112: 2-key Triple-DES (112-bit effective key size). ■ RC4_56: RSA RC4 (56-bit key size). ■ DES: Standard DES (56-bit key size). ■ RC4_40: RSA RC4 (40-bit key size). ■ DES40: DES40 (40-bit key size).
		Default	If no algorithms are defined in the local `sqlnet.ora` file, all installed algorithms are used in a negotiation.

		Usage Notes	You can specify multiple encryption algorithms—either a single value or a list of algorithm names. For example, either of the following encryption parameters is acceptable:
			
			`SQLNET.ENCRYPTION_TYPES_CLIENT=(DES,DES40,RC4_ 56,RC4_40)`
			`SQLNET.ENCRYPTION_TYPES_CLIENT=(RC4_40)`

Table A–3   Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description
Integrity	Server	Purpose	This parameter specifies a list of data integrity algorithms this server or client to another server uses, in order of intended use. This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. Each algorithm is checked against the list of available client algorithm types until a match is found. If an algorithm is specified that is not installed on this side, the connection terminates with error message ORA-12650.
		Syntax	`SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (valid_ crypto_checksum_algorithm [,valid_crypto_ checksum_algorithm])`
		Values	■ SHA-1: Secure Hash Algorithm ■ MD5: Message Digest 5
		Default	If no algorithms are defined in the local `sqlnet.ora` file, all installed algorithms are used in a negotiation—in the above sequence.
	Client	Purpose	This parameter specifies a list of data integrity algorithms this client or server acting as a client uses. This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. If an algorithm that is not installed on this side is specified, the connection terminates with error message ORA-12650.
		Syntax	`SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (valid_ crypto_checksum_algorithm [,valid_crypto_ checksum_algorithm])`
		Values	■ SHA-1: Secure Hash Algorithm ■ MD5: Message Digest 5
		Default	If no algorithms are defined in the local `sqlnet.ora` file, all installed algorithms are used in a negotiation.

Seeding the Random Key Generator

```
SQLNET.CRYPTO_SEED = "10-70 random characters"
```

The characters that form the value for this parameter are used when generating cryptographic keys. The more random the characters entered into this field are, the stronger the keys are. You set this parameter by entering from 10 to 70 random characters into the above statement.

> **Note:** Oracle Corporation recommends that you enter as many characters as possible, up to 70, to make the resulting key more random and therefore stronger.

This parameter must be present in the `sqlnet.ora` file whenever data encryption or integrity is enabled.

B

Authentication Parameters

This appendix illustrates some sample configuration files with the necessary profile file (`sqlnet.ora`) and database initialization file (`init.ora`) authentication parameters, when using CyberSafe, Kerberos, RADIUS, or SSL authentication.

This appendix contains the following topics:

- Parameters for Clients and Servers using CyberSafe Authentication
- Parameters for Clients and Servers using Kerberos Authentication
- Parameters for Clients and Servers using RADIUS Authentication
- Parameters for Clients and Servers using SSL

Parameters for Clients and Servers using CyberSafe Authentication

Following is a list of parameters to insert into the configuration files for clients and servers using CyberSafe.

Table B–1   CyberSafe Configuration Parameters

File Name	Configuration Parameters

`sqlnet.ora`	`SQLNET.AUTHENTICATION_SERVICES=(cybersafe)`
	`SQLNET.AUTHENTICATION_GSSAPI_SERVICE=`
	`oracle/dbserver.someco.com@SOMECO.COM`
	`SQLNET.AUTHENITCATION_KERBEROS5_SERVICES=oracle`
	`SQLNET.KERBEROS5_CONF=/krb5/krb.conf`
	`SQLNET.KERBEROS5_REALMS=/krb5/krb.realms`
	`SQLNET.KERBEROS5_KEYTAB=/krb5/v5srvtab`

initialization parameter file (`init.ora`)	`REMOTE_OS_AUTHENT=FALSE` `OS_AUTHENT_PREFIX=""`

Parameters for Clients and Servers using Kerberos Authentication

Following is a list of parameters to insert into the configuration files for clients and servers using Kerberos.

Table B–2   Kerberos Authentication Parameters

File Name	Configuration Parameters

`sqlnet.ora`	`SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)`
	`SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=oracle`
	`SQLNET.KERBEROS5_CC_NAME=/usr/tmp/DCE-CC`
	`SQLNET.KERBEROS5_CLOCKSKEW=1200`
	`SQLNET.KERBEROS5_CONF=/krb5/krb.conf`
	`SQLNET.KERBEROS5_CONF_MIT=(FALSE)`
	`SQLNET.KERBEROS5_REALMS=/krb5/krb.realms`
	`SQLNET.KERBEROS5_KEYTAB=/krb5/v5srvtab`

`initialization parameter file (init.ora)`	`REMOTE_OS_AUTHENT=FALSE` `OS_AUTHENT_PREFIX=""`

Parameters for Clients and Servers using RADIUS Authentication

The following sections describe the parameters for RADIUS authentication

- sqlnet.ora File Parameters
- Minimum RADIUS Parameters
- Initialization File (init.ora) Parameters

sqlnet.ora File Parameters

SQLNET.AUTHENTICATION_SERVICES

Table B–3   SQLNET.AUTHENTICATION_SERVICES

Description	Configure the client or the server to use the RADIUS adapter: value = radius.
Default	None

SQLNET.RADIUS_AUTHENTICATION

Table B–4   SQLNET.RADIUS_AUTHENTICATION

Description	To set the location of the primary RADIUS server, either host name or dotted decimal format. If the RADIUS server is on a different machine from the Oracle server, you must specify either the host name or the IP address of that machine: format = *IP_address_of RADIUS_Server*.
Default	localhost

SQLNET.RADIUS_AUTHENTICATION_PORT

Table B–5   SQLNET.RADIUS_AUTHENTICATION_PORT

Description	To set the listening port of the primary RADIUS server.
Default	1645

SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

Table B–6   SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

Description	To set the time to wait for response.

Table B–6   SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

Default	5

SQLNET.RADIUS_AUTHENTICATION_RETRIES

Table B–7   SQLNET.RADIUS_AUTHENTICATION_RETRIES

Description	To set the number of times to re-send.
Default	3

SQLNET.RADIUS_SEND_ACCOUNTING

Table B–8   SQLNET.RADIUS_SEND_ACCOUNTING

Description	To set the turn accounting ON/OFF. If you enable accounting, packets will be sent to the active RADIUS server at listening port plus one. Default port is 1646. You need to turn this feature on only when your RADIUS server supports accounting and you want to keep track of the number of times the user is logging on to the system.
Default	OFF

SQLNET.RADIUS_SECRET

Table B–9   SQLNET.RADIUS_SECRET

Description	The file name and location of the RADIUS secret key.
Default	`$ORACLE_HOME/network/security/radius.key`

SQLNET.RADIUS_ALTERNATE

Table B–10   SQLNET.RADIUS_ALTERNATE

Description	To set the location of alternate RADIUS server to be used in case the primary server becomes unavailable. This feature is set to OFF by default. If you want to set up a second RADIUS server for fault tolerance, you need to specify the host name or the IP address of the host where the second RADIUS server is located.
Default	NONE

SQLNET.RADIUS_ALTERNATE_PORT

Table B–11   SQLNET.RADIUS_ALTERNATE_PORT

Description	To set the listening port for the alternate RADIUS server.
Default	1645

SQLNET.RADIUS_ALTERNATE_TIMEOUT

Table B–12   SQLNET.RADIUS_ALTERNATE_TIMEOUT

Description	To set the time to wait for response.
Default	5

SQLNET.RADIUS_ALTERNATE_RETRIES

Table B–13   SQLNET.RADIUS_ALTERNATE_RETRIES

Description	To set the number of times to re-send messages.
Default	3

SQLNET.RADIUS_CHALLENGE_RESPONSE

Table B–14   SQLNET.RADIUS_CHALLENGE_RESPONSE

Description	To turn challenge/response support ON/OFF.
Default	OFF

SQLNET.RADIUS_CHALLENGE_KEYWORD

Table B–15   SQLNET.RADIUS_CHALLENGE_KEYWORD

Description	To set the keyword to request a challenge from the RADIUS server. User types no password on client.
Default	challenge

SQLNET.RADIUS_AUTHENTICATION_INTERFACE

Table B–16   SQLNET.RADIUS_AUTHENTICATION_INTERFACE

Description	To set the name of the Java class that contains the graphical user interface when RADIUS is in the challenge-response (asynchronous) mode.
Default	DefaultRadiusInterface (oracle/net/radius/DefaultRadiusInterface)

SQLNET.RADIUS_CLASSPATH

Table B–17   SQLNET.RADIUS_CLASSPATH

Description	If you decide to use the challenge-response authentication mode, RADIUS presents the user with a Java-based graphical interface requesting first a password, then additional information—for example, a dynamic password that the user obtains from a token card. Add the SQLNET.RADIUS_ CLASSPATH parameter in the `sqlnet.ora` file to set the path for the Java classes for that graphical interface, and to set the path to the JDK Libjava.
Default	$ORACLE_HOME/jlib/netradius.jar:$ORACLE_ HOME/JRE/lib/sparc/native_threads

Minimum RADIUS Parameters

```
sqlnet.authentication_services = (radius)
sqlnet.authentication = IP-address-of-RADIUS-server
sqlnet.radius_challenge_response = ON
```

Initialization File (init.ora) Parameters

```
REMOTE_OS_AUTHENT=FALSE
OS_AUTHENT_PREFIX=""
```

Parameters for Clients and Servers using SSL

There are two ways to configure a parameter:

- Static: The name of the parameter that exists in the `sqlnet.ora` file.

- Dynamic: The name of the parameter used in the security subsection of the Oracle Net address.

Authentication Parameters

Table B–18   SSL Authentication Parameters

Parameter Name (static):	SQLNET.AUTHENTICATION_SERVICES
Parameter Name (dynamic):	AUTHENTICATION
Parameter Type:	String LIST
Parameter Class:	Static
Permitted Values:	Add TCPS to the list of available authentication services.
Default Value:	No default value.
Description:	To control which authentication services a user wants to use. Note: The dynamic version supports only the setting of one type.
Existing/New Parameter	Existing
Syntax (static):	`SQLNET.AUTHENTICATION_SERVICES = (TCPS, selected_method_1, selected_method_2)`
Example (static):	`SQLNET.AUTHENTICATION_SERVICES = (TCPS, cybersafe)`
Syntax (dynamic):	`AUTHENTICATION = string`
Example (dynamic):	`AUTHENTICATION = (TCPS)`

Cipher Suites

Table B–19   Cipher Suite Parameters

Parameter Name (static):	SSL_CIPHER_SUITES
Parameter Name (dynamic):	SSL_CIPHER_SUITES
Parameter Type:	String LIST
Parameter Class:	Static
Permitted Values:	Any known SSL cipher suite
Default Value:	No default
Description:	Controls the combination of encryption and data integrity used by SSL.
Existing/New Parameter	Existing
Syntax (static):	SSL_CIPHER_SUITES=(*SSL_cipher_suite1*[, *SSL_cipher_suite2*, ... *SSL_cipher_suiteN*])
Example (static):	SSL_CIPHER_SUITES=(SSL_DH_DSS_WITH_DES_CBC_SHA)
Syntax (dynamic):	SSL_CIPHER_SUITES=(*SSL_cipher_suite1* [, *SSL_cipher_suite2*, ...*SSL_cipher_suiteN*])
Example (dynamic):	SSL_CIPHER_SUITES=(SSL_DH_DSS_WITH_DES_CBC_SHA)

Supported SSL Cipher Suites

Oracle Advanced Security supports the following cipher suites:

- SSL_RSA_WITH_3DES_EDE_CBC_SHA

- SSL_RSA_WITH_RC4_128_SHA

- SSL_RSA_WITH_RC4_128_MD5

- SSL_RSA_WITH_DES_CBC_SHA

- SSL_DH_anon_WITH_3DES_EDE_CBC_SHA

- SSL_DH_anon_WITH_RC4_128_MD5

- SSL_DH_anon_WITH_DES_CBC_SHA

- SSL_RSA_EXPORT_WITH_RC4_40_MD5

- SSL_RSA_EXPORT_WITH_DES40_CBC_SHA

- SSL_DH_anon_EXPORT_WITH_RC4_40_MD5

- SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA

SSL Version

Table B–20   SSL Version Parameters

Parameter Name (static):	SSL_VERSION
Parameter Name (dynamic):	SSL_VERSION
Parameter Type:	string
Parameter Class:	Static
Permitted Values:	Any version which is valid to SSL. (0, 3.0)
Default Value:	"0"
Description:	To force the version of the SSL connection.
Existing/New Parameter	New
Syntax (static):	`SSL_VERSION=version`
Example (static):	`SSL_VERSION=3.0`
Syntax (dynamic):	`SSL_VERSION=version`
Example (dynamic):	`SSL_VERSION=3.0`

SSL Client Authentication

Table B–21   SSL Client Authentication Parameters

Parameter Name (static):	SSL_CLIENT_AUTHENTICATION
Parameter Name (dynamic):	SSL_CLIENT_AUTHENTICATION
Parameter Type:	Boolean
Parameter Class:	Static
Permitted Values:	TRUE/FALSE
Default Value:	TRUE

Table B–21   SSL Client Authentication Parameters

Description:	To control whether a client, in addition to the server, is authenticated using SSL.
Existing/New Parameter	New
Syntax (static):	`SSL_CLIENT_AUTHENTICATION={TRUE	FALSE}`
Example (static):	`SSL_CLIENT_AUTHENTICATION=FALSE`
Syntax (dynamic):	`SSL_CLIENT_AUTHENTICATION={TRUE	FALSE}`
Example (dynamic):	`SSL_CLIENT_AUTHENTICATION=FALSE`

Table B–22   SSL X.509 Server Match Parameters

Parameter Name	SSL_SERVER_DN_MATCH
Where stored	`sqlnet.ora`
Purpose	Use this parameter to force the server's **distinguished name (DN)** to match its service name. If you force the match verifications, SSL ensures that the certificate is from the server. If you choose to not enforce the match verification, SSL performs the check but permits the connection, regardless if there is a match. *Not forcing the match lets the server potentially fake its identity.*
Values	`yes|on|true`—Specify to enforce a match. If the DN matches the service name, the connection succeeds; otherwise, the connection fails.
	`no|off|false`—Specify to not enforce a match. If the DN does not match the service name, the connection is successful, but an error is logged to the `sqlnet.log` file.
Default	Oracle8*i* and Oracle9*i*.FALSE. SSL client (always) checks server DN. If it does not match the service name, the connection succeeds but an error is logged to `sqlnet.log` file.
Usage Notes	Additionally configure the tnsnames.ora parameter `SSL_SERVER_CERT_DN` to enable server DN matching.
Example	`SSL_SERVER_DN_MATCH=yes`
Parameter Name	SSL_SERVER_CERT_DN
Where stored	`tnsnames.ora`—Can be stored on the client, for every server it connects to, OR it can be stored in the LDAP directory, for every server it connects to, updated centrally.

Table B–22   SSL X.509 Server Match Parameters

Purpose	This parameter specifies the **distinguished name (DN)** of the server. The client uses this information to obtain the list of DNs it expects for each of the servers—to force the server's DN to match its service name.
Values	Set equal to **distinguished name (DN)** of the server.
Default	n/a
Usage Notes	Additionally configure the sqlnet.ora parameter `SSL_SERVER_DN_MATCH` to enable server DN matching.
Example	`dbalias=(description=address_list=(address=(protocol=tcps)(host=hostname)(port=portnum)))(connect_data=(sid=Finance))(security=(SSL_SERVER_DN="CN=Finance,CN=OracleContext,C=US,O=Acme"))`

Wallet Location

For any application that must access a wallet for loading the security credentials into the process space, you must specify the wallet location parameters defined by Table B–23 in each of the following configuration files:

- `sqlnet.ora`
- `listener.ora`

Table B–23   Wallet Location Parameters

Static Configuration	Dynamic Configuration

`WALLET_LOCATION =`
`(SOURCE=`
`  (METHOD=File)`
`  (METHOD_DATA=`
`    (DIRECTORY=your wallet location)`
`      )`
`  )`
	`MY_WALLET_DIRECTORY`
	`= your_wallet_dir`

The default wallet location is the `$ORACLE_HOME` directory.

C

Integrating Authentication Devices Using RADIUS

This appendix describes how third party authentication vendors customize the RADIUS challenge-response user interface to fit their particular device.

This appendix contains the following topics:

- About the RADIUS Challenge-Response User Interface
- Customizing the RADIUS Challenge-Response User Interface

> **See Also:** Chapter 4, Configuring RADIUS Authentication

About the RADIUS Challenge-Response User Interface

You can set up any authentication device that supports the RADIUS standard to authenticate Oracle users. When your authentication device uses the challenge-response mode, a graphical interface prompts the user first for a password, then for additional information—for example, a dynamic password that the user obtains from a token card. This interface is Java-based to provide optimal platform independence.

Third party vendors of authentication devices must customize this graphical user interface to fit their particular device. For example, a smart card vendor customizes the Oracle client to issue the challenge to the smart card reader. Then, when the smart card receives a challenge, it responds by prompting the user for more information, such as a PIN.

Customizing the RADIUS Challenge-Response User Interface

You can customize this interface by creating your own class to support the functionality described in Table C–1. You can then open the `sqlnet.ora` file, look up the SQLNET.RADIUS_AUTHENTICATION_INTERFACE parameter, and replace the name of the class listed there (`DefaultRadiusInterface`), with the name of the new class you have just created. When you make this change in the `sqlnet.ora` file, the class is loaded on the Oracle client in order to handle the authentication process.

The third party must implement the Oracle RADIUS Interface, which is located in the ORACLE.NET.RADIUS package.

```
public interface OracleRadiusInterface {
  public void radiusRequest();
  public void radiusChallenge(String challenge);
  public String getUserName();
  public String getPassword();
```

Table C–1   Server Encryption Level Setting

Parameter	Description
radiusRequest	Generally, this prompts the user for a user name and password which will later be retrieved through getUserName and getPassword.
getUserName	Extracts the user name the user enters. If this method returns an empty string, it is assumed that the user wants to cancel the operation. The user then receives a message indicating that the authentication attempt failed.
getPassword	Extracts the password the user enters. If getUserName returns a valid string, but getPassword returns an empty string, the challenge keyword is replaced as the password by the database. If the user enters a valid password, a challenge may or may not be returned by the RADIUS server.
radiusChallenge	Presents a request sent from the RADIUS server for the user to respond to the server's challenge.
getResponse	Extracts the response the user enters. If this method returns a valid response, that information then populates the User-Password attribute in the new Access-Request packet. If an empty string is returned, the operation is aborted from both sides by returning the corresponding value.

D

Oracle Advanced Security FIPS 140-1 Settings

Oracle Advanced Security Release 8.1.6 has been validated under U.S. Federal Information Processing Standard (FIPS) 140-1 at the Level 2 security level. This appendix describes the formal configuration required for Oracle Advanced Security to comply with the FIPS 140-1 standard. Refer to the NIST Cryptographic Modules Validation list at the following web site address:

`http://csrc.nist.gov/cryptval/140-1/1401val.htm`

This appendix contains the following topics:

- Configuration Parameters
- Post Installation Checks
- Status Information
- Physical Security

> **Note:** The information contained in this appendix should be used with the information provided in Appendix A, Data Encryption and Integrity Parameters.

Configuration Parameters

This appendix contains information on the Oracle Advanced Security parameters required in the `sqlnet.ora` files that ensure that any connections created between a client and server are encrypted under the control of the server.

Configuration parameters are contained in the `sqlnet.ora` file that is held locally for each of the client and server processes. The protection placed on these files should be equivalent to the level of a DBA.

The following configuration parameters are described in this appendix:

- ENCRYPTION_SERVER
- ENCRYPTION_CLIENT
- ENCRYPTION_TYPES_SERVER
- CRYPTO_SEED
- CRYPTO_SEED_CLIENT
- FIPS_140

Server Encryption Level Setting

The server side of the negotiation notionally controls the connection settings. The following parameter in the server file is mandatory:

```
SQLNET.ENCRYPTION_SERVER=REQUIRED
```

Setting the encryption as REQUIRED on the server side of the connection ensures that a connection is only permitted if encryption is used, irrespective of the parameter value on the client.

Client Encryption Level Setting

The `ENCRYPTION_CLIENT` parameter specifies the connection behavior for the client. One of the following parameter settings in the client file is mandatory:

```
SQLNET.ENCRYPTION_CLIENT=(ACCEPTED|REQUESTED|REQUIRED)
```

A connection to the server is only possible if there is agreement between client and server for the connection encryption. The server has this set to REQUIRED, therefore the client must not reject encryption for a valid connection to be the result. Failure to specify one of these values results in error when attempting to connect to a FIPS 140-1 compliant server.

Server Encryption Selection List

The `ENCRYPTION_TYPES_SERVER` parameter specifies a list of encryption algorithms that the server is permitted to use when acting as a server in the order of required usage. The specified algorithm must be installed or the connection terminates. For FIPS 140-1 compliance, only DES encryption is permitted and therefore the following parameter setting is mandatory:

```
SQLNET.ENCRYPTION_TYPES_SERVER=(DES|DES40)
```

Client Encryption Selection List

The `ENCRYPTION_TYPES_CLIENT` parameter specifies the list of encryption algorithms which the client is prepared to use for the connection with the server. In order for a connection to be successful, the algorithm must first be installed and the encryption type must be mutually acceptable to the server.

To create a connection with a server that is configured for FIPS 140-1, the following parameter setting is mandatory:

```
SQLNET.ENCRYPTION_TYPES_CLIENT=(DES|DES40)
```

Cryptographic Seed Value

The `CRYPTO_SEED` parameter contains characters which are part of the seed for the random number generator. There are no explicit requirements for the value of this parameter within the FIPS 140-1 standard, however it is suggested that a large set of random characters, up to 70, is chosen as follows:

```
SQLNET.CRYPTO_SEED=10_to_70_random_characters
```

FIPS Parameter

The default setting of the `FIPS_140` parameter is FALSE. Setting the parameter to TRUE is mandatory for both client and server to ensure Oracle Advanced Security complies with the standards defined in FIPS 140-1 as follows:

```
SQLNET.FIPS_140=TRUE
```

> **Note:** Use a text editor to set the `FIPS_140` parameter in the `sqlnet.ora` file. You cannot use Oracle Net Manager to set this parameter.

Post Installation Checks

After the installation, the following permissions must be verified in the operating system:

- Execute permissions must be set on all Oracle Advanced Security executable files so as to prevent execution of Oracle Advanced Security by users who are unauthorized to do so in accordance with the system security policy.

- Read and write permissions must be set on all executable files so as to prevent accidental or deliberate reading or modification of Oracle Advanced Security files by any user.

To comply with FIPS 140-1 Level 2 requirements, the security policy must include procedures to prevent unauthorized users from reading or modifying executing Oracle Advanced Security processes and the memory they are using in the operating system.

Status Information

Status information for Oracle Advanced Security is available after the connection has been established. The information is contained in the RDBMS virtual table v$session_connect_info.

Running the query `SELECT * from V$SESSION_CONNECT_INFO` displays all of the product banner information for the active connection. Table D–1 shows an example of a connection configuration where both DES encryption and MD5 data integrity is defined:

Table D–1   Sample Output from v$session_connect_info

SID	AUTHENTICATION	OSUSER	NETWORK_SERVICE_BANNER
7	DATABASE	oracle	Oracle Bequeath OS adapter for Solaris, v8.1.6.0.0
7	DATABASE	oracle	Oracle Advanced Security: encryption service for Solaris
7	DATABASE	oracle	Oracle Advanced Security: DES encryption service adapter
7	DATABASE	oracle	Oracle Advanced Security: crypto-checksumming service
7	DATABASE	oracle	Oracle Advanced Security: MD5 crypto-checksumming service adapter.

Physical Security

To comply with FIPS 140-1 Level 2 requirements, tamper-evident seals must be applied to the cover of each machine—to ensure that removal of the cover is detectable.

E

Oracle Implementation of Java SSL

This appendix describes the Oracle implementation of Java Secure Socket Extension (JSSE), in the following sections:

- Prerequisites
- Oracle Java SSL Features
- Oracle Java SSL Examples
- Typical Errors
- Oracle Java SSL API

> **Assumption:** This appendix assumes that you are familiar with the basic principles of Java socket programming and the SSL protocol.

> **See Also:** Java documentation at the following web site, for further information about Java SSL packages:
>
> `http://java.sun.com/products/jsse`

Prerequisites

To use the Oracle Java SSL implementation, JDK version 1.1 or 1.2 must be installed, and the CLASSPATH environment variable must include the following jar files:

- For JDK1.1: `javax-ssl-1_1.jar,  jssl-1_1.jar`

- For JDK1.2: `javax-ssl-1_2.jar,  jssl-1_2.jar`

In addition, the Java SSL shared library must be added to the shared library path:

- **For Solaris:** `libnjssl8.so` must be included in the library path specified by the LD_LIBRARY_PATH environment variable.

- **For Windows NT:** `njssl8.dll` must be included in the path specified by the PATH environment variable.

> **See Also:**   Platform-specific documentation.

Oracle Java SSL Features

Oracle Java SSL is a commercial-grade implementation of Java Secure Socket Extension (JSSE). In order to create a secure, fast implementation of SSL, Oracle Java SSL uses native code to improve the performance.

In addition to the functionality included in the JSSE specifications, Oracle Java SSL supports the following:

- Multiple cryptographic algorithms

- Certificate and key management using Oracle Wallet Manager

- SSL-specific session capabilities, including authentication, that can be used by applications built on top of Oracle Java SSL

Oracle Java SSL features are described in further detail in the following sections:

- SSL Cipher Suites Supported by Oracle Java SSL

- Certificate and Key Management with Oracle Wallet Manager

- Security-Aware Applications Support

SSL Cipher Suites Supported by Oracle Java SSL

Before data can flow through an SSL connection, both sides of the connection must negotiate common algorithms to be used for data transmission. A set of such algorithms combined to provide a mix of security features is called a **cipher suite**. Selecting a particular cipher suite lets the participants in an SSL connection establish the appropriate level for their communications.

Oracle Java SSL supports cipher suites with the following options:

- Key exchange of 512, 768, or 1024 bit asymmetric keys using the following algorithms:

  - RSA

  - Diffie-Hellman

- NULL encryption, or symmetric key encryption with 40 and 128 bit symmetric keys using the following algorithms:

  - RC4 stream cipher

  - DES, DES40, and 3DES-EDE, in **Cipher Block Chaining (CBC)** mode

> **Note:** With NULL encryption, SSL is only used for authentication and data integrity purposes.

- Message Authentication Code using MD5 or SHA1 data integrity.

Certificate and Key Management with Oracle Wallet Manager

You can use Oracle Wallet Manager to generate **public/private key pair**s and certificate requests. A signed certificate request and the appropriate trusted certificates must be added to produce a complete Oracle wallet.

You can export a complete wallet with a certificate in *Ready* status, in a BASE64-formatted file, using the menu option *Operation ->ExportWallet*. This file can be used to add SSL credentials in a Java SSL-based program.

If you are not using Oracle Wallet Manager, you can manually add individual components to a file:

- Add the **certificate** first, followed by the **private key**.
- Add the **certificate authority** certificate and other **trusted certificate**s subsequently.

See Also:

- Public Class: OracleSSLCredential on page E-19, for information about setting the credentials in Java SSL
- Chapter 16, Using Oracle Wallet Manager

Security-Aware Applications Support

Some security-aware application do not set **trust point**s. In order to let these applications perform their own validation, Oracle Java SSL lets handshakes complete if no security credentials are set—if a complete **certificate chain** is sent by the peer. This feature is useful when there is a large number of trust points stored in a database, and the application is constrained from passing all of them to the SSL layer.

Once the handshake is complete, it is possible to obtain the peer certificate chain and extract individual peer certificates. These certificates can be used for

application-specific validation, such as matching the certificate's **distinguished name (DN)** against a user database.

Security-unaware applications that need the trust point check must ensure that trust points are set in the application.

> **See Also:** Public Class: OracleSSLCredential on page E-19, for more information about checking peer credentials

Oracle Java SSL Examples

The examples in this section illustrate the use of Oracle Java SSL. For purposes of the examples, we created a model server and client named *SSLServerExample* and *SSLClientExample,* respectively. Together, they demonstrate some common features of Oracle Java SSL, as well as the basics of socket communication. In addition, SSLProxyClientExample demonstrates one of the possible ways to implement firewall tunnelling connections.

We present the complete code for each program, and discuss some of its more important sections.

This example does not cover every feature available in Oracle Java SSL. For more detailed information about different security options available in this package please consult the latter sections of this chapter.

> **See Also:**
> - Public Class: OracleSSLServerSocketFactoryImpl on page E-21, for general information about socket programming.
> - Java documentation for the *java.net* package, for information about sockets and socket streams.

Oracle Java SSL examples are described in the following sections:

- SSLServerExample Program
- SSLClientExample Program
- SSLProxyClientExample Program

SSLServerExample Program

SSLServerExample is a simple SSL server. It uses a wallet exported from Oracle Wallet Manager to set up its security credentials. When the server is started it waits for a client to initiate a connection. After the SSL handshake is complete, the server sends a short message to the client and closes the connection.

```
import oracle.security.ssl.*;
import java.net.*;
import java.io.*;
import java.util.*;
import javax.net.*;
import javax.net.ssl.*;
```

```
public class SSLServerExample
{
    private OracleSSLServerSocketFactoryImpl _socketFactory;
    private OracleSSLCredential _credential;
    private SSLServerSocket _svrSoc;

    private void initCredential(String wltPath, String password)
        throws java.io.IOException
    {
        _credential = new OracleSSLCredential();
        _credential.setWallet(wltPath, password);
    }

    private void initSocketFactory()
        throws javax.net.ssl.SSLException
    {
        _socketFactory
            = (OracleSSLServerSocketFactory)SSLServerSocketFactory.getDefault();
        _socketFactory.setSSLProtocolVersion(
            OracleSSLProtocolVersion.SSL_Version_3_0_With_2_0_Hello);
        _socketFactory.setSSLCredentials(_credential);
    }

    private void initServerSocket(int port)
        throws java.io.IOException
    {
        _svrSoc = (SSLServerSocket)_socketFactory.createServerSocket(port);
        _svrSoc.setUseClientMode(false);
        _svrSoc.setNeedClientAuth(false);
        _svrSoc.setEnabledCipherSuites(new String[]{"SSL_RSA_WITH_RC4_128_SHA",
            "SSL_RSA_WITH_RC4_128_MD5"});
    }

    public SSLServerExample(String wltPath, String password, int port)
        throws java.io.IOException, javax.net.ssl.SSLException
    {
        initCredential(wltPath, password);
        initSocketFactory();
        initServerSocket(port);
    }

    public void runServer()
    {
        String message = "Hello! Current Server Time is " + new Date() + "\n";
        Socket csocket = null;
```

```
                OutputStreamWriter out = null;
                try
                {
                    csocket = _svrSoc.accept();
                    out = new OutputStreamWriter(csocket.getOutputStream());
                    out.write(message);
                    System.out.println("Connection Succeeded");
                }
                catch(IOException e)
                {
                    System.out.println("Connection Failed");
                    e.printStackTrace();
                }
                finally
                {
                    try
                    {
                        if(out != null)
                            out.close();
                        if(csocket != null)
                            csocket.close();
                        _svrSoc.close();
                    }
                    catch(IOException e){}
                }
            }

            public static void main(String[] argv)
            {
                System.getProperties().put("SSLServerSocketFactoryImplClass",
                    "oracle.security.ssl.OracleSSLServerSocketFactoryImpl");
                try
                {
                    SSLServerExample myServer = new SSLServerExample("mywallet.txt",
                        "welcome1", 19978);
                    myServer.runServer();
                }
                catch(IOException i)
                {
                    System.out.println("Failied to start up server");
                    i.printStackTrace();
                }
            }
        }
```

Initializing the Credentials:

SSLServerExample uses a wallet created by Oracle Wallet Manager, so the job of setting up the credential object is quite easy. In `initCredential()` we call

```
_credential = new OracleSSLCredential();
_credential.setWallet(wltPath, password);
```

to read the wallet located at `wltPath`. The **private key**, user **certificate**, **certificate** and **trust points** located in the wallet are used in the connection. An IOException is thrown if an error occurs while accessing the wallet.

If you do not elect to use the wallet, you can install the necessary security credentials manually.

> **See Also:** Public Class: OracleSSLCredential on page E-19 for more information about:

- `addTrustedCert()`
- `addCertChain()`
- `setPrivateKey()`

Initializing the Socket Factory:

To create SSL sockets, you must access the proper socket factory. For Oracle Java SSL, `oracle.security.ssl.OracleSSLSocketFactoryImpl` is the name of the class that implements `javax.net.ServerSocketFactory`. In order to make sure we access it correctly we must set up the System Properties in the `main()` function using

```
System.getProperties().put("SSLServerSocketFactoryImplClass","oracle.security.ss
l.OracleSSLServerSocketFactoryImpl");
```

Once the system properties are set, we can obtain an instance of the socket factory and customize it. In `initSocketFactory()` we specify the SSL protocol the sockets created by this factory are to support, and install the security credentials to be used by all sockets created by this factory.

Initializing Server Socket:

The method `initServerSocket()` uses the socket factory to create a new server socket that listens in server mode on the specified port:

```
_svrSoc = (SSLServerSocket)_socketFactory.createServerSocket(port);
_svrSoc.setUseClientMode(false);
```

Once the socket is created, we can change some of its attributes:

```
_svrSoc.setNeedClientAuth(false);
_svrSoc.setEnabledCipherSuites(new String[]{"SSL_RSA_WITH_RC4_128_SHA"
"SSL_RSA_WITH_RC4_128_MD5"});
```

For this example we do not require the clients to authenticate themselves to the server. However, instead of using the default enabled cipher suites, we only let those clients connect that support RSA_WITH_RC4_128_SHA, SSL_RSA_WITH_ RC4_128_MD5 cipher suites.

Use `OracleSSLServerSocketFactory.getSupportedCipherSuites()` to determine which cipher suites are supported by Java SSL.

> **See Also:**
>
> - SSL Cipher Suites Supported by Oracle Java SSL on page E-3
> - Public Class: OracleSSLServerSocketFactoryImpl on page E-21

Waiting for the Connection and Sending Data:

SSLServerExample waits until the client connects to the server. Notice that the method `accept()` blocks until a connection is established. Once the client connects we obtain the output stream for the socket by calling `getOutputStream()`. We use this output stream to send information to the client. When the server has no more data to send to the client, the server closes the corresponding output stream and socket. In order to stop accepting connections, the server must close the corresponding server socket. The server closes the ServerSocket when it cannot accept any further connections.

> **See Also:** Java documentation about their *java.net* package, for information about sockets and socket streams.

SSLClientExample Program

The SSLClientExample is a simple program (JDK1.1) used to connect to the SSLServerExample program. Notice that the initilization of the SSLClientExample is very similar to that of the server. However, certain differences have been included in this example to demonstrate some of the features of JavaSSL. The explanations focus on these differences whenever appropriate:

```
import oracle.security.ssl.*;
import java.net.*;
import java.io.*;
```

```
import java.util.*;
import javax.net.*;
import javax.net.ssl.*;
import javax.security.cert.*;

public class SSLClientExample
{
    protected OracleSSLSocketFactoryImpl _socketFactory;
    private OracleSSLCredential _credential;
    protected SSLSocket _socket;

    private void initCredential(String wltPath, String password)
        throws java.io.IOException
    {
        _credential = new OracleSSLCredential();
        _credential.setWallet(wltPath, password);
    }

    private void initSocketFactory()
        throws javax.net.ssl.SSLException
    {
        _socketFactory
            = (OracleSSLSocketFactoryImpl)SSLSocketFactory.getDefault();
        _socketFactory.setSSLProtocolVersion(
            OracleSSLProtocolVersion.SSL_Version_3_0);
        _socketFactory.setSSLCredentials(_credential);
    }

    private void initSocket(String host, int port)
        throws java.io.IOException
    {
        _socket = (SSLSocket)_socketFactory.createSocket(host, port);
        _socket.setUseClientMode(true);
    }

    public SSLClientExample(String wltPath, String pass, String host, int port)
        throws java.io.IOException, javax.net.ssl.SSLException
    {
        initCredential(wltPath, pass);
        initSocketFactory();
        initSocket(host, port);
    }

    public void connectSocket()
    {
```

```
            try
            {
                _socket.startHandshake();
                getData();
            }
            catch(IOException e)
            {
                System.out.println("Connection Failed");
                e.printStackTrace();
            }
            finally
            {
                try
                {
                    _socket.close();
                }
                catch(IOException e){}
            }
        }

        public void getData()
        {
            InputStreamReader in = null;
            try
            {
                int ch;
                SSLSession session = _socket.getSession();

                System.out.println("Negotiated Cipher Suite " +
                    session.getCipherSuite());
                X509Certificate[] peerCerts = session.getPeerCertificateChain();
                for(int i = 0; i < peerCerts.length; i++)
                {
                    System.out.println(peerCerts[i]);
                }
                System.out.println("Server Response:");
                in = new InputStreamReader(_socket.getInputStream());
                ch =  in.read();
                while((char)ch != '\n')
                {
                    if(ch != -1)
                    System.out.print((char)ch);
                    ch=in.read();
                }
                System.out.println();
```

```
        }
        catch(IOException e)
        {
            System.out.println("Connection Failed");
            e.printStackTrace();
        }
        finally
        {
            try
            {
                if(in != null)
                in.close();
            }
            catch(IOException e){}
        }
    }

    public static void main(String[] argv)
    {
        System.getProperties().put("SSLSocketFactoryImplClass",
            "oracle.security.ssl.OracleSSLSocketFactoryImpl");
        try
        {
            SSLClientExample myClient = new
            SSLClientExample("mywallet.txt","welcome1","localhost", 19978);
            myClient.connectSocket();
        }
        catch(IOException i)
        {
            System.out.println("Failied to start up client");
            i.printStackTrace();
        }
    }
}
```

> **Note:** For JDK1.2, change `import javax.security.cert.*;` to `import java.security.cert.*;`

Initializing the Credentials:

The client intializes the credentials in the same way as the server. For purposes of the example, the client and the server use the same wallet. However, in real life

applications the client and the server must have different security credentials. In order for an SSL connection to complete successfully it is important that the proper **trusted certificates** are present in the wallets.

> **See Also:** Chapter 16, Using Oracle Wallet Manager, for more information about trusted certificates

Initializing the Socket Factory:

The socket factory class used to create client sockets is similar to the one used by the server. Once again it is necessary to set the system properties in order to obtain the correct socket factory before configuring it in `initSocketFactory()`. The correct socket factory is set in `main()` using:

```
System.getProperties().put("SSLSocketFactoryImplClass",
"oracle.security.ssl.OracleSSLSocketFactoryImpl");
```

Initializing and Connecting the Client Socket:

Client sockets are created by the socket factory just as server sockets are created by the server socket factory. However, in order to connect the client socket to a specific server, we supply the server's name and the port number at creation. In addition, we ensure that the socket connects in client mode:

```
_socket = (SSLSocket)_socketFactory.createSocket(host, port);
_socket.setUseClientMode(true);
```

Once the socket is created it can connect to the server using:

```
_socket.startHandshake();
```

Viewing Peer Credentials:

After the socket connects to the server, information about the connection can be accessed. The information is stored in the OracleSSLSession class, an instance of which can be obtained using `_socket.getSession()`.

In our program we print out the cipher suite negotiated between the client and the server as well as the security credentials of the server. This information can be used by security-aware applications to determine whether it should trust the connection. For example, most browsers check to confirm that the common name in the server certificate matches the URL that was accessed, and they display a warning if it does not. However, this check is not required by the SSL protocol.

Receiving Data:

Receiving and sending data through an SSL Socket is no different than receiving data through any other socket. In this example we access the socket's input stream, and proceed to read until an end-of-line character occurs.

> **See Also:** Java documentation about their *java.net* package, for information about sockets and socket streams.

SSLProxyClientExample Program

SSLProxyClientExample uses firewall tunneling to establish a secure connection to the server. Please note that this program might not work for all firewalls. For example, some firewalls do not permit a connection to non-standard ports, such as port 19978 that is used here. In this case you have to set up a secure server on port 443 and modify the client appropriately.

```java
import oracle.security.ssl.*;
import java.net.*;
import java.io.*;
import java.util.*;
import javax.net.*;
import javax.net.ssl.*;
import javax.security.cert.*;

public class SSLProxyClientExample extends SSLClientExample
{
    private String _proxyName;
    private int _proxyPort;

    protected void initSocket(String host, int port)
        throws java.io.IOException
    {
        final String connString = "CONNECT" + host + ":" + port +
            " HTTP/1.0 \n" + "User-Agent: Oracle Proxy Enabled SSL Socket\n\n";
        Socket normalSocket = new Socket(_proxyName, _proxyPort);
        OutputStreamWriter out
            = new OutputStreamWriter(normalSocket.getOutputStream());
        out.write(connString, 0, connString.length());
        _socket = (SSLSocket)_socketFactory.createSocket(normalSocket);
    }

    public SSLProxyClientExample(String wltPath, String password,String host,
                                 int port, String proxyName, int proxyPort)
        throws java.io.IOException, javax.net.ssl.SSLException
```

```
    {
        super(wltPath, password, host, port);
        _proxyName = proxyName;
        _proxyPort = proxyPort;
    }

    public static void main(String[] argv)
    {
        System.getProperties().put("SSLSocketFactoryImplClass",
        "oracle.security.ssl.OracleSSLSocketFactory");
        try{
            SSLClientExample myClient
                = new SSLProxyClientExample("mywallet.txt", "welcome1",
                "localhost", 19978,   "www-proxy", 80);
            myClient.connectSocket();
        }
        catch(IOException i)
        {
            System.out.println("Failied to start up client");
            i.printStackTrace();
        }
    }
}
```

Note: For JDK1.2, change `import javax.security.cert.*;` to `import java.security.cert.*;`

Initializing and Connecting the Client Socket:

The only significant difference between SSLProxyClientExample and its superclass, SSLClientExample, lies in the method `initSocket()`. In order to set up a tunnelling connection it is necessary to create a plain socket. This socket is used to send a special message, `connString`, to the firewall, setting up the connection to the actual server. Once this connection is set up, we can use the plain socket to initialize an SSL Socket using:

`_socketFactory.createSocket(normalSocket)`

Typical Errors

This section describes some typical Java SSL errors.

SSLException X509CertExpiredErr

Problem

During the handshake the program fails with SSLException and message `X509CertExpiredErr`. The program worked yesterday, and no changes were made.

Solution

Your user certificate has expired. You must obtain a new user certificate.

> **See Also:** Chapter 16, Using Oracle Wallet Manager

SSLException X509CertChainInvalidErr

Problem

The handshake fails on the client side with SSLException and message `X509CertChainInvalidErr`. A web browser can connect to the server successfully.

Solution

Either your server or your client does not have the proper credentials. If the client program sets trusted certificates, you must ensure that the list includes at least one of the certificates in the server's certificate chain. In addition you must ensure that the server sends the complete certificate chain to the client—as Java SSL cannot build the certificate chain itself. If you are using an Apache server, you must set the variables `SSLCertificateChainFile` and `SSLCertificateFile` correctly. This is especially important if the client program does not set trusted certificates.

For more information consult your web server documentation.

Client Connection with No Credentials

Problem

Server lets client connect even though no `OracleSSLCredentials` are set in the client program.

Solution

In order to enable security-aware applications to perform their own validation, Java SSL permits a connection if no credentials are set by the client, if the server sends a complete certificate chain. To avoid this behavior, your application must set at least one trusted certificate.

> **See Also:** Public Class: OracleSSLCredential on page E-19

Oracle Java SSL API

This section describes the public classes and interfaces used in Oracle Java SSL. Since Oracle Java SSL is an implementation of JSSE, only the Oracle additions to the JSSE package are described.

This section describes the following Oracle Java SSL classes and interfaces:

- Public Class: OracleSSLCredential
- Public Interface: OracleSSLProtocolVersion
- Public Class: OracleSSLServerSocketFactoryImpl
- Public Class: OracleSSLSession
- Public Class: OracleSSLSocketFactoryImpl

> **See Also:**
>
> `http://java.sun.com/products/jsse/doc/apidoc/index` `.html`, for a description of JSSE classes.

Public Class: OracleSSLCredential

This public class extends java.lang.Object.

Credentials are used to authenticate the server and the client to each other. OracleSSLCredential is used to load user certificates, trusted certificates (trust points), and private keys from base64 or *der* encoded certificates.

Constructor

*public **OracleSSLCredential**()*

Creates an empty OracleSSCredential. An empty credential lets the socket connect to any peer that sends a complete certificate chain during the handshake.

Methods

*public void **addTrustedCert**(java.lang.String b64TrustedCert)*

Adds a trusted certificate to the credential.

Parameters: b64TrustedCert - A Base64 encoded X509 certificate.

*public void **addTrustedCert**(byte[] trustedCert)*

Adds a trusted certificate to the credential.

Parameters: `trustedCert` - A der encoded X509 trusted certificate.

public void **setPrivateKey***(java.lang.String b64PvtKey,*
   java.lang.String password)

Adds a private key to the credential.

Parameters: `b64PvtKey` - A Base64 encoded X509 Private Key

   `password` - The password needed to decipher the private key.

public void **setPrivateKey***(byte[] pvtKey,*
   java.lang.String password)

Adds a private key to the credential.

Parameters: `b64PvtKey` - A der encoded X509 Private Key

   `password` - The password needed to decipher the private key.

public void **addCertChain***(java.lang.String b64certChainCert)*

Adds a certificate to the certificate chain. The certificate chain is sent along with the user certificate during the SSL handshake. It is used by the peer to verify the user certificate. The first certificate added to the certificate chain must be the Root CA certificate. Each subsequent certificate added must be signed by its immediate predecessor.

Parameters: `b64certChainCert` - A Base64 encoded X509 certificate.

public void **addCertChain***(byte[] certChainCert)*

Adds a certificate to the certificate chain.

Parameters: `certChainCert` - A der encoded X509 certificate.

public void **setWallet***(java.lang.String wltPath,*
   java.lang.String password) throws java.io.IOException

If Oracle Wallet Manager is used to create a wallet, the wallet can be exported in text format and used by JavaSSL. The text file must contain the user certificate, followed by the private key, the certificate chain, and any other trusted certificates. The method throws a `java.io.IOException` if the wallet cannot be opened.

Parameters: `wltPath` - The pathname of the wallet

           `password` - The password needed to decrypt the private key

Public Interface: OracleSSLProtocolVersion

This interface defines the available SSL protocol versions.

Fields

public static final int ***SSL_Version_Undetermined***

SSL protocol version undetermined.

public static final int ***SSL_Version_3_0_With_2_0_Hello***

SSL protocol version 3.0 with 2.0 hello.

public static final int ***SSL_Version_3_0_Only***

SSL protocol version 3.0 only.

public static final int ***SSL_Version_2_0***

SSL protocol version 2.0

public static final int ***SSL_Version_3_0***

SSL protocol version 3.0.

Public Class: OracleSSLServerSocketFactoryImpl

This public class extends javax.net.ssl.SSLServerSocketFactory; it is used to create SSL server sockets.

This class implements `javax.net.ssl.SSLServerSocketFactory` methods that are needed to create server sockets. In addition it provides extra methods, described bellow, that are necessary to configure options specific to Oracle Java SSL.

Constructor

*public **OracleSSLServerSocketFactoryImpl**()*

Creates a socket factory that may be used to create sockets. However, setting the system property `SSLServerSocketFactoryImplClass` to `oracle.security.sslOracleSSLServerSocketFactoryImpl` is the preferred method for creating socket factories. For example:

```
System.getProperties().put("SSLServerSocketFactoryImplClass",
        "oracle.security.ssl.OracleSSLServerSocketFactoryImpl");
SSLServerSocketFactory factory = OracleSSLServerSocketFactoryImpl.getDefault();
```

Methods

*public void **setSSLCredentials**(OracleSSLCredential sslCredential)throws javax.net.ssl.SSLException*

Sets the OracleSSLCredential (holding private keys, certificate chains, and similar data) that is to be used for the SSL connection. The method throws a `javax.net.ssl.SSLSocketException` if an error occurs.

*public void **setSSLProtocolVersion**(int version) throws javax.net.ssl.SSLException*

Sets the SSL protocol version. The method throws a `javax.net.ssl.SSLSocketException` if the SSL version is not supported.

Public Class: OracleSSLSession

This public class extends java.lang.Object; implements javax.net.ssl.SSLSession.

This class implements most methods specified in `javax.net.ssl.SSLSession`. However, the following methods have not been implemented: `getPeerHost()`, `getValue()`, `invalidate()`, `removeValue()`, and `getValueNames()`. This class provides extra methods, described bellow, that are specific to Oracle Java SSL.

Methods

*public byte[][] **getPeerRawCertificateChain()***
    throws javax.net.ssl.SSLPeerUnverifiedException

Returns the **certificate chain** presented by the peer as and array of peer **X.509** certificates in der format. The peer's certificate is first in the chain, and the *root* CA last. The method throws a javax.net.ssl.SSLPeerUnverifiedException if the peer certificate could not be verified

*public java.lang.String **getNegotiatedProtocolVersion()***

Returns the SSL protocol version used for this session.

Public Class: OracleSSLSocketFactoryImpl

This public class extends javax.net.ssl.SSLSocketFactory.

This class implements javax.net.ssl.SSLSocketFactory methods that are needed to create server sockets. In addition it provides extra methods, described below, that are necessary to configure options specific to Oracle Java SSL.

Constructor

*public **OracleSSLSocketFactoryImpl()***

Creates a socket factory that may be used to create sockets. However, setting the system property SSLSocketFactoryImplClass to oracle.security.sslOracleSSLSocketFactoryImpl is the preferred method for creating socket factories. For example:

```
System.getProperties().put("SSLSocketFactoryImplClass",
        "oracle.security.ssl.OracleSSLSocketFactoryImpl");
SSLSocketFactory factory = OracleSSLSocketFactoryImpl.getDefault();
```

Methods

public java.net.Socket createSocket(java.net.Socket socket)
    throws java.io.IOException

Returns a new instance of an SSLSocket that will read and write using an existing socket. This is particularly useful when tunneling through firewalls.

The method throws a `java.io.IOException` if an error occurs while creating the socket.

Parameters: `socket` - a socket object through which data will be transferred

```
public void setSSLCredentials(OracleSSLCredential
    sslCredential)throws javax.net.ssl.SSLException
```

Sets the OracleSSLCredential (holding private keys, certificate chains, and similar data) that is to be used for the SSL connection. The method throws a `javax.net.ssl.SSLSocketException` if an error occurs.

```
public void setSSLProtocolVersion(int version)
    throws javax.net.ssl.SSLException
```

Sets the SSL protocol version. The method throws a `javax.net.ssl.SSLSocketException` if the SSL version is not supported.

F

Abbreviations and Acronyms

This appendix defines abbreviations and acronyms used in the Oracle Advanced Security Administrator's Guide (Table F–1):

Table F–1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
3DES	A version of the DES encryption algorithm that provides triple-encryption; see Triple-DES.
ACL	Access Control List
CA	Certificate Authority
CBC	Cipher-Block-Chaining
CDS	Cell Directory Service
CORBA	Common Object Request Broker Architecture
DBCA	Oracle Database Configuration Assistant
DCE	Distributed Computing Environment
DES	Data Encryption Standard (U.S.)
DES40	Data Encryption Standard with 40-bit encryption keys
DES56	Data Encryption Standard with 56-bit encryption keys
DIT	Directory Information Tree
DN	Distinguished Name
ESM	Oracle Enterprise Security Manager
FIPS	Federal Information Processing Standard
GSSAPI	Generic Security Services Application Programming Interface

Table F–1 Abbreviations and Acronyms

Abbreviation / Acronym	Description
IIOP	Internet Inter-ORB Protocol
ISM	Bull Integrated System Management
ISP	Internet Service Provider
JDBC	Java Database Connectivity
JDK	Java Development Kit
JRE	Java Runtime Environment
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MD4	Message Digest 4; a checksumming algorithm that produces a128-bit hash total (checksum); see MD5.
MD5	Message Digest 5; a checksumming algorithm that produces a 128-bit hash total (checksum); stronger successor to MD4.
NetCA	Oracle Net Configuration Assistant
OCI	Oracle Call Interface
OID	Oracle Internet Directory
OSF	Open Software Foundation
PIN	Personal Identification Number
PKE	Public Key Encoding
PKI	Public Key Infrastructure
RADIUS	Remote Authentication Dial-In User Service
RC4	A symmetric encryption algorithm from RSA Data Security, Inc.
RSA	RSA Data Security, Inc.; refers to the RSA encryption module
SASL	Simple Authentication and Security Layer
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
SSO	Single Sign-on
Triple-DES	A version of the DES encryption algorithm that provides triple-encryption; see 3DES

Table F–1   Abbreviations and Acronyms

Abbreviation / Acronym	Description
WAN	Wide Area Network

Glossary

access control

The ability of a system to grant or limit access to specific data for specific clients or groups of clients.

Access Control List (ACL)

The group of access directives that you define. The directives grant levels of access to specific data for specific clients and/or groups of clients.

administrative context

A directory entry under which an **Oracle Context** resides. An administrative context can be a **directory naming context**. During directory access configuration, clients are configured with an administrative context in the directory configuration file (ldap.ora). The administrative context specifies the location of the Oracle Context in the directory whose entries a client expects to access

attribute

An item of information that describes some aspect of an entry. An entry comprises a set of attributes, each of which belongs to an **object class**. Moreover, each attribute has both a *type*, which describes the kind of information in the attribute, and a *value,* which contains the actual data.

authentication

The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to granting access to resources in a system. A recipient of an authenticated message can be certain of the message's origin (its sender). Authentication is presumed to preclude the possibility that another party has impersonated the sender.

authorization

Permission given to a user, program, or process to access an object or set of objects. In Oracle, authorization is done through the role mechanism. A single person or a group of people can be granted a role or a group of roles. A role, in turn, can be granted other roles. The set of privileges available to an authenticated entity.

base

The entry point in an **LDAP**-compliant directory.

CDS

See **Cell Directory Services (CDS)**.

Cell Directory Services (CDS)

An external naming method that enables users to use Oracle tools transparently and applications to access Oracle9*i* databases in a Distributed Computing Environment (DCE) environment.

certificate

An ITU x.509 v3 standard data structure that securely binds an identify to a public key.

A certificate is created when an entity's public key is signed by a trusted identity, a certificate authority. The certificate ensures that the entity's information is correct and that the public key actually belongs to that entity.

A certificate contains the entity's name, identifying information, and public key. It is also likely to contain a serial number, expiration date, and information about the rights, uses, and privileges associated with the certificate. Finally, it contains information about the certificate authority that issued it.

certificate authority

A trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are. When it certifies a user, the certificate authority first seeks verification that the user is not on the certificate revocation list (CRL), then verifies the user's identity and grants a certificate, signing it with the certificate authority's private key. The certificate authority has its own certificate and public key which it publishes. Servers and clients use these to verify signatures the certificate authority has made. A certificate authority might be an external company that offers certificate services, or an internal organization such as a corporate MIS department.

certificate chain

An ordered list of certificates containing an end-user or subscriber certificate and its certificate authority certificates.

checksumming

A mechanism that computes a value for a message packet, based on the data it contains, and passes it along with the data to authenticate that the data has not been tampered with. The recipient of the data recomputes the cryptographic checksum and compares it with the cryptographic checksum passed with the data; if they match, it is "probabilistic" proof the data was not tampered with during transmission.

Cipher Block Chaining (CBC)

An encryption method that protects against block replay attacks by making the encryption of a cipher block dependent on all blocks that precede it; it is designed to make unauthorized decryption incrementally more difficult. Oracle Advanced Security employs *outer* cipher block chaining because it is more secure than *inner* cipher block chaining, with no material performance penalty.

cipher suite

A set of authentication, encryption, and data integrity algorithms used for exchanging messages between network nodes. During an SSL handshake, for example, the two nodes negotiate to see which cipher suite they will use when transmitting messages back and forth.

cipher suite name

Cipher suites describe the kind of cryptographics protection that is used by connections in a particular session.

ciphertext

Message text that has been encrypted.

cleartext

Unencrypted plain text.

client

A client relies on a service. A client can sometimes be a user, sometimes a process acting on behalf of the user during a database link (sometimes called a proxy).

confidentiality

A function of cryptography. Confidentiality guarantees that only the intended recipient(s) of a message can view the message (decrypt the ciphertext).

CORBA

Common Object Request Broker Architecture. An architecture that enables pieces of programs, called objects, to communicate with one another regardless of the programming language in which they are written or the operating system on which they are running. CORBA was developed by an industry consortium known as the Object Management Group (OMG).

cryptography

The practice of encoding and decoding data, resulting in secure messages.

Database Administrator

(1) A person responsible for operating and maintaining an Oracle Server or a database application. (2) An Oracle username that has been given DBA privileges and can perform database administration functions. Usually the two meanings coincide. Many sites have multiple DBAs.

Database Installation Administrator

Also called a database creator. This administrator is in charge of creating new databases. This includes registering each database in the directory using the Database Configuration Assistant. This administrator has create and modify access to database service objects and attributes. This administrator can also modify the Default **domain**.

database link

A network object stored in the local database or in the network definition that identifies a remote database, a communication path to that database, and optionally, a username and password. Once defined, the database link is used to access the remote database.

A public or private database link from one database to another is created on the local database by a DBA or user.

A global database link is created automatically from each database to every other database in a network with Oracle Names. Global database links are stored in the network definition.

database method

See **Oracle database method**.

Database Security Administrator

Has create, modify, and read access for enterprise user security. This administrator has permissions on all of the domains in the enterprise and is responsible for:

- Administering the Oracle DBSecurityAdmins and OracleDBCreators groups.
- Creating new enterprise domains.
- Moving databases from one **domain** to another within the enterprise.

decryption

The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).

DES

The U.S. Data Encryption Standard.

dictionary attack

A common attack on passwords. the attacker creates a dictionary of many possible passwords and their corresponding verifiers. Through some means, the attacker then obtains the verifier corresponding to the target password, and obtains the target password by looking up the verifier in the dictionary.

Diffie-Hellman key negotiation algorithm

This is a method that lets two parties communicating over an insecure channel to agree upon a random number known only to them. Though the parties exchange information over the insecure channel during execution of the Diffie-Hellman key negotiation algorithm, it is computationally infeasible for an attacker to deduce the random number they agree upon by analyzing their network communications. Oracle Advanced Security uses the Diffie-Hellman key negotiation algorithm to generate session keys.

digital signature

A digital signature is created when a public key algorithm is used to sign the sender's message with the sender's private key. The digital signature assures that the document is authentic, has not been forged by another entity, has not been altered, and cannot be repudiated by the sender.

directory information tree (DIT)

A hierarchical tree-like structure consisting of the DNs of the entries.

directory naming context

A subtree which is of significance within a directory server. It is usually the top of some organizational subtree. Some directories only permit one such context which is fixed; others permit none to many to be configured by the directory administrator.

distinguished name (DN)

The unique name of a directory entry. It comprises all of the individual names of the parent entries back to the root.

domain

Any tree or subtree within the **Domain Name System (DNS)** namespace. Domain most commonly refers to a group of computers whose host names share a common suffix, the domain name.

Domain Name System (DNS)

A system for naming computers and network services that is organized into a hierarchy of **domains**. DNS is used in TCP/IP networks to locate computers through user-friendly names. DNS resolves a friendly name into an IP address, which is understood by computers.

In **Oracle Net**, DNS translates the host name in a TCP/IP address into an IP address.

encrypted text

Text that has been encrypted, using an encryption algorithm; the output stream of an encryption process. On its face, it is not readable or decipherable, without first being subject to **decryption**. Also called **ciphertext**. Encrypted text ultimately originates as **plaintext**.

encryption

The process of disguising a message rendering it unreadable to any but the intended recipient.

enterprise domain

A directory construct that consists of a group of databases and **enterprise roles**. A database should only exist in one enterprise domain at any time.

Enterprise Domain Administrator

User authorized to manage a specific **enterprise domain**, including the authority to add new enterprise domain administrators.

enterprise role

Access privileges assigned to **enterprise users**. A set of Oracle role-based **authorizations** across one or more databases in an **enterprise domain**. Enterprise roles are stored in the directory and contain one or more **global roles**.

enterprise user

A user defined and managed in a directory. Each enterprise user has a unique identify across an enterprise and uses a **wallet** to store its login credentials.

entry

The building block of a directory, it contains information about an object of interest to directory users.

external authentication

Verification of a user identity by a third party authentication service, such as Kerberos or RADIUS.

file system method

Storing fingerprint templates in files when configuring Identix Biometric authentication. The alternative is to use the **Oracle database method**.

global role

A role managed in a directory, but its privileges are contained within a single database.

HTTP

Hypertext Transfer Protocol: The set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. Relative to the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), HTTP is an application protocol.

HTTPS

The use of Secure Sockets Layer (SSL) as a sublayer under the regular HTTP application layer.

identity

The combination of the public key and any other public information for an entity. The public information may include user identification data such as, for example, an e-mail address. A user certified as being the entity it claims to be.

initial ticket

In Kerberos authentication, an initial ticket or ticket granting ticket (TGT) identifies the user as having the right to ask for additional service tickets. No tickets can be obtained without an initial ticket. An initial ticket is retrieved by running the kinit program and providing a password.

integrity

The guarantee that the contents of the message received were not altered from the contents of the original message sent.

IIOP

Internet Inter-ORB Protocol. A protocol developed by the Object Management Group (OMG) to implement CORBA solutions over the World Wide Web. IIOP enables browsers and servers to exchange integers, arrays, and more complex objects, unlike HTTP, which supports only transmission of text.

java code obfuscation

Java code **obfuscation** is used to protect Java programs from reverse engineering. A special program (an obfuscator) is used to scramble Java symbols found in the code. The process leaves the original program structure intact, letting the program run correctly while changing the names of the classes, methods, and variables in order to hide the intended behavior. Although it is possible to decompile and read non-obfuscated Java code, the obfuscated Java code is sufficiently difficult to decompile to satisfy U.S. government export controls.

KDC/TGS

Key Distribution Center/Ticket Granting Service. In Kerberos authentication, the KDC maintains a list of user principals and is contacted through the kinit program for the user's **initial ticket**. The Ticket Granting Service maintains a list of service principals and is contacted when a user wants to authenticate to a server providing such a service.

The KDC/TGS is a trusted third party that must run on a secure host. It creates ticket-granting tickets and service tickets. The KDC and TGS are usually the same entity.

Kerberos

A network authentication service developed under Massachusetts Institute of Technology's Project Athena that strengthens security in distributed environments. Kerberos is a trusted third-party authentication system that relies on shared secrets and assumes that the third party is secure. It provides single sign-on capabilities and database link authentication (MIT Kerberos only) for users, provides centralized password storage, and enhances PC security.

key

When encrypting data, a key is a value which determines the ciphertext that a given algorithm will produce from given plaintext. When decrypting data, a key is a value required to correctly decrypt a ciphertext. A ciphertext is decrypted correctly only if the correct key is supplied.

With a symmetric encryption algorithm, the same key is used for both encryption and decryption of the same data. With an asymmetric encryption algorithm (also called a public-key encryption algorithm or public-key cryptosystem), different keys are used for encryption and decryption of the same data.

key pair

A **public key** and its associated **private key**.

See **public/private key pair**.

kinstance

An instantiation or location of a service. This is an arbitrary string, but the host machine name for a service is typically specified.

kservice

An arbitrary name of a Kerberos service object.

LDAP

See **Lightweight Directory Access Protocol (LDAP)**.

Lightweight Directory Access Protocol (LDAP)

A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. The framework of design conventions supporting industry-standard directory products, such as the Oracle Internet Directory.

listener

A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.

Every time a client requests a network session with a server, a listener receives the actual request. If the client information matches the listener information, then the listener grants a connection to the server.

listener.ora file

A configuration file for the listener that identifies the:

- Listener name

- Protocol addresses that it is accepting connection requests on

- Services it is listening for

The `listener.ora` file typically resides in `$ORACLE_HOME/network/admin` on UNIX platforms and `ORACLE_HOME\network\admin` on Windows NT.

man-in-the-middle

A security attack characterized by the third-party, surreptitious interception of a message, wherein the third-party, the *man-in-the-middle*, decrypts the message, re-encrypts it (with or without alteration of the original message), and re-transmits it to the originally-intended recipient—all without the knowledge of the legitimate sender and receiver. This type of security attack works only in the absence of **authentication**.

MD5

An algorithm that assures data integrity by generating a 128-bit cryptographic message digest value from given data. If as little as a single bit value in the data is modified, the MD5 checksum for the data changes. Forgery of data in a way that will cause MD5 to generate the same result as that for the original data is considered computationally infeasible.

message authentication code

Also known as data authentication code (DAC). A **checksumming** with the addition of a secret key. Only someone with the key can verify the cryptographic checksum.

message digest

See **checksumming**.

network authentication service

A means for authenticating clients to servers, servers to servers, and users to both clients and servers in distributed environments. A network authentication service is a repository for storing information about users and the services on different servers to which they have access, as well as information about clients and servers on the network. An authentication server can be a physically separate machine, or it can be a facility co-located on another server within the system. To ensure availability, some authentication services may be replicated to avoid a single point of failure.

non-repudiation

Incontestable proof of the origin, delivery, submission, or transmission of a message.

obfuscation

A process by which information is scrambled into a non-readable form, such that it is extremely difficult to de-scramble if the algorithm used for scrambling is not known.

obfuscator

A special program used to obfuscate Java source code. See: **obfuscation**.

object class

A named group of **attributes**. When you want to assign attributes to an entry, you do so by assigning to that entry the object classes that hold those attributes. All objects associated with the same object class share the same attributes.

Oracle Context

An entry in an LDAP-compliant internet directory called `cn=OracleContext`, under which all Oracle software relevant information is kept, including entries for **Oracle Net** directory naming and **enterprise user** security. A top-level directory entry that contains the data used by any installed Oracle product that uses the directory.

There can be one or more Oracle Contexts in a directory. An Oracle Context is located under an **administrative context**.

Oracle database method

Using an Oracle database to store fingerprint templates when configuring Indentix Biometric authentication. The alternative is to use the **file system method**.

Oracle Net

An Oracle product that enables two or more computers that run the Oracle server or Oracle tools such as Designer/2000 to exchange data through a third-party network. Oracle Net supports distributed processing and distributed database capability. Oracle Net is an *open system* because it is independent of the communication protocol, and users can interface Oracle Net to many network environments.

Oracle PKI certificate usages

Defines Oracle application types that a **certificate** supports.

Password-Accessible Domains List

A group of **enterprise domain**s configured to accept connections from password-authenticated users.

peer identity

SSL connect sessions are between a particular client and a particular server. The identity of the peer may have been established as part of session setup. Peers are identified by **X.509 certificate chains**.

PKCS #12

A **public-key encryption** standard (PKCS). RSA Data Security, Inc. PKCS #12 is an industry standard for storing and transferring personal authentication credentials—typically in a format called a **wallet**.

plaintext

Message text that has not been encrypted.

principal

A uniquely-identified client or server. A Kerberos object, consisting of *kservice/kinstance@REALM*. See also *kservice, kinstance*, and *realm*.

private key

In public-key cryptography, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures. See **public/private key pair**.

proxy authentication

A process typically employed in an environment with a middle tier such as a firewall, wherein the end user authenticates to the middle tier, which thence

authenticates to the directory on the user's behalf—as its *proxy*. The middle tier logs into the directory as a *proxy user*. A proxy user can switch identities and, once logged into the directory, switch to the end user's identity. It can perform operations on the end user's behalf, using the authorization appropriate to that particular end user.

public key

In public-key cryptography, this key is made public to all. It is primarily used for encryption but can be used for verifying signatures. See **public/private key pair**.

public-key encryption

The process where the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the message is decrypted by the recipient using its private key.

public-key infrastructure

Information security technology utilizing the principles of public key cryptography. Public key cryptography involves encrypting and decrypting information using a shared public and private key pair. Provides for secure, private communications within a public network.

public/private key pair

A set of two numbers used for **encryption** and **decryption**, where one is called the **private key** and the other is called the **public key**. Public keys are typically made widely available, while private keys are held by their respective owners. Though mathematically related, it is generally viewed as computationally infeasible to derive the private key from the public key. Public and private keys are used only with asymmetric encryption algorithms, also called public-key encryption algorithms, or public-key cryptosystems. Data encrypted with either a public key or a private key from a **key pair** can be decrypted with its associated key from the key-pair. However, data encrypted with a public key cannot be decrypted with the same public key, and data enwrapped with a private key cannot be decrypted with the same private key.

realm

A Kerberos object. A set of clients and servers operating under a single key distribution center/ticket-granting service (KDC/TGS). *kservices* that are in different realms that share the same name are unique.

root key certificate

See **trusted certificate**.

schema

A collection of **attributes**, **object class**es, and their corresponding matching rules.

schema mapping

A pair of values in a database: the **base** in an **LDAP**-compliant directory at which users exist, and the name of the database schema to which they are mapped.

Secure Hash Algorithm (SHA)

An algorithm that assures data integrity by generating a 160-bit cryptographic message digest value from given data. If as little as a single bit in the data is modified, the Secure Hash Algorithm checksum for the data changes. Forgery of a given data set in a way that will cause the Secure Hash Algorithm to generate the same result as that for the original data is considered computationally infeasible.

An algorithm that takes a message of less than 264 bits in length and produces a 160-bit message digest. The algorithm is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision and inversion attacks.

Secure Sockets Layer (SSL)

An industry standard protocol designed by Netscape Communications Corporation for securing network connections. SSL provides authentication, encryption, and data integrity using public key infrastructure (PKI).

server

A provider of a service.

service

A network resource used by clients; for example, an Oracle database server.

service name

For Kerberos-based authentication, the **kservice** portion of a service principal.

service principal

See **principal**.

service table

In Kerberos authentication, a service table is a list of service principals that exist on a *kinstance*. This information must be extracted from Kerberos and copied to the Oracle server machine before Kerberos can be used by Oracle.

service ticket

Trusted information used to authenticate the client. A ticket-granting ticket is also known as the initial ticket, is obtained by directly or indirectly running kinit and providing a password, and is used by the client to ask for service tickets. A *service ticket* is used by a client to authenticate to a service.

session key

A key shared by at least two parties (usually a client and a server) that is used for data encryption for the duration of a single communication session. Session keys are typically used to encrypt network traffic; a client and a server can negotiate a session key at the beginning of a session, and that key is used to encrypt all network traffic between the parties for that session. If the client and server communicate again in a new session, they negotiate a new session key.

session layer

A network layer that provides the services needed by the presentation layer entities that enable them to organize and synchronize their dialogue and manage their data exchange. This layer establishes, manages, and terminates network sessions between the client and server. An example of a session layer is Network Session.

SHA

See **Secure Hash Algorithm (SHA)**.

shared schema

Database or application schemas that can be used by multiple enterprise users. Oracle Advanced Security supports the mapping of multiple enterprise users to the same shared schema on a database, which lets an administrator avoid creating an account for each user in every database. Instead, the administrator can create a user in one location, the enterprise directory, and map the user to a shared schema that other enterprise users can also map to. Sometimes called **user/schema separation**.

single key-pair wallet

A **PKCS #12**-format **wallet** that contains a single user **certificate** and its associated **private key**. The **public key** is imbedded in the certificate.

single password authentication

The ability of a user to authenticate with a single, globally unique password. The user authenticates to one database or application with a single password, and can thence authenticate to other databases or applications with the same password. In the Oracle Advanced Security implementation, the password is stored in an LDAP-compliant directory and protected with encryption and Access Control Lists. Using single password authentication, users use a single, globally unique password to authenticate multiple times (to multiple databases and applications). *Single password, multiple authentications.*

single sign-on

The ability of a user to *authenticate once,* combined with strong authentication occurring transparently in subsequent connections to other databases or applications. Single sign-on lets a user access multiple accounts and applications with a single password, entered during a single connection. *Single password, single authentication.* Oracle Advanced Security supports Kerberos, CyberSafe, DCE, and SSL-based single sign-on.

smart card

A plastic card (like a credit card) with an embedded integrated circuit for storing information, including such information as user names and passwords, and also for performing computations associated with authentication exchanges. A smart card is read by a hardware device at any client or server.

A smartcard can generate random numbers which can be used as one-time use passwords. In this case, smartcards are synchronized with a service on the server so that the server expects the same password generated by the smart card.

sniffer

Device used to surreptitiously listen to or capture private data traffic from a network.

sqlnet.ora file

A configuration file for the client or server that specifies:

- Client domain to append to unqualified service names or net service names
- Order of naming methods the client should use when resolving a name
- Logging and tracing features to use
- Route of connections

- Preferred Oracle Names servers

- External naming parameters

- Oracle Advanced Security parameters

The `sqlnet.ora` file typically resides in `$ORACLE_HOME/network/admin` on UNIX platforms and `ORACLE_HOME\network\admin` on Windows platforms.

ticket

A piece of information that helps identify who the owner is. See **service ticket**.

token card

A device for providing improved ease-of-use for users through several different mechanisms. Some token cards offer one-time passwords that are synchronized with an authentication service. The server can verify the password provided by the token card at any given time by contacting the authentication service. Other token cards operate on a challenge-response basis. In this case, the server offers a challenge (a number) which the user types into the token card. The token card then provides another number (cryptographically-derived from the challenge), which the user then offers to the server.

transport layer

A networking layer that maintains end-to-end reliability through data flow control and error recovery methods. **Oracle Net** uses Oracle protocol supports for the transport layer.

trusted certificate

A trusted certificate, sometimes called a root key certificate, is a third party identity that is qualified with a level of trust. The trusted certificate is used when an identity is being validated as the entity it claims to be. Typically, the certificate authorities you trust are called trusted certificates. If there are several levels of trusted certificates, a trusted certificate at a lower level in the certificate chain does not need to have all its higher level certificates reverified.

trusted certificate authority

See **certificate authority**.

trust point

See **trusted certificate**.

user/schema separation

See **shared schema**.

wallet

A wallet is a data structure used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services. A **Wallet Resource Locator** (WRL) provides all the necessary information to locate the wallet.

wallet obfuscation

Wallet **obfuscation** is used to store and access an Oracle **wallet** without querying the user for a password prior to access (supports **single sign-on**).Certain machine-specific information is used to generate a secret key that is used to encrypt the wallet.

Wallet Resource Locator

A wallet resource locator (WRL) provides all necessary information to locate a **wallet**. It is a path to an operating system directory that contains a wallet.

WRL

See **Wallet Resource Locator**.

X.509

Public keys can be formed in various data formats. The X.509 v3 format is one such popular format.

Index

X