Oracle® Advanced Security

Administrator’s Guide

Release 8.1.6

December 1999

Part No. A76932-01

Oracle Advanced Security Administrator’s Guide, Release 8.1.6

Part No. A76932-01

Primary Author: Gilbert Gonzalez

Contributors: Pierre Baudin, Kamalendu Biswas, Kristy Browder, Mary Ann Davidson, Quan Dinh, Pramodini Gattu, Gary Gilchrist, Michael Hwa, Shuvayu Kanjilal, Lakshmi Kethana, Cynthia Kibbe, Nina Lewis, Van Le, Jack Melnick, Rita Moran, Andy Philips, Vipin Samar, Radhika Shah, P.V. Shivkumar, Richard Smith, Deborah Steiner, Ginger Tabora, Juliet Tran, Ramana Turlapati, Sandy Venning, Rick Wessman

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

Portions of Oracle Advanced Security have been licensed by Oracle Corporation from RSA Data Security.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle, SQL*Net, and SQL*PLus are registered trademarks, and Oracle8*i*, Oracle7, and Net8 are trademarks of Oracle Corporation. All other company or product names mentioned are used for identification purposes only and may be trademarks of their respective owners.

Contents

2 Configuring Data Encryption and Integrity

3 Thin JBDC Support

4 Configuring RADIUS Authentication

5 Configuring CyberSafe Authentication

6 Configuring Kerberos Authentication

7 Configuring SecurID Authentication

8 Configuring Identix Biometric Authentication

9 Configuring DCE GSSAPI Authentication

10 Configuring Secure Socket Layer Authentication

11 Choosing and Combining Authentication Methods

Part II Oracle DCE Integration

12 Overview of Oracle DCE Integration

13 Configuring DCE for Oracle DCE Integration

14 Configuring Oracle for Oracle DCE Integration

15 Connecting to an Oracle Database in DCE

16 DCE and Non-DCE Interoperability

Part III Oracle8i Security/Directory Integration

17 Managing Enterprise User Security

20 Using Oracle Enterprise Security Manager

Part IV 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 LDAP Directory Schema for Oracle Database Security

F Oracle Implementation of Java SSL

Glossary

Index

List of Figures

List of Tables

Send Us Your Comments

Oracle Advanced Security Administrator's Guide, Release 8.1.6

Part No. A76932-01

Oracle Corporation welcomes your comments and suggestions on the quality and usefulness of this publication. 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 about this manual?

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

- E-mail - infodev@us.oracle.com
- FAX - (650) 506-7228. Attn: Information Development
- Postal service:

  Oracle Corporation
  Server Technologies Documentation Manager
  500 Oracle Parkway, 4OP12
  Redwood Shores, CA 94065
  USA

If you would like a reply, please give your name, address, and telephone number below.

____

____

____

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

Preface

Oracle Advanced Security (formerly Oracle Advanced Networking Option) provides enhanced security and authentication to the Net8 network. This guide provides generic information on each Oracle Advanced Security feature.

This Preface contains the following topics:

- What This Guide Contains
- How This Guide Is Organized
- Notational Conventions
- Terms and Abbreviations
- Related Publications
- Your Comments Are Welcome

What This Guide Contains

This guide contains generic information on how to configure Net8 network to use Oracle Advanced Security. Use this guide with the guide that describes how to install and configure Oracle Advanced Security on your particular platform.

You can install and configure Oracle Advanced Security with other Oracle networking products and configure everything at once, or you can add Oracle Advanced Security to an already existing network.

How This Guide Is Organized

This guide is organized into the following parts:

- Part I: "Oracle Advanced Security Features"
- Part II: "Oracle DCE Integration"
- Part III: "Oracle8i Security / Directory Integration"
- Part IV: "Appendixes"

Each part describes a different set of Oracle Advanced Security features.

Part I: Oracle Advanced Security Features

Chapter 1, "Introduction to Oracle Advanced Security"

This chapter provides an overview of Oracle Advanced Security single sign-on and security features. These features include:

- network security
- data encryption
- data integrity checking
- external authentication
- single sign-on
- authorizations

Note: These features have been previously packaged as Oracle Advanced Networking Option and Secure Network Services.

This chapter also includes a brief overview of the authentication methods available with this release.

Chapter 2, "Configuring Data Encryption and Integrity"

This chapter describes how to configure data encryption and integrity within an existing Net8 release 8.1.6 network.

Chapter 3, "Thin JBDC Support"

This chapter provides an overview of the Java implementation of Oracle Advanced Security, which allows Thin Java Database Connectivity (JDBC) clients to connect securely to Oracle8*i* databases.

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 SecurID Authentication"

This chapter describes how to configure SecurID authentication in combination with the Oracle server and Oracle clients for use with the Security Dynamics ACE/Server and token cards. It includes system requirements and known limitations. It also contains troubleshooting information if you experience problems while configuring SecurID authentication.

Chapter 8, "Configuring Identix Biometric Authentication"

This chapter describes how to configure and use Oracle biometric authentication, which enables use of the Identix fingerprint authentication device.

Chapter 9, "Configuring DCE GSSAPI Authentication"

This chapter describes how to configure Oracle DCE GSSAPI authentication to provide DCE authentication even if you are not using other DCE services in your network.

Chapter 10, "Configuring Secure Socket Layer Authentication"

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

Chapter 11, "Choosing and Combining Authentication Methods"

This chapter describes how to use conventional username/password authentication even if you have configured another authentication service. It also describes how to configure the network to use one or more authentication services in the network using Oracle Advanced Security and how to set up more than one authentication service on a client or on a server.

Part II: Oracle DCE Integration

Chapter 12, "Overview of Oracle DCE Integration"

This chapter provides a brief discussion of OSF DCE and Oracle DCE Integration.

Chapter 13, "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 14, "Configuring Oracle 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 15, "Connecting to an Oracle Database in DCE"

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

Chapter 16, "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 III: Oracle8i Security/Directory Integration

Chapter 17, "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 18, "Using Oracle Wallet Manager"

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

Chapter 19, "Oracle Enterprise Login Assistant"

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

Chapter 20, "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 Oracle8*i* databases.

Part IV: 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 provides the required configuration parameter options required to provide the FIPS 140-1 Level 2 evaluated configuration.

Appendix E, "LDAP Directory Schema for Oracle Database Security"

This appendix describes the object classes and attributes defined in the LDAP directory schema for Oracle database security.

Appendix F, "Oracle Implementation of Java SSL"

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

Notational Conventions

The following syntax conventions are used in this guide:

Italic Font

Italic characters indicate that the parameter, variable, or expression in the command syntax must be replaced by a value that you provide. Italics can also indicate emphasis or the first mention of a technical term.

`Monospace Font`

Monospace font indicates something the computer displays.

`Bolded Monospace Font`

Bolded monospace font indicates:

- Terms defined in the Glossary
- Text you need to enter exactly as shown.

Punctuation

Punctuation other than brackets and vertical bars must be entered as shown.

[ ]

Brackets enclose optional items. Do not enter the brackets.

( )

Parentheses enclose all SQL*Net and Net8 Keyword-Value pairs in connect descriptors. They must be entered as part of the connect descriptor, as in `(KEYWORD=`*`value`*`)`.

|

A vertical bar represents a choice of two or more options. You must enter one of the options separated by the vertical bar. Do not enter the vertical bar.

UPPERCASE

Uppercase characters within the text represent parameters.

Terms and Abbreviations

Refer to the following table for a list of terms and abbreviations used in this document and their definitions.

Term or Abbreviation	Definition
CDS	Cell Directory Service
DCE	Distributed Computing Environment
GSSAPI	Generic Security Services Application Programming Interface
JDK	Java Development Kit
JRE	Java Runtime Environment
OSF	Open Software Foundation
PIN	Personal Identification Number

Related Publications

Refer to the appropriate Oracle platform-specific documentation to install and configure Oracle Advanced Security software on your particular platform.

In addition, see the following Oracle documents for information that applies across platforms:

- *Net8 Administrator's Guide*
- *Oracle8i Distributed Database Systems*
- *Oracle8i Enterprise JavaBeans and CORBA Developer's Guide*
- *Oracle8i JDBC Developer's Guide and Reference*
- *Oracle Internet Directory Administrator's Guide*

For information on roles and privileges, see:

- *Oracle8i Administrator's Guide*

For third-party vendor documentation on security and single sign-on features see:

- *RADIUS Administrator's Guide*
- *Security Dynamics' ACE/Server Installation Manual, Release 3.3*
- *Security Dynamics' ACE/Server Version 3.3 Administration Manual*
- *ACE/Server Version 2.0 Client for UNIX*

- *CyberSafe TrustBroker Release Notes, Release 5.2.6*
- *CyberSafe TrustBroker Administrator's Guide, Release 5.2.6*
- *CyberSafe TrustBroker Navigator Administrator's Guide, Release 5.2.6*
- *CyberSafe TrustBroker UNIX User's Guide, Release 5.2.6*
- *CyberSafe TrustBroker Windows and Windows NT User's Guide, Release 5.2.6*

For information on MIT Kerberos see:

- CyberSafe Trust Broker documentation
- Notes on building and installing Kerberos from Kerberos V5 source distribution
- CNS (Cygnus Network Security) documentation from http://www.cygnus.com/library-dir.html

For additional information about the Open Software Foundation (OSF) Distributed Computing Environment (DCE), see the following OSF documents published by Prentice Hall, Inc.:

- *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*

For information about Identix products, see the following Identix documentation.

Client side documentation:

- *Identix TouchNet II User's Guide*

Server side documentation:

- *Identix TouchNet II System Administrator's Guide*

Your Comments Are Welcome

We value and appreciate your comment as an Oracle user and reader of our manuals. As we write, revise, and evaluate our documentation, your opinions are the most important feedback we receive.

You can send comments and suggestions about this reference to the Information Development department at the following e-mail address:

infodev@us.oracle.com

If you prefer, you can send letters or faxes containing your comments to:

Server Technologies Documentation Manager
Oracle Corporation
500 Oracle Parkway, 4OP12
Redwood Shores, CA 94065
Fax: (650) 506-7228 Attn.: Information Development

Part I

Oracle Advanced Security Features

Part I of this document includes information on how to configure encryption, data integrity, and authentication into your existing Net8 release 8.1.6 network. Refer also to the port-specific documentation on how to install and configure Oracle Advanced Security.

The following chapters of the *Oracle Advanced Security Administrator's Guide* provide generic information on the security related features of Oracle Advanced Security.

- Chapter 1, "Introduction to Oracle Advanced Security"
- Chapter 2, "Configuring Data Encryption and Integrity"
- Chapter 3, "Thin JBDC Support"
- Chapter 4, "Configuring RADIUS Authentication"
- Chapter 5, "Configuring CyberSafe Authentication"
- Chapter 6, "Configuring Kerberos Authentication"
- Chapter 7, "Configuring SecurID Authentication"
- Chapter 8, "Configuring Identix Biometric Authentication"
- Chapter 9, "Configuring DCE GSSAPI Authentication"
- Chapter 10, "Configuring Secure Socket Layer Authentication"
- Chapter 11, "Choosing and Combining Authentication Methods"

For information on DCE integration: See Part II, "Oracle DCE Integration."

For information on Security and Directory integration: See Part III, "Oracle8i Security / Directory Integration."

1

Introduction to Oracle Advanced Security

This chapter introduces Oracle Advanced Security (formerly Oracle Advanced Networking Option) **encryption**, **integrity**, and **authentication** features. These features are available to network products using Net8, including Oracle8*i*, Oracle Designer, Oracle Developer, and any other Oracle or third-party products that support Net8.

Topics covered in this chapter include the following:

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

About Oracle Advanced Security

Oracle Advanced Security (formerly Oracle Advanced Networking Option and Secure Network Services) provides a comprehensive suite of security features to protect enterprise networks and securely extend corporate networks to the Internet. Oracle Advanced Security 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 and beyond.

This section contains the following topics:

- Network Security in a Distributed Environment
- Security Threats

Network Security in a Distributed Environment

Oracle databases power the largest and most popular web sites. Organizations around the world are deploying distributed databases and client/server applications in record numbers, often on a global scale, based on Net8 and Oracle8*i*. This proliferation of distributed computing has been matched by an increase in the amount of information that organizations now place on computers. Employee records, financial records, product testing information, and other sensitive or critical data have moved from filing cabinets into file structures. The volume of critical or sensitive information on computers has increased the value of data that can be compromised.

Security Threats

The increased distribution of data in distributed environments brings with it serious security threats, including the following:

- Data Tampering
- Eavesdropping and Data Theft
- Falsifying User Identities
- Managing Multiple Passwords

Data Tampering

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

Eavesdropping and Data Theft

Over the Internet and in Wide Area Network (WAN) environments, both public carriers and private network owners often route portions of their network through insecure land lines, extremely vulnerable microwave and satellite links, or a number of servers, leaving valuable data open to view by any interested party. In Local Area Network (LAN) 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 easily installed to eavesdrop on network traffic.

Falsifying User Identities

In a distributed environment, it becomes more feasible for a user to falsify an identity to gain access to sensitive and important 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 routed instead to a terminal masquerading as Server B.

Managing Multiple Passwords

In a distributed system, users often need to remember multiple passwords for the different applications and services that they use. For example, a developer can have access to an application in development on a workstation, a production system on a mini-computer, a PC for creating documents, and several computers or intranet sites for testing, reporting bugs, and managing configurations.

Users generally respond to managing the passwords of multiple accounts in one of the following ways:

- If they can choose their own passwords, users can 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 can also choose to standardize them so that they are the same on all machines. 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 guessed from knowing one password.

- Users with complex passwords can simply write them down where an attacker can easily find them, or forget them, requiring administration and support efforts.

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

Oracle Advanced Security Features

Oracle Advanced Security protects against these threats to the security of distributed environments. It provides the following features, each of which is described in this section:

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

Data Privacy

Oracle Advanced Security ensures that data is not disclosed during transmission through the following types of encryption:

- RSA Encryption
- DES Encryption

RSA Encryption

RSA encryption is an encryption module that uses the RSA Data Security 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, and 128 bits.

Since the Oracle Advanced Security RSA RC4 40-bit and 56-bit implementations meet the U.S. government export guidelines for encryption products, Oracle provides an export version of the media and exports it to all but a few countries, allowing most companies to safeguard their entire worldwide operations with this software.

DES Encryption

The U.S. Data Encryption Standard (DES) is required for financial and many other institutions. Oracle Advanced Security offers a standard, optimized 56-bit key DES encryption algorithm. Due to former U.S. government export restrictions, Oracle Advanced Security also offers DES40, a version of DES that combines the standard DES encryption algorithm with the international availability of a 40-bit key. While DES56 is now exportable, Oracle Advanced Security supports DES40 for backwards compatibility. Selecting the algorithm to use for network encryption is a user configuration option, allowing varying levels of security and performance for different types of data transfers.

> **More Information:** For more information, see Chapter 2, "Configuring Data Encryption and Integrity" and Appendix A, "Data Encryption and Integrity Parameters."

Data Integrity

To ensure that data has not been modified, deleted, or replayed during transmission, Oracle Advanced Security optionally generates a cryptographically secure message digest—through cryptographic checksums using the MD5 algorithm—and includes it with each packet sent across the network.

Moreover, the SSL feature of Oracle Advanced Security allows the use of the Secure Hash Algorithm (SHA). SHA is slightly slower than MD5, but produces a larger message digest to make it more secure against brute-force collision and inversion attacks.

Authentication

Establishing user identity is of primary concern in distributed environments; otherwise, there can be little confidence in limiting privileges by user. Passwords are the most common authentication method in use, and Oracle Advanced Security integrates with stronger authentication services. Oracle Advanced Security release 8.1.6 provides authentication through Oracle authentication adapters that support various third-party authentication services.

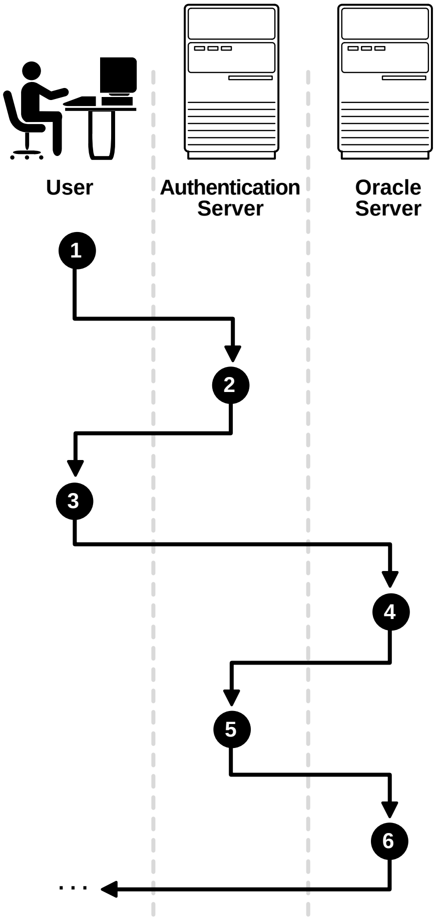
Many Oracle Advanced Security authentication methods use centralized authentication. This can give you high confidence in the identity of users, clients, and servers in distributed environments. Having a central facility authenticate all members of the network (clients to servers, servers to servers,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

Single Sign-On

Centralized authentication can also provide the benefit of single sign-on for users. Single sign-on allows users to 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.

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, providing some identification—such as a token or password—proving that the user is who he or she claims to be.
2. After authenticating the user, the authentication server passes a ticket or credentials back to the client. This ticket may include an expiration time.
3. The client can now take these credentials and pass them to the Oracle server while asking for a service, such as connection to a database.
4. The server, to verify that the credentials are valid, sends them back to the authentication server.
5. If the authentication server accepts the credentials, it notifies the Oracle server.
6. The Oracle server performs the requested task for the user. If the credentials are not accepted, the requested service is denied.

Oracle Advanced Security supports the following authentication methods:

- SSL
- RADIUS
- Kerberos and CyberSafe
- Smart Cards (RADIUS-Compliant)
- Token Cards (SecurID or RADIUS-Compliant)
- Biometric Authentication (Identix or RADIUS-Compliant)
- Bull ISM

SSL

SSL (Secure Sockets Layer) 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).

The Oracle Advanced Security SSL feature can be used to secure communications between any client and any server. Specifically, you can use SSL to authenticate the following:

- any client or server to one or more Oracle servers
- an Oracle server to any client

SSL features can be used by themselves or in combination with other authentication methods supported by Oracle Advanced Security. For example, SSL can be used with Kerberos, using the encryption provided by SSL in combination with the Kerberos authentication method.

You can configure SSL to require server authentication only, or both client and server authentication.

RADIUS

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

The Oracle Advanced Security support for Kerberos and CyberSafe provides the benefits of single sign-on and centralized authentication in an Oracle environment. Kerberos is a trusted third-party authentication system that relies on shared secrets. It assumes 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 Kerberos authentication and through the CyberSafe TrustBroker, a commercial Kerberos-based authentication server.

Note: Oracle authentication for Kerberos provides database link authentication (also called proxy authentication). CyberSafe and SecurID do *not* provide support for proxy authentication.

Smart Cards (RADIUS-Compliant)

This authentication method uses a hardware device that looks like a credit card. It has memory and a processor and is read by a smart card reader located at the client workstation.

Smart cards offer the following benefits:

Increased security	Smart cards rely on two-factor authentication. The smart card can be locked, and only the user who possesses the card and 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.

Token Cards (SecurID or RADIUS-Compliant)

Token cards can provide improved 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, namely, another number cryptographically-derived from the challenge, which the user offers to the server.

Token cards offer the following benefits:

Ease of password management	Password management is easy because there is one token card rather than multiple passwords.
Enhanced password security	To masquerade as a user, a malefactor has to 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 only need to remember, at most, a PIN instead of multiple passwords.
Enhanced accountability	Token cards provide a stronger authentication mechanism, therefore users are more accountable for their actions.

You can use SecurID tokens through either the SecurID adapter or through RADIUS.

Biometric Authentication (Identix or RADIUS-Compliant)

Identix Biometric Authentication is used on both the clients and Oracle servers to communicate fingerprint-based authentication data between the authentication server and the clients. Other biometric authentication devices that are RADIUS compliant can integrate with Oracle Advanced Security using RADIUS to authenticate Oracle users.

Bull ISM

Bull Integrated System Management (ISM) is an offering from Bull Worldwide Information Systems that provides system administrators with a variety of management tools. This authentication method is available on the AIX platform only. See the AIX-specific documentation for more information.

Authorization

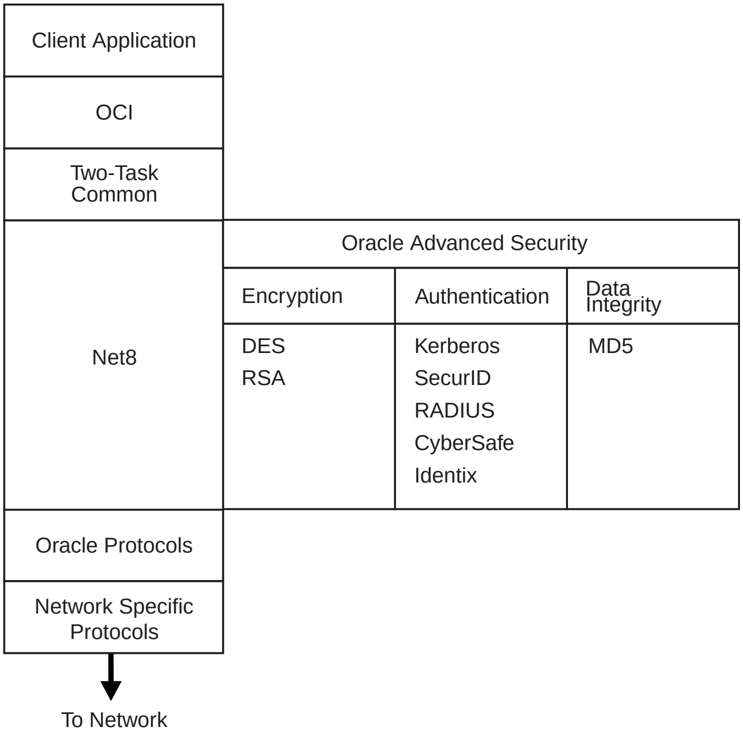
User authorization, already a standard feature of Oracle8*i* with roles and privileges, is significantly enhanced by using the authentication methods supported by Oracle Advanced Security. For example, on certain platforms such as Solaris, Oracle Advanced Security supports authorization with DCE.

Authorizations are also provided with the Oracle Advanced Security directory integration feature. Oracle Advanced Security can integrate with LDAP version 3-compliant directories. 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 intend to use it for additional purposes.

Oracle Advanced Security Architecture

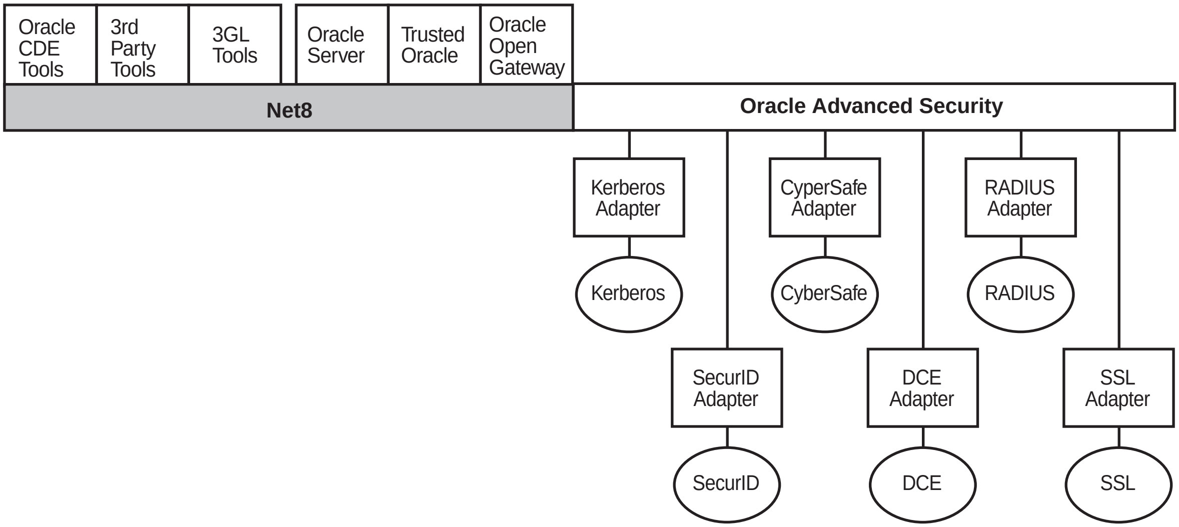
Oracle Advanced Security is an add-on product to a standard Net8 Server or Net8 Client. Figure 1–2 shows the location of Oracle Advanced Security within a typical stack in an Oracle networking environment.

Figure 1–2 Oracle Advanced Security in an Oracle Networking Environment



Oracle Advanced Security supports authentication through adapters that are very much like the existing Oracle protocol adapters. As shown in Figure 1–3, authentication adapters integrate below the Net8 interface and allow existing applications to take advantage of new authentication systems transparently, without any changes to the application.

Figure 1–3 Net8 with Authentication Adapters



More Information: For more information on stack communications in an Oracle networking environment, see *Net8 Administrator's Guide.*

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 to the standard Net8 Server or Net8 Client. It must be purchased and installed on both the client and the server.

Oracle Advanced Security release 8.1.6 requires Net8 release 8.1.6 and supports Oracle8*i* Enterprise Edition. Table 1–1 lists additional system requirements.

Important: Oracle Advanced Security is not available with Oracle8*i* Standard Edition, nor are any of its components.

Install Oracle Advanced Security on each client and server where Oracle Advanced Security is required.

Table 1–1 Authentication Methods and System Requirements

Authentication Method	System Requirements
CyberSafe TrustBroker	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 TrustBroker, release 1.2 or later installed on a physically secure machine that runs the authentication server. CyberSafe TrustBroker 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.
SecurID	ACE / Server 3.3 or higher running on the authentication server.
Identix Biometric	Identix hardware and driver installed on each Biometric Manager station and client.
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 a platform that supports the Java Native Interface as specified in release 1.1 of the Java Development Kit from JavaSoft.

Authentication Method	System Requirements
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.

Note: Oracle Advanced Security release 8.1.6 provides secure communication when used with earlier releases, however the security functionality defaults to that provided by the earlier release.

Oracle Configuration for Network Authentication

This section describes the following parameters that are set when configuring Oracle for network authentication. Specifically, it describes 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

More Information: For information on configuring a particular authentication method, see the method's corresponding chapter in this guide. See also Appendix A, "Data Encryption and Integrity Parameters."

Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora

The following parameter must be set in the `sqlnet.ora` file for clients and servers to be able to use an Oracle Advanced Security authentication method:

```
SQLNET.AUTHENTICATION_SERVICES=(oracle_authentication_method)
```

For example, the parameter must be set in the `sqlnet.ora` files on all clients and servers that use the Kerberos Authentication as follows:

```
SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)
```

Verifying that REMOTE_OS_AUTHENT Is Not Set to TRUE

Oracle Corporation recommends that you add the following parameter to the initialization file used for the database instance when you configure the Oracle authentication method:

```
REMOTE_OS_AUTHENT=FALSE
```

Attention: Setting REMOTE_OS_AUTHENT to TRUE can allow a security breach, because it allows someone using a non-secure protocol, such as TCP, to 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 following parameter is set in the `sqlnet.ora` file on either the client or server side:

```
SQLNET.AUTHENTICATION_SERVICES=(NONE)
```

the database attempts to use the provided user name and password to log the user in. However, if REMOTE_OS_AUTHENT is set to FALSE, 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 result in previously created externally-identified users not being able to connect 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.

Note: This applies to creating Oracle users for use with all Oracle authentication methods.

More Information: Refer to *Oracle8i Administrator's Guide* and *Oracle8i Distributed Database Systems.*

Oracle Advanced Security Restrictions

Oracle Applications support Oracle Advanced Security encryption and data integrity. However, because Oracle Advanced Security requires Net8 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 the Windows platform. The portions of these products that use Oracle Display Manager (ODM) do not take advantage of Oracle Advanced Security, since ODM does not use Net8.

2

Configuring Data Encryption and Integrity

This chapter covers 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 discusses and compares the various encryption algorithms used in both domestic and export editions of Oracle Advanced Security.

The following topics are included in this section:

- Domestic and Export Editions
- DES Algorithm for Standards-Based Encryption
- DES40 Algorithm for Backwards Compatibility
- RSA RC4 Algorithm for High Speed Encryption
- RC4_128 for Domestic Customers
- RC4_40 and RC4_56 for International Customers
- Triple-DES Support in SSL

Domestic and Export Editions

Due to export controls placed on encryption technology, Oracle Advanced Security is available in a domestic edition and an export edition.

Table 2–1 provides a summary of domestic and export algorithms.

Table 2–1 Differences Between Domestic and Export Editions

Domestic Edition contains:	Export Edition contains:
Diffie-Hellman key negotiation algorithm	Diffie-Hellman key negotiation algorithm
MD5 message digest algorithm	MD5 message digest algorithm
The following encryption algorithms:	The following encryption algorithms:
■ DES40	■ DES40
■ DES	■ DES
■ RC4_40	■ RC4_40
■ RC4_56	■ RC4_56
■ RC4_128	

In certain circumstances, a special license can be obtained to export the domestic version. Special licenses can allow banks, wholly owned subsidiaries of U.S. corporations, and e-commerce venders to obtain the domestic edition. Export and import regulations vary from country to country and change from time to time, so it is important to check on current restrictions in your area.

> **Note:** Previous releases of Oracle Advanced Security had three versions: domestic, upgrade, and export. The recent relaxation of U.S. government export laws allows Oracle Corporation to ship both 40-bit and 56-bit encryption algorithms in the Oracle Advanced Security 8.1.6 Export Edition and to obsolete the need for an Upgrade Edition containing 56-bit DES.

DES Algorithm for Standards-Based Encryption

Oracle Advanced Security for international use provides the DES (Data Encryption Standard) algorithm for customers with specialized encryption needs. DES has been a U.S. government standard for many years and is sometimes mandated in the financial services industry. In most specialized banking systems today, DES is the algorithm used to protect large international monetary transactions. Oracle Advanced Security allows this high-security system to be used to protect any kind of application, without any custom programming.

In a secure cryptosystem, the plaintext (a message that has not been encrypted) cannot be derived from the ciphertext (the encrypted message) except by using the secret decryption key. In a symmetric cryptosystem, a single key serves as both the encryption and the decryption key. DES is a secret-key, symmetric cryptosystem: both the sender and the receiver must know the same secret key, which is used both to encrypt and decrypt the message. DES is the most well-known and widely-used cryptosystem in the world.

DES40 Algorithm for Backwards Compatibility

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 for use by customers outside the U.S. and Canada who wanted to use a DES-based encryption algorithm while there were stronger encryption export laws. Now, in Oracle Advanced Security release 8.1.6, both DES40 and DES are available internationally. DES40 is still supported to ensure backward-compatibility for international customers.

RSA RC4 Algorithm for High Speed Encryption

The RC4 algorithm, developed by RSA Data Security Inc., has quickly become the de-facto international standard for high-speed data encryption. Despite ongoing attempts by cryptographic researchers to crack the RC4 algorithm, the only feasible method of breaking its encryption known today remains brute-force, systematic guessing. RC4 is a stream cipher that operates at several times the speed of DES, making it possible to encrypt even large bulk data transfers with minimal performance consequences.

RC4_128 for Domestic Customers

RC4 is a variable key-length stream cipher. Oracle Advanced Security release 8.1.6 for domestic use offers an implementation of RC4 with a 128 bit key. This provides strong encryption with no sacrifice in performance when compared to other key lengths of the same algorithm.

RC4_40 and RC4_56 for International Customers

Oracle has obtained a special license to export the RC4 data encryption algorithm with a 40-bit or 56-bit key to virtually all destinations where other Oracle products are available. This allows international corporations to safeguard their entire operations with fast, strong cryptography.

Triple-DES Support in SSL

The Oracle Advanced Security Secure Sockets Layer (SSL) feature allows the use of triple-DES (3DES). This form of encryption involves encrypting input data three times, which can occur in a number of ways. A potential drawback of triple-DES, depending on the speed of the communications channel, is that it requires more computing power than regular DES.

More Information: See Chapter 10.

Oracle Advanced Security Data Integrity

Encryption of network data provides data privacy, so no unauthorized party is able to view the plaintext data as it passes over the network. Oracle Advanced Security also provides protection against two other forms of attack: data modification attack and replay attack.

Types of Attacks

Data Modification Attack

In a data modification attack, an unauthorized party on the network intercepts data in transit and changes parts of that data before retransmitting it. An example of this is changing the dollar amount of a banking transaction from $100 to $10,000.

Replay Attack

In a replay attack, an entire set of valid data is repeatedly interjected onto the network. An example would be to repeat a valid bank $100 account transfer transaction several times.

Data Integrity Algorithms Supported

Oracle Advanced Security uses a keyed, sequenced implementation of the MD5 message digest algorithm to protect against both of these forms of active attack. MD5 creates a checksum that changes if the data is altered in any way. This protection is activated independently from the encryption features provided, so you can enable data integrity with or without enabling encryption.

The Oracle Advanced Security SSL feature allows the use of either Message Digest 5 (MD5) or Secure Hash Algorithm (SHA-1) for data integrity.

Diffie-Hellman-Based Key Management

The secrecy of encrypted data depends on the existence of a secret key shared between the communicating parties. Providing and maintaining such secret keys is known as key management. In a multi-user environment, secure key distribution may be difficult; public-key cryptography was invented to solve this problem. Oracle Advanced Security uses the public-key based 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 should be changed frequently to minimize the effects of a compromised key. For this reason, the Oracle Advanced Security key management facility changes the session key with every session.

Overview of Site-Specific Diffie-Hellman Encryption Enhancement

Oracle Advanced Security includes the Diffie-Hellman key negotiation algorithm to choose keys both for encryption and for data integrity.

A key is a secret shared by both sides of the connection and by no one else. Without the key, it is extremely difficult to decrypt an encrypted message or to tamper undetectably with a crypto-checksummed message.

Overview of Authentication Key Fold-in Encryption Enhancement

The purpose of Authentication Key Fold-in encryption enhancement is to defeat a possible "person-in-the-middle attack" on the Diffie-Hellman key negotiation. It strengthens the session key significantly by combining a shared secret that is 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 itself to the server, they establish a shared secret that is only known to both sides. Oracle Advanced Security then combines the shared secret and Diffie-Hellman session key to generate a stronger session key that would defeat the person-in-the-middle who has no way of knowing the shared secret.

Note: The authentication key fold-in encryption enhancement feature is included in Oracle Advanced Security and requires no configuration by the system or network administrator.

Configuring Data Encryption and Integrity

The following configuration instructions assume that the Net8 network software has already been installed and is running. The network administrator sets up the encryption and checksumming configuration parameters. The profile (`sqlnet.ora`) on clients and servers using data encryption and integrity must contain some or all of the parameters listed in this section.

> **Note:** The following instructions demonstrate how to configure Oracle Advanced Security native encryption and integrity. See Chapter 10, "Configuring Secure Socket Layer Authentication," to configure the SSL feature for encryption, integrity, and authentication.

This section contains the following topics:

- Activating Encryption and Integrity
- Negotiating Encryption and Integrity
- Setting Encryption and Integrity Parameters Using Net8 Assistant

Activating Encryption and Integrity

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

When the server is searching for a match between the algorithms it has made available and the algorithms the client has made available, it picks the first algorithm in its own list that also appears in the client's list. If one side of the connection does not specify a list of algorithms, all the algorithms that are installed on that side are acceptable. The connection fails with error message ORA-12650 if an algorithm that is not installed is specified on either side.

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

More Information: See 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 below 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 to not 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 allowed. 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 allowed if the other side is set to REQUIRED or REQUESTED. If the other side is set to REQUIRED or REQUESTED, and an 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 allows it.

In this scenario, this side of the connection specifies that the security service is wanted 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 disallow 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 2–2 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, a 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 2–2 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 the Net8 Assistant or by modifying the `sqlnet.ora` file.

Note: Encryption is not used when both the client and the server have ACCEPTED set to ON.

Setting Encryption and Integrity Parameters Using Net8 Assistant

You can enter or change encryption and integrity parameter settings using Net8 Assistant.

This section describes the following tasks:

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

> **More Information:** For a description of each parameter and a sample configuration file using encryption and integrity, see Appendix A, "Data Encryption and Integrity Parameters".
>
> For more detailed configuration information, see Net8 Assistant online help.

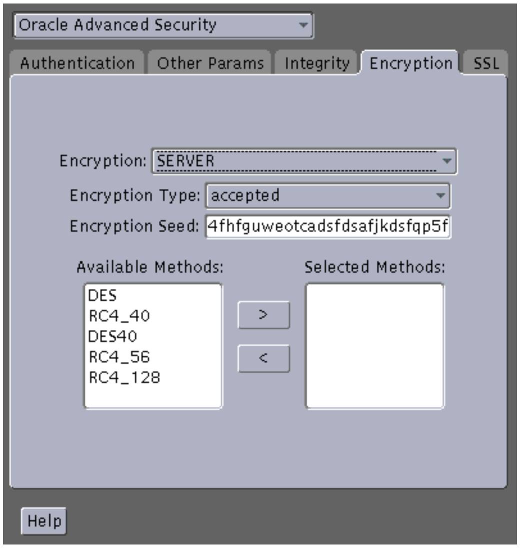
Configure Encryption on the Client and on the Server

To configure encryption on the client and on the server:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Encryption tab.



Encryption: SERVER

Encryption Type: accepted

Encryption Seed: 4fhfguweotcadsfdsafjkdsfqp5f

Available Methods:

DES
RC4_40
DES40
RC4_56
RC4_128

Selected Methods:

5. Depending on which machine you are configuring, in the Encryption list, select CLIENT or SERVER.

6. From the Encryption Type list, select REQUESTED, REQUIRED, ACCEPTED, or REJECTED.

   More Information: See Appendix A for valid encryption algorithms.

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 method in the Available Methods list.

   Move it to the Selected Methods list by clicking the right arrow button [>]. Repeat for each additional method you want to use.

9. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entries:

```
SQLNET.ENCRYPTION_SERVER = [accepted | rejected | requested | required]
SQLNET.ENCRYPTION_TYPES_SERVER = (valid_encryption_algorithm [,valid_
encryption_algorithm])
```

```
SQLNET.CRYPTO_SEED = "10-70 random characters"
```

The `sqlnet.ora` file should contain the following entries for 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"
```

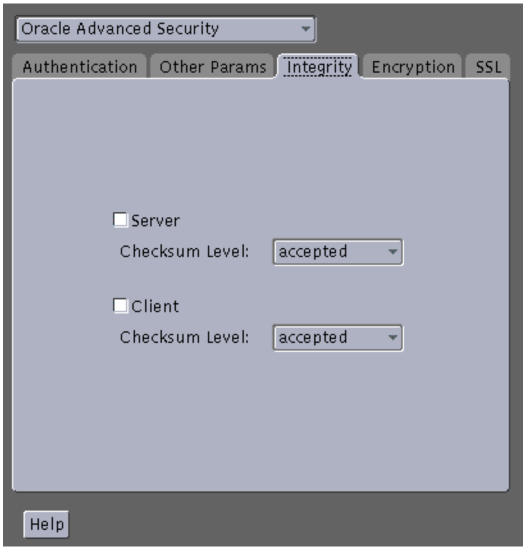
Configure Integrity on the Client and Server

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

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Integrity tab.



□ Server

Checksum Level: accepted

□ Client

Checksum Level: accepted

5. Depending on which machine you are configuring, select the Server or Client check box.
6. From the Checksum Level list, select one of the following checksum level values: required, requested, accepted, rejected.
7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entries:

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

   The `sqlnet.ora` file should contain the following entries for the client:

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

Note: Currently, the only supported crypto-checksum algorithm choice is RSA Data Security's MD5 algorithm.

3

Thin JBDC Support

This chapter provides an overview of the Java implementation of Oracle Advanced Security, which allows Thin Java Database Connectivity (JDBC) clients to connect securely to Oracle8*i* databases.

This chapter covers the following topics:

- About the Java Implementation
- Configuration Parameters

> **More Information:** For information on JDBC, including examples, see the *Oracle8i JDBC Developer's Guide and Reference.*

About the Java Implementation

The Java implementation of Oracle Advanced Security provides network encryption and integrity protection for Thin JDBC clients communicating with Oracle8*i* databases with Oracle Advanced Security enabled.

The following topics are described in this section:

- JDBC Support
- Securing Thin JDBC
- Implementation Overview
- Obfuscation

JDBC Support

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

Furthermore, Oracle implements two types of JDBC drivers: Thick JDBC drivers built on top of the C-based Net8 client, as well as a Thin (Pure Java) JDBC driver to support downloadable applets. Oracle JDBC drivers are used to create JDBC applications to communicate with Oracle databases.

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 Oracle8*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. The Oracle JDBC Thin driver does not support Oracle Advanced Security third party authentication features such as RADIUS, Kerberos, and SecurID. However, the Oracle JDBC OCI driver support is the same as thick client support, in which all of Oracle Advanced Security features are implemented.

Oracle Advanced Security continues to encrypt and provide integrity checking of Net8 traffic between Net8 clients and Oracle servers using algorithms written in C. The Oracle Advanced Security Java implementation provides Java versions of the following encryption algorithms:

- DES40
- DES56
- RC4_40
- RC4_56

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

In addition, this implementation provides data integrity checking for Thin JDBC with the MD5 algorithm.

Note: The Java implementation is available only in Oracle Advanced Security Export Edition (not Domestic Edition). Consequently, only export-level key lengths are implemented.

Implementation Overview

On the server side, the negotiation of algorithms and the generation of keys function exactly the same as native Oracle Advanced Security encryption. This allows 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 Net8 clients. Thin JDBC contains a complete implementation of a Net8 client in pure Java.

Consistent with other encryption implementations, the Java implementation of Oracle Advanced Security prevents access to the encryption algorithms, makes it impossible to double encrypt data, and encrypts data only as it passes through the network. Users cannot alter the keyspace nor alter the encryption algorithms themselves.

Obfuscation

Code implementing cryptography and written in Java must be obfuscated in order to comply with U.S. government export controls. Therefore, this implementation protects Java classes and methods that contain encryption and decryption capabilities with obfuscation software.

Java Byte Code Obfuscation is a process often used by companies 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, allowing the program to run correctly while changing the names of the classes, methods, and variables in order to hide the intended behavior. Although it is easy to decompile and read non-obfuscated Java code, the obfuscation process renders the target code nearly impossible to interpret once decompiled.

Configuration Parameters

A properties class object containing several configuration parameters is passed to the Oracle Advanced Security interface. The tables in this chapter list the configuration parameters for the properties class objects for the following:

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

Table 3–1 Client Encryption Level

parameter name:	oracle.net.encryption_client
parameter type:	string
parameter class:	static
allowable values:	REJECTED, ACCEPTED, REQUESTED, REQUIRED
default value	ACCEPTED
description:	defines the level of security that the client wants to negotiate with the server
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();

Table 3–2 Client Encryption Selected List

parameter name:	oracle.net.encryption_types_client
parameter type:	string
parameter class:	static
allowable values:	DES40C, DES56C, RC4_40, RC4_56
description:	defines the encryption algorithm to be used
syntax:	up.put("oracle.net.encryption_types_client", *alg*);
example:	up.put("oracle.net.encryption_types_client", "DES40C") where up is defined as Properties up = new Properties();

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

Table 3–3 Client Integrity Selected List

parameter name:	oracle.net.crypto_checksum_types_client
parameter type:	string
parameter class:	static
allowable values:	MD5
description:	defines the data integrity algorithm to be used
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();

Table 3–4 Client Integrity Level

parameter name:	oracle.net.crypto_checksum_client
parameter type:	string
parameter class:	static
allowable values:	REJECTED, ACCEPTED, REQUESTED, REQUIRED
default value	ACCEPTED
description:	defines the level of security that it wants to negotiate with the server for data integrity
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();

Note: The Java implementation is available only in Oracle Advanced Security Export Edition, therefore only export-level key lengths are available.

4

Configuring RADIUS Authentication

This chapter describes how to configure Oracle8*i* for use with RADIUS (Remote Authentication Dial-In User Service).

This chapter covers the following topics:

- RADIUS Overview
- RADIUS Authentication Modes
- Enabling RADIUS Authentication and Accounting
- Logging on to the Database

RADIUS Overview

RADIUS (Remote Authentication Dial-In User Service) is a client-server security protocol most widely known for enabling remote authentication and access. Oracle Advanced Security uses this industry standard in a client-server network environment.

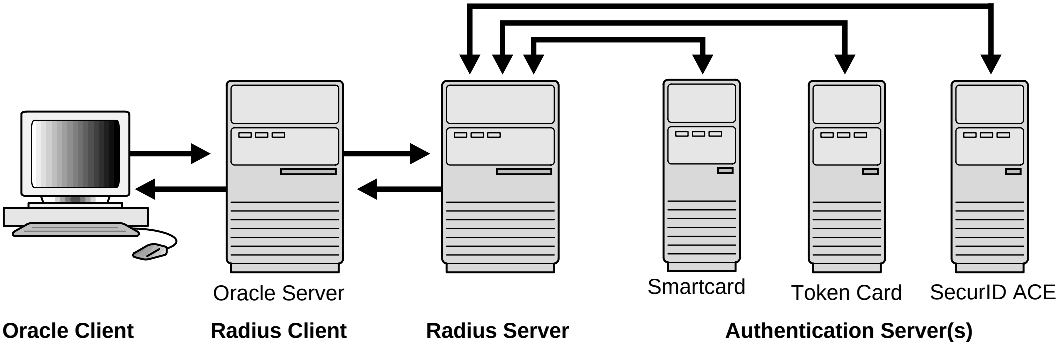
You can enable the network to use any authentication method that supports the RADIUS standard, including token cards and smart cards, simply by installing and configuring the RADIUS feature. Moreover, when you use RADIUS, you can change the authentication method without modifying either the Oracle client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

From the user's perspective, the entire authentication process takes place seamlessly and transparently. When the user seeks access to an Oracle server, the Oracle server, acting as the RADIUS client, notifies the RADIUS server. The RADIUS server then:

- looks up the user's security information
- passes authentication and authorization information between the appropriate authentication server or servers and the Oracle server
- grants the user access to the Oracle server
- logs, by means of the RADIUS accounting feature, such information as when, how often, and for how long the user logged on the Oracle server

In an Oracle environment, as shown in Figure 4–1, the Oracle server acts as the RADIUS client; it passes information between the Oracle client and the RADIUS server. Similarly, the RADIUS server passes information between the Oracle server and the appropriate authentication server or servers. To secure authentication information during transport, RADIUS converts it to a hash value.

Figure 4–1 RADIUS in an Oracle Environment



The four components, Oracle client, Oracle server / RADIUS client, RADIUS server, and authentication server, can reside on the same machine or on separate machines. When the Oracle client and Oracle server reside on the same machine, they share the same `sqlnet.ora` file.

RADIUS server vendors are often the authentication server vendors as well, and therefore authentication can be processed on the RADIUS server. For example, the Security Dynamics ACE / Server is a RADIUS server and an authentication server. It authenticates the user’s passcode itself.

More Information: For information about the `sqlnet.ora` file, see *Net8 Administrator’s Guide.*

Table 4–1 lists each RADIUS component and the information it stores.

Table 4–1 RADIUS Authentication Components

Component	Stored Information
Oracle client	Configuration setting for communicating through RADIUS
Oracle 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 and authentication server can be one in the same.

RADIUS Authentication Modes

User authentication can take place in either of two ways:

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

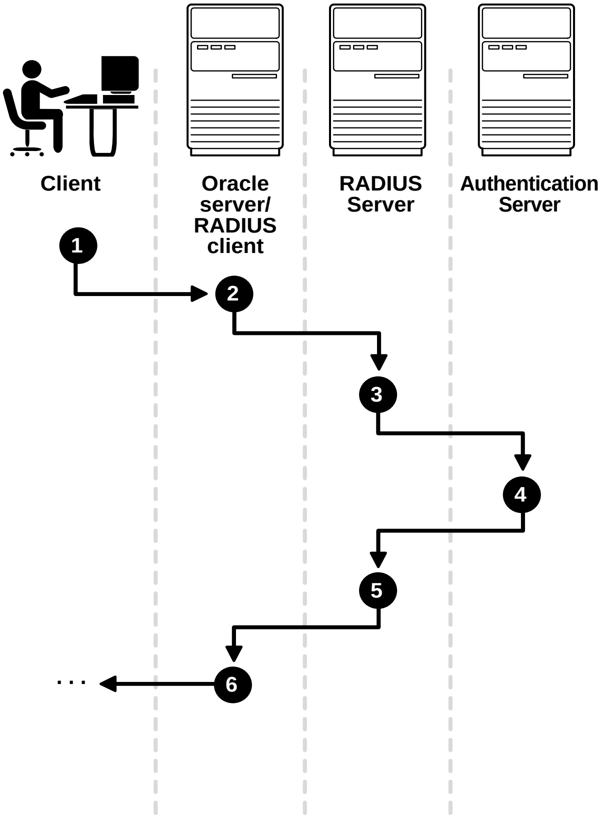
Synchronous Authentication Mode

In the synchronous mode, RADIUS allows you to use various authentication methods, including passwords, SecurID token cards, and smart 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 machine passes this data to the Oracle server.
2. The Oracle 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 back to the RADIUS server either an Access Accept or an Access Reject message.

   Note: If the RADIUS server is the authentication server, Steps 3 and 4 are combined.
5. The RADIUS server passes this response to the Oracle server / RADIUS client.
6. The Oracle server / RADIUS client passes the response to the Oracle client.



Example: Synchronous Authentication with SecurID Token Cards

With SecurID authentication, each user has a token card which displays a dynamic number that changes every sixty seconds. To gain access to the Oracle 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 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 server, which, in turn, passes it to the Oracle client. The user is now authenticated and able to access the appropriate tables and applications.

More Information: For more information on SecurID token cards, see Chapter 1, "Introduction to Oracle Advanced Security" and Chapter 7, "Configuring SecurID Authentication". See also the documentation provided by Security Dynamics.

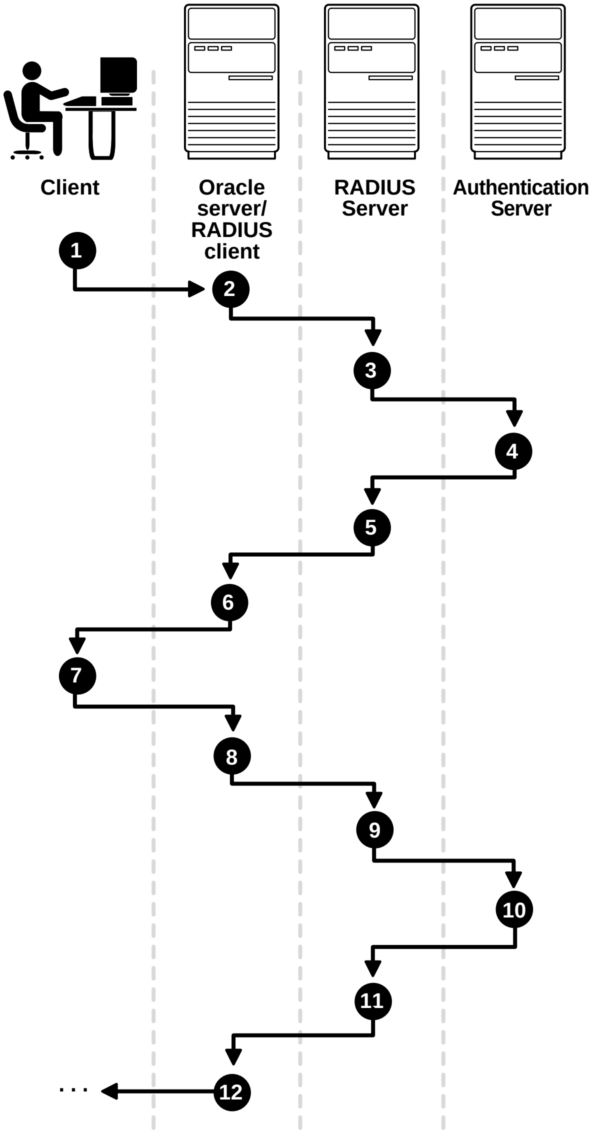
Challenge-Response (Asynchronous) Authentication Mode

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, or 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 the Oracle server. The client machine passes the data to the Oracle server.
2. The Oracle 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.
4. The authentication server sends a challenge, such as a random number, to the RADIUS server.
5. The RADIUS server sends the challenge to the Oracle server / RADIUS client.
6. The Oracle server / RADIUS client, in turn, sends it to the Oracle client. A graphical 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 the user with a dynamic password that he or she enters into the graphical interface. The Oracle client passes the user's response to the Oracle server / RADIUS client.
8. The Oracle 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 back to the RADIUS server either an Access Accept or an Access Reject message.
11. The RADIUS server passes the response to the Oracle server / RADIUS client.
12. The Oracle 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 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 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 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 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 server by entering a password, the information is passed to the appropriate authentication server by way of the Oracle 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 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 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 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 general tasks as described in this section:

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

Task 2: Configure RADIUS Authentication

Task 3: Add the RADIUS Client Name to the RADIUS Server Database

Task 4: Create a User and Grant Access

Task 5: Configure RADIUS Accounting

Task 6: Configure the Authentication Server for Use with RADIUS.

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

Task 8: Create and Grant Roles

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

RADIUS is installed with Oracle Advanced Security during a typical installation of Oracle8*i*.

> **More Information:** For information on installing Oracle Advanced Security and the RADIUS adapter, see the platform-specific installation documentation for Oracle8*i*.

Task 2: Configure RADIUS Authentication

This step contains the following topics:

- Basic RADIUS Configuration on the Oracle Client
- Basic RADIUS Configuration on the Oracle Server
- Configuration of Additional RADIUS Features

Unless otherwise indicated, perform these configuration tasks by using the Net8 Assistant or by using any text editor to modify the `sqlnet.ora` file.

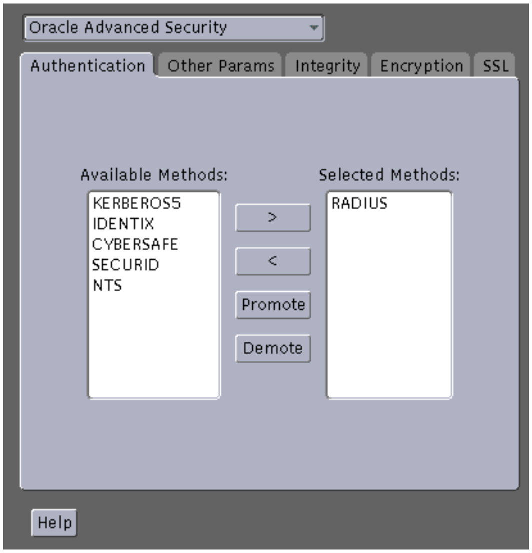
Basic RADIUS Configuration on the Oracle Client

Perform the following steps to configure the client for RADIUS authentication:

1. Start Net8 Assistant as follows:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Authentication tab.



5. From the Available Methods list, select RADIUS.
6. Click the right-arrow button [>] 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 updates with the following entry:

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

Basic RADIUS Configuration on the Oracle Server

Perform the following tasks as described in this section:

- Create the RADIUS Secret Key File on the Oracle Server
- Set RADIUS Parameters in the sqlnet.ora File
- Set Oracle Server Initialization Parameters

Create the RADIUS Secret Key File on the Oracle Server

Perform the following steps to create the RADIUS secret key file on the Oracle server.

1. Obtain the RADIUS secret key from the RADIUS server. The administrator of the RADIUS server creates a shared secret key for each RADIUS client, which can be as simple as "test123".
2. On the Oracle 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 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 server.

More Information: For information on obtaining the secret key, see the administration documentation for the RADIUS server.

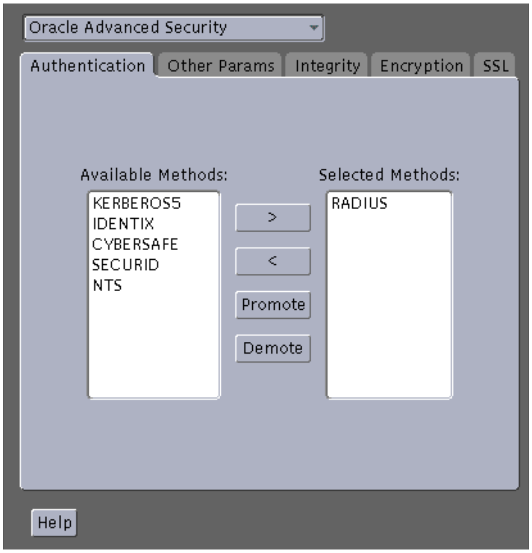
Note: For security reasons, Oracle Corporation recommends that you change this file to root access only.

Set RADIUS Parameters in the `sqlnet.ora` File

To configure RADIUS parameters on the server:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the Authentication tab.

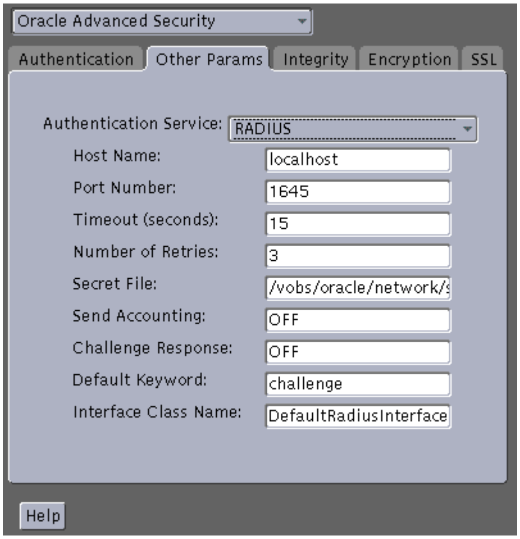


Available Methods:
KERBEROS5
IDENTIX
CYBERSAFE
SECURID
NTS

Selected Methods:
RADIUS

5. From the Available Methods list, select RADIUS.
6. Move RADIUS to the Selected Methods list by clicking the right-arrow button [>].
7. Arrange the selected methods in order of desired 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 RADIUS to be the first service used, put it at the top of the list.

8. Click the Other Params tab.



Authentication Service: RADIUS
Host Name: localhost
Port Number: 1645
Timeout (seconds): 15
Number of Retries: 3
Secret File: /vobs/oracle/network/s
Send Accounting: OFF
Challenge Response: OFF
Default Keyword: challenge
Interface Class Name: DefaultRadiusInterface

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.

> **Note:** Ensure that the default value of the Secret File field is valid.

11. Choose File > Save Network Configuration.

The `sqlnet.ora` file updates 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 Server Initialization Parameters

Configure the initialization parameter file which you can find in the directory `$ORACLE_BASE\admin\`*`db_name`*`\pfile` on UNIX and *`ORACLE_BASE/`*`admin/`*`db_name/`*`pfile` on Windows NT. Specify the following values in this file:

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

Caution: Setting REMOTE_OS_AUTHENT to `TRUE` can allow a security breach because it allows someone using a non-secure protocol, such as TCP, to perform an operating system-authorized login (formerly referred to as an OPS$ login).

More Information: For information on setting initialization parameters on the Oracle server, refer to the *Oracle8i Reference* and the *Oracle8i Administrator's Guide.*

Configuration of Additional RADIUS Features

Perform the following tasks as described in this section:

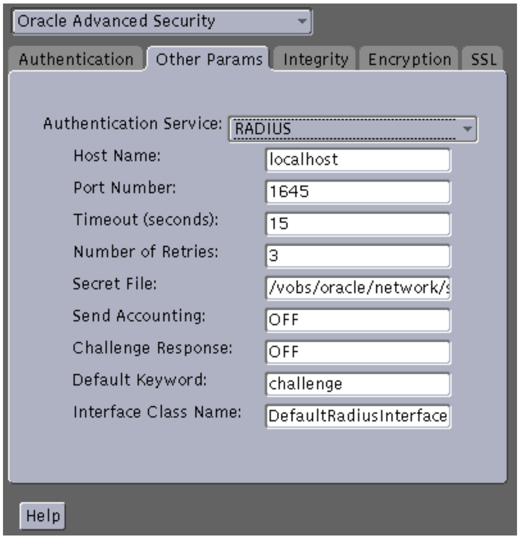
- Change Default Settings
- Configure Challenge-Response
- Set Parameters for an Alternate RADIUS Server

Change Default Settings

Perform the following steps to change default settings using the Net8 Assistant.

1. Start Net8 Assistant as follows:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the Other Params tab.



Authentication Service: RADIUS
Host Name: localhost
Port Number: 1645
Timeout (seconds): 15
Number of Retries: 3
Secret File: /vobs/oracle/network/s
Send Accounting: OFF
Challenge Response: OFF
Default Keyword: challenge
Interface Class Name: DefaultRadiusInterface

5. From the Authentication Service list, select RADIUS.

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

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 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 server resends messages to the primary RADIUS server. The default is three retries. **More Information:** For instructions on configuring RADIUS accounting, see "Task 5: Configure RADIUS Accounting" in this chapter.
Secret File	Specifies the location of the secret key on the Oracle server. The field specifies the location of the secret key file, not the secret key itself. **More Information:** For information on specifying the secret key, see "Create the RADIUS Secret Key File on the Oracle Server". **Note:** For security reasons, Oracle Corporation recommends that you change this file to root access only.

7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates 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.

More Information: For information on how to customize the challenge-response user interface, see Appendix C, "Integrating Authentication Devices Using RADIUS".

To configure challenge-response, perform the following steps.

1. Set the JAVA_HOME environment variable to the JRE or JDK location on the system where the Oracle client is to run:
   - On UNIX, enter the following at the command 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 to the following:

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

2. If you use the challenge-response authentication mode, RADIUS presents a Java-based graphical interface requesting a password and additional information, such as 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 the graphical interface.

   Use a text editor to add the following parameter to the `sqlnet.ora` file:

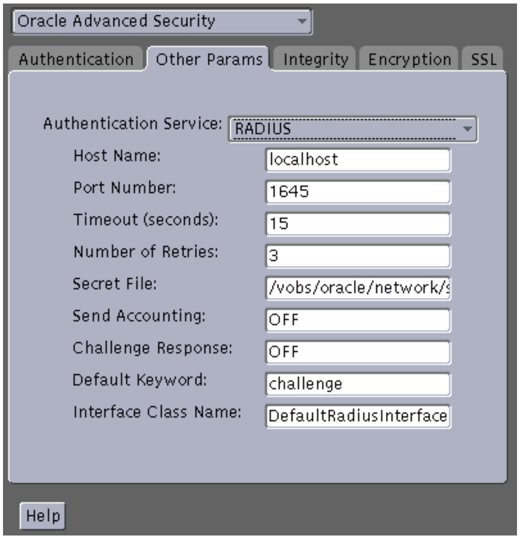
   ```
   SQLNET.RADIUS_CLASSPATH=(location of netradius.jar) and (location of JRE
   rt.jar)
   ```

The following is an example of setting the path for the Java classes:

```
SQLNET.RADIUS_CLASSPATH=/ohome/network_src/jlib/
                             netradius.jar:/usr/local/packages/jre1.1.7B/lib/rt.jar
```

3. Start Net8 Assistant as follows:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
4. In the navigator’s pane, expand Local > Profile.
5. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
6. Click the Other Params tab.



Authentication Service: RADIUS
Host Name: localhost
Port Number: 1645
Timeout (seconds): 15
Number of Retries: 3
Secret File: /vobs/oracle/network/
Send Accounting: OFF
Challenge Response: OFF
Default Keyword: challenge
Interface Class Name: DefaultRadiusInterface

7. From the Authentication Service list, select RADIUS.
8. In the Challenge Response field, enter ON to enable challenge-response.

9. In the Default Keyword[1] field, accept the default value of the challenge or enter a keyword for requesting a challenge from the RADIUS server.
10. In the Interface Class Name field, accept the default value of DefaultRadiusInterface or enter name of the class you have created to handle the challenge-response conversation between the Oracle client and the RADIUS server.
11. Choose File > Save Network Configuration.

    The `sqlnet.ora` file updates 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 the following 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=(1645)
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: 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.

[1] 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 allow the user not to use a password to verify his or her 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 on the configuration of the RADIUS server.

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 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.

   > **More Information:** See the administration documentation for your RADIUS server.

Task 4: Create a User and Grant Access

1. Perform the following steps to create and grant access to a user identified externally on the Oracle server.

   Launch SQL*Plus and enter the following commands:

   ```
   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 a Windows NT platform, you can use the Security Manager tool in the Oracle Enterprise Manager.

   > **More Information:** See *Oracle8i Administrator's Guide* and *Oracle8i Distributed Database Systems.*

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

More Information: See the administration documentation for the RADIUS server.

Task 5: Configure RADIUS Accounting

RADIUS Accounting logs information about access to the Oracle 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.

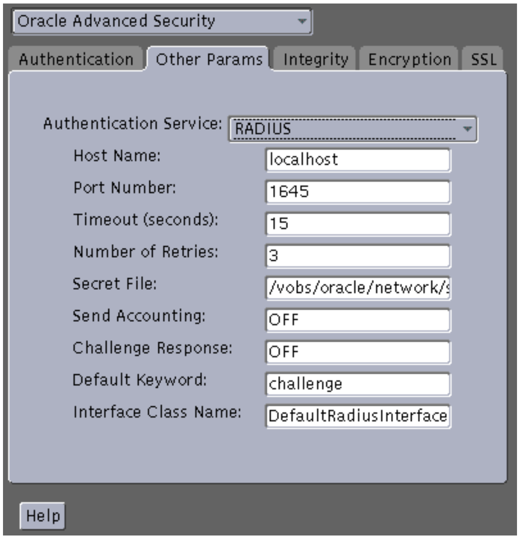
Set RADIUS Accounting on the Oracle Server

To enable or disable RADIUS accounting:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Other Params tab.



Authentication Service: RADIUS

Host Name: localhost

Port Number: 1645

Timeout (seconds): 15

Number of Retries: 3

Secret File: /vobs/oracle/network/s

Send Accounting: OFF

Challenge Response: OFF

Default Keyword: challenge

Interface Class Name: DefaultRadiusInterface

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 updates 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.

> **More Information:** For information on configuring RADIUS accounting, see the administration documentation for the RADIUS server.

Task 6: Configure the Authentication Server for Use with RADIUS

More Information: For instructions on configuring the authentication server, see the documentation for the authentication server. The "Related Publications" section in the Preface contains a list of possible resources.

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

More Information: See the documentation for the RADIUS server.

Task 8: Create and Grant Roles

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

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

Perform the following steps to configure roles on the Oracle server:

1. Use a text editor to set the OS_ROLES parameter in the initialization parameters file on the Oracle server.
2. Stop and restart the Oracle server.
3. Create each role the RADIUS server is to manage on the Oracle server with IDENTIFIED EXTERNALLY.

More Information: See *Oracle8i Administrator's Guide.*

To configure roles on the RADIUS server, see Table 4–2 for a list of parameters and their descriptions. Enter the following to create a role name:

```
ORA_DatabaseName.DatabaseDomainName_RoleName
```

Table 4–2 Create a Role on the RADIUS Server

Parameter	Description
DatabaseName	The name of the Oracle 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 server belongs. The value is the same as the value of the db_domain initialization.
RoleName	The name of the role created in the Oracle server.

The following is an example of a role created on the RADIUS server:

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

More Information: See the RADIUS server administration documentation.

Logging on to the Database

If you are using the synchronous authentication mode, launch SQL*Plus and type the following 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 (asynchronous) mode, launch SQL*Plus and, at the prompt, type the following:

```
CONNECT /@database_alias
```

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

5

Configuring CyberSafe Authentication

This chapter contains information on how to configure Oracle for use with CyberSafe, as well as a brief overview of the steps to configure CyberSafe to authenticate Oracle users.

This chapter covers the following topics:

- Enabling CyberSafe Authentication
- Troubleshooting the Configuration of the CyberSafe Authentication Adapter

Enabling CyberSafe Authentication

Enable CyberSafe authentication by performing the following tasks.

Note: Perform the tasks in the order listed.

Task 1: Install the CyberSafe Server

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

More Information: See the CyberSafe documentation listed in the "Related Publications" in the Preface.

Task 2: Install the CyberSafe TrustBroker Client

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

More Information: See the CyberSafe documentation listed in "Related Publications" in the Preface.

Task 3: Install the CyberSafe Application Security Toolkit

Perform this task on the client and on the server.

More Information: See the CyberSafe documentation listed in "Related Publications" in the Preface.

Task 4: Configure a Service Principal for an Oracle Server

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

The name of the principal should have 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 the fully-qualified name of the machine on which Oracle is running
REALM	the domain 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 actual programs that are run. 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 machine on which Oracle is running is dbserver.someco.com, and the realm is SOMECO.COM, the principal name is as follows:

```
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 server and CyberSafe TrustBroker client machines.

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
   ```

   Note: 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 above.

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 the additional entries.
3. Move the CyberSafe service table to the CyberSafe TrustBroker client machine. If the service table is on the same machine 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 machine 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 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, since this can allow a security breach.

Task 6: Install an Oracle Server

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

More Information: See the Oracle8*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 Oracle8*i*. The Oracle Universal Installer guides you through the entire installation process.

More Information: See the Oracle installation documentation for your platform.

Task 8: Configure Net8 and Oracle on the Server and Client

More Information: See the platform-specific documentation.

Task 9: Configure CyberSafe Authentication

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

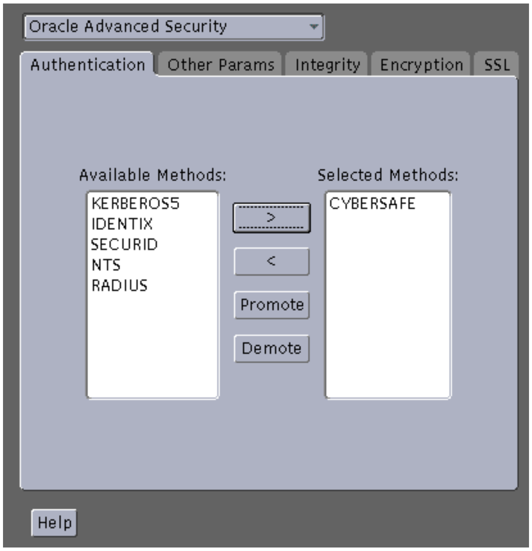
- Configure CyberSafe on the Client and on the Server
- Set REMOTE_OS_AUTHENT in the Initialization Parameter File

Configure CyberSafe on the Client and on the Server

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

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the Authentication tab.



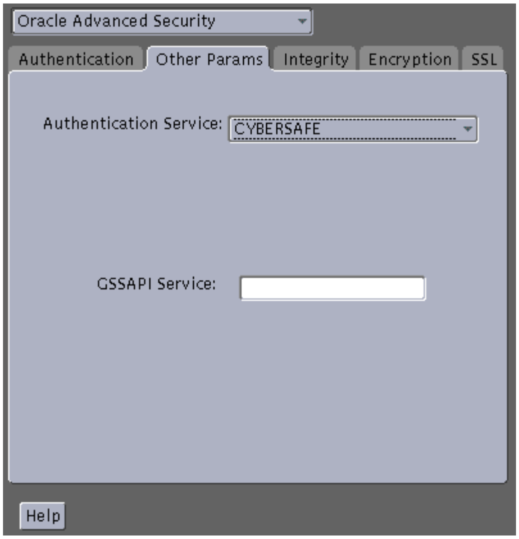
Available Methods:
KERBEROS5
IDENTIX
SECURID
NTS
RADIUS

Selected Methods:
CYBERSAFE

5. In the Available Methods list, select CYBERSAFE.
6. Move CYBERSAFE to the Selected Methods list by clicking the right-arrow button [>].
7. Arrange the selected methods in order of desired 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 CYBERSAFE to be the first service used, put it at the top of the list.

8. Click the Other Params tab.



Authentication Service: CYBERSAFE

GSSAPI Service:

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
```

Note: You must insert the principal name, using the format described in Task 4: Configure a Service Principal for an Oracle Server.

---

11. Choose File > Save Network Configuration.

The `sqlnet.ora` file updates 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:

```
REMOTE_OS_AUTHENT=FALSE
```

> **Note:** Setting REMOTE_OS_AUTHENT to TRUE can allow a security breach because it allows someone using a non-secure protocol, such as TCP, to perform an operating system-authorized login (formerly referred to as 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 server after modifying the configuration files to enable the changes.

> **More Information:** For information on how to restart the Oracle server, see the operating system-specific documentation and *Oracle8i Administrator's Guide*.

Task 10: Create a CyberSafe User on the Authentication Server

In order 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 actual programs that are run. 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.

The command prompts and responses that you enter are as follows:

```
#
kdb5_edit
kdb5_edit:
ank cyberuser
Enter password:
password
Re-enter password for verification:
password
kdb5_edit:
quit
```

More Information: For information on creating the realm, see "Related Publications" in the Preface.

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

Run SQL*Plus to create the Oracle user and enter the following commands on the Oracle server machine:

```
SQL> CONNECT INTERNAL;
SQL> CREATE USER "USNERNAME" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "USERNAME";
```

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

Note: When you create the Oracle user, the name must be in uppercase and double-quoted.

In the following example, OS_AUTHENT_PREFIX is set to null ("").

```
SQL> CREATE USER "JDOE" IDENTIFIED EXTERNALLY
SQL> GRANT CREATE SESSION TO "JDOE"
```

More Information: See *Oracle8i Administrator’s Guide*.

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

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

1. Enter the following

   ```
   % kinit user_name
   ```

2. Enter the password for CYBERUSER@US.ORACLE.COM. The password is not echoed to the screen.
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/SOMECO.COM@SOMECO.COM
11-Aug-99 16:29:51   12-Aug-99 00:29:21   oracledbserver.someco.com@SOMECO.COM
```

Task 13: Connect to an Oracle Server Authenticated by CyberSafe

After running `kinit` to get an initial ticket, users can connect to an Oracle 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 Net8 service name.

For example:

```
% sqlplus /@npddoc_db
```

> **More Information:** See Chapter 1, "Introduction to Oracle Advanced Security" and *Oracle8i Distributed Database Systems*.

Troubleshooting the Configuration of the CyberSafe Authentication Adapter

This section lists 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 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 machines 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 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 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 contains information on how to configure Oracle for use with Kerberos authentication and to configure Kerberos to authenticate Oracle users.

This chapter covers the following topics:

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

Enabling Kerberos Authentication

Kerberos authentication is enabled by performing the following tasks, each of which is described in this section.

Perform the following tasks in the order listed below.

Task 1: Install Kerberos

Task 2: Configure a Service Principal for an Oracle Server

Task 3: Extract a Service Table from Kerberos

Task 4: Install an Oracle Server and an Oracle Client

Task 5: Install Net8 and Oracle Advanced Security

Task 6: Configure Net8 and Oracle

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 machine that functions as the authentication server

More Information: For information on how to install Kerberos, see "Related Publications" in the Preface.

Task 2: Configure a Service Principal for an Oracle Server

For the Oracle Server to validate the identity of clients that authenticate themselves using Kerberos, you must first create a service principal for Oracle.

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` typically the fully-qualified name of the machine on which Oracle is running.

`REALM` the domain of the server. REALM must always be uppercase, and is typically the DNS domain name.

> **Note:** The utility names in this section are actual programs that are run. 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 machine on which Oracle is running is dbserver.someco.com, and the realm is SOMECO.COM, the principal name is as follows:

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

It is a common 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 the following 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 server/Kerberos client machine.

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

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 additional entries.

   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 machine as the Kerberos client, you can move it. If the service table is on a different machine 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`. If a different name is used, that name should be substituted for the default name.

4. Verify that the owner of the Oracle 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 allow a security breach.

Task 4: Install an Oracle Server and an Oracle Client

Install the Oracle server and client software.

> **More Information:** See the operating system-specific documentation.

Task 5: Install Net8 and Oracle Advanced Security

Install Net8 and Oracle Advanced Security on the Oracle server and Oracle client machines.

> **More Information:** See the platform-specific installation documentation.

Task 6: Configure Net8 and Oracle

Configure Net8 on the Oracle server and client.

> **More Information:** See the operating system-specific documentation. See also the *Net8 Administrator's Guide*.

Task 7: Configure Kerberos Authentication

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

- Configure Kerberos on the Client and on the Server
- Set the Initialization Parameters
- Optionally Set sqlnet.ora Parameters

Configure Kerberos on the Client and on the Server

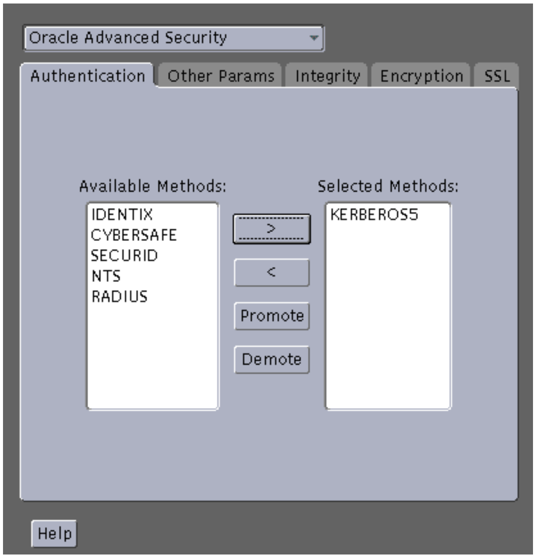
Perform the following steps to configure Kerberos authentication service parameters on the client and on the server:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Authentication tab.



Available Methods:
IDENTIX
CYBERSAFE
SECURID
NTS
RADIUS

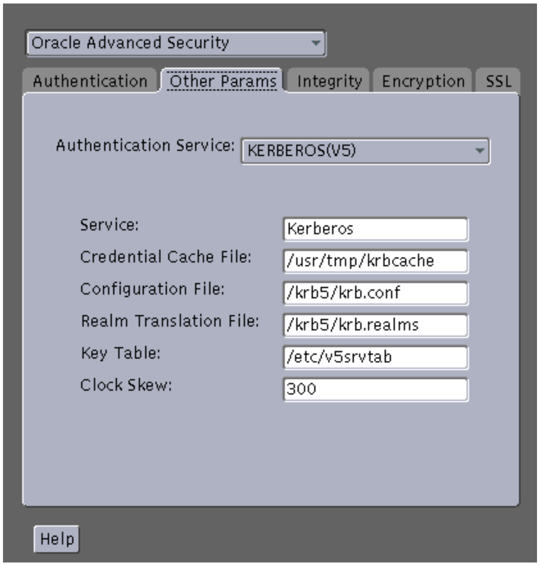
Selected Methods:
KERBEROS5

5. From the Available Methods list, select KERBEROS5.

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

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. Click the Other Params tab.



Authentication Service: KERBEROS(V5)

Service: Kerberos
Credential Cache File: /usr/tmp/krbcache
Configuration File: /krb5/krb.conf
Realm Translation File: /krb5/krb.realms
Key Table: /etc/v5srvtab
Clock Skew: 300

9. From the Authentication Service list, select KERBEROS(V5).

10. In the Service field, enter `kerberos` in lowercase. You must provide the value for this parameter. When you do this, the other fields are enabled.

    This field specifies the name of the service Oracle uses to obtain a Kerberos service ticket. Substitute a value for the kservice part of the service name.

11. Optionally provide values for the following fields:

    - Credential Cache File
    - Configuration File
    - Realm Translation File
    - Key Table
    - Clock Skew

    More Information: Net8 Assistant online help and "Optionally Set sqlnet.ora Parameters" in this section for more information about the fields and the parameters they configure.

12. Choose File > Save Network Configuration.

    The `sqlnet.ora` file updates with the following entries:

    ```
    SQLNET.AUTHENTICATION_SERVICES=(KERBEROS5)
    SQLNET.AUTHENTICATION_KERBEROS5_SERVICE=kservice
    ```

Set the Initialization Parameters

Perform the following steps 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 allow a security breach, because it allows someone using a non-secure protocol, such as TCP, to perform an operating system-authorized login (formerly referred to as 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 the value of OS_AUTHENT_PREFIX be set to null as follows:

   ```
   OS_AUTHENT_PREFIX=""
   ```

   Setting OS_AUTHENT_PREFIX to a null value overrides the default value of OPS$.

Optionally Set sqlnet.ora Parameters

In addition to the above required parameters, you can optionally set the following `sqlnet.ora` parameters on the client or on the 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`.

Note: You can also set this parameter by using the KRB5CCNAME environment variable.

The value set for the SQLNET.KERBEROS5_CC_NAME parameter in the `sqlnet.ora` file takes precedence over the value set in the KRB5CCNAME environment variable.

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 server. An Oracle 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=*[Boolean = True/False]*

Description: This parameter specifies whether the new MIT Kerberos configuration format will be used. If the value is set to Yes, 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 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.

It is assumed that the realm already exists.

Note: The utility names in this section are actual programs that you run. 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 not echoed>
Re-enter password for principal: "krbuser@SOMECO.COM": <password not echoed>
kadmin.local: exit
```

More Information: See "Related Publications" in the Preface.

Task 9: Create an Externally-Authenticated Oracle User

Run SQL*Plus on the Oracle 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 must be in uppercase and double-quoted, such as "KRBUSER@SOMECO.COM".

```
SQL> CONNECT INTERNAL;
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. You should be on the Oracle client before running the following commands:

```
% okinit -f
Password for krbuser@SOMECO.COM:password
```

Utilities for the Kerberos Authentication Adapter

Table 6–1 describes utilities that are shipped with the Oracle Kerberos authentication adapter.

Table 6–1 Oracle Kerberos Adapter Utilities for the Client

Command	Description
`okinit`	Gets an initial ticket
`oklist`	Displays a list of currently-owned tickets
`okdstry`	Removes all tickets from the credentials cache

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

> **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 inadvertently used.

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 6–2.

Table 6–2 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 2wld6h20m30s` 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

Users can run the `oklist` utility to display the list of tickets they hold.

The options available with `oklist` are listed in Table 6–3.

Table 6–3 Options for the oklist Utility

Option	Description
`-f`	Show flags with credentials. Relevant flags are I, credential is a ticket-granting ticket, F, credential is forwardable, 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 Cache File

Use the `okdstry` utility to remove credentials from the credentials cache file as follows

To use the `okdstry` utility, enter the following:

```
$ okdstry -f
```

Table 6–4 provides information on the `-f` command option.

Table 6–4 Option for the okdstry Utility

Option	Description
`-f`	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 Server Authenticated by Kerberos

You can now connect to an Oracle 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 Net8 service name. For example:

```
$ sqlplus /@oracle_dbname
```

More Information: For information on external authentication, see Chapter 1 and *Oracle8i Distributed Database Systems.*

Troubleshooting the Configuration of Kerberos Authentication

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 server side has a service name that corresponds to a service known by Kerberos.
- Check that the clocks on all machines 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 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 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 SecurID Authentication

This chapter describes how to configure and use SecurID authentication with the Oracle server and clients. It assumes that you are familiar with the Security Dynamics ACE/Server and that the ACE/Server is installed and running.

This chapter covers the following topics:

- System Requirements
- Known Limitations
- Enabling SecurID Authentication
- Creating Users for SecurID Authentication
- Using SecurID Authentication
- Troubleshooting the Configuration of SecurID Authentication

> **More Information:** See "Related Publications" in the Preface for a list of documents related to SecurID authentication.

System Requirements

The requirements for using SecurID authentication included in Oracle Advanced Security are as follows:

- Net8
- Oracle 8.0.3 or higher
- ACE/Server 1.2.4 or higher
- The Oracle server must be running the UDP/IP and TCP/IP protocols to enable communication with the ACE/Server. Even if the client uses SQL*Net or Net8 to connect to Oracle, Oracle needs UDP to connect to the ACE/Server.

Known Limitations

Because SecurID card codes can be used only once, SecurID authentication does not support database links, also known as proxy authentication.

When using SecurID authentication, password encryption is disabled. This means that the SecurID card code and, if you use standard cards, the PIN, are sent over to the Oracle server in clear text. This can be a security problem. Consequently, Oracle Corporation recommends that you enable Oracle Advanced Security encryption, which ensures that the PIN is encrypted when it is sent to the Oracle server.

> **More Information:** See Chapter 2, "Configuring Data Encryption and Integrity".

Enabling SecurID Authentication

Enable SecurID authentication by performing the following tasks:

Task 1: Register Oracle as a SecurID Client

Task 2: Install Oracle Advanced Security

Task 3: Ensure that Oracle Can Find the Correct UDP Port

Task 4: Configure Oracle as a SecurID Client

Task 5: Configure SecurID Authentication

Task 1: Register Oracle as a SecurID Client

Register the machine on which the Oracle Server resides as a SecurID client with the ACE server. You can do this with the Security Dynamics tool `sdadmin`. To create a client, go to the Client menu and choose Create Client on the ACE/Server 1.2.4 or Add Client on the ACE/Server 2.0.

> **More Information:** See the Security Dynamics ACE/Server Instruction manual, version 1.2.4, or to the Security Dynamics ACE/Server version 2.0 Administration manual for more detailed information.

Task 2: Install Oracle Advanced Security

Install Oracle Advanced Security on the Oracle server and Oracle client when you install Oracle8*i* using the Oracle Installer.

> **More Information:** See the platform-specific installation instructions.

Task 3: Ensure that Oracle Can Find the Correct UDP Port

Verify that the ACE/Server, the Oracle server, and the Oracle Advanced Security are installed.

Ensure that the Oracle server can discover the correct UDP port for contacting the ACE/Server. Port numbers are typically stored in a file called `services`. On UNIX-based operating systems, the file is typically located in the `/etc` directory. If you are using Network Information Services (NIS) as a naming service, ensure that the services map contains the correct entries for SecurID.

> **Note:** You can verify which port the ACE server is using by running the Security Dynamics tool `Kitconts`, for ACE/Server 1.2.4, or `sdinfo`, for ACE/Server 2.0.

Task 4: Configure Oracle as a SecurID Client

This section provides separate instructions for Windows and UNIX-based platform for the following options:

- Windows NT and Windows 95/98 Platforms
- UNIX-Based Platform and ACE/Server Release 1.2.4
- UNIX-Based Platforms and ACE/Server Release 2.0

Windows NT and Windows 95/98 Platforms

You need the following from the SecurID administrator:

- `SDCONF.REC` file present in the root drive `\VAR\ACE`
- port numbers and service names present in the Windows NT `SERVICES` file

UNIX-Based Platform and ACE/Server Release 1.2.4

Install the SecurID configuration files on the Oracle server machine. You can obtain the files from any other SecurID client or from the machine that runs the ACE/Server.

These files are typically stored in `/var/ace`. Create this directory on the Oracle server machine and copy the configuration files to it. At the minimum, you need the `sdconf.rec` file. The configuration files are used by both Oracle and the standard SecurID tools. Because the SecurID tools run setuid root, there can be a problem with the access permissions on the directory `/var/ace` and the files in this directory. Ensure that the owner of the `oracle` executable, such as the user "oracle8," is able to read all the files in `/var/ace` and can create new files in this directory.

Caution: Do not attempt to overcome this by running Oracle setuid root. It is not necessary, and it is dangerous to do so.

There are two methods for configuring Oracle as a SecurID client without compromising security. Both methods work, but Oracle Corporation recommends that you use Method #1. Both methods allow you to use Oracle with SecurID authentication and still continue using the other SecurID tools.

Method #1

The owner of the `oracle` executable should also own the `/var/ace` directory and the files in `/var/ace`. For example, if the owner of the `oracle` executable is the user "oracle8," perform the following steps as root:

```
# chown oracle8 /var/ace
# chmod 0770 /var/ace
# chown oracle8 /var/ace/*
# chmod 0660 /var/ace/*
```

Method #2

The other option is to have root own the `/var/ace` directory and the files in `/var/ace`, but give the Oracle group read and write access. If the Oracle group is dba, execute the following commands as root:

```
# chown root /var/ace
# chmod 0770 /var/ace
# chgrp dba /var/ace
# chown root /var/ace/*
# chmod 0660 /var/ace/*
# chgrp dba /var/ace/*
```

UNIX-Based Platforms and ACE/Server Release 2.0

The VAR_ACE environment variable is not supported. You have to store the configuration data in the `/var/ace` directory. If you currently have the ACE configuration data in a different location, create a symbolic link using the following command:

```
# ln -s $VAR_ACE /var/ace
```

Oracle needs to be able to read and write the ACE configuration data. This data is stored in the directory `/var/ace`, or $VAR_ACE if you use the symbolic link shown above.

Whether Oracle can read the configuration data depends on how the ACE client software is installed on the Oracle server. During the installation of the ACE client software, specify which administrator should own the configuration files.

Attention: Whether you use Method 1 or Method 2, described below, ensure that you do not install Oracle as root.

Method #1

If root is the owner of the ACE server configuration data files, change the UNIX file permissions so that the owner of the `oracle` executable can read and write to these files. For example, the following commands give Oracle access to the files, and all the Security Dynamics tools that run as setuid root can still access the files.

```
# chown oracle8 /var/ace
# chown oracle8 /var/ace/*
# chmod 0770 /var/ace
# chmod 0660 /var/ace/*
```

If the environment variable VAR_ACE is set to a different location than `/var/ace`, you should instead execute the following commands:

```
# ln -s $VAR_ACE /var/ace
# chown oracle8 $VAR_ACE
# chown oracle8 $VAR_ACE/*
# chmod 0770 $VAR_ACE
# chmod 0660 $VAR_ACE/*
```

Method #2

If the ACE files are not owned by root, you have the following options:

- Install the ACE client or server and Oracle under the same UNIX account.

  You must install the ACE software as root, but you can specify which administrator should own the files. Specify the same user as the owner of the Oracle executable, typically "oracle8".

- Add the owner of the Oracle executable to the ACE administrators' group.

Note: Ensure that the owner of the `oracle` executable remains a member of the DBA group; otherwise you cannot control the database.

For the change to take effect, perform the following steps:

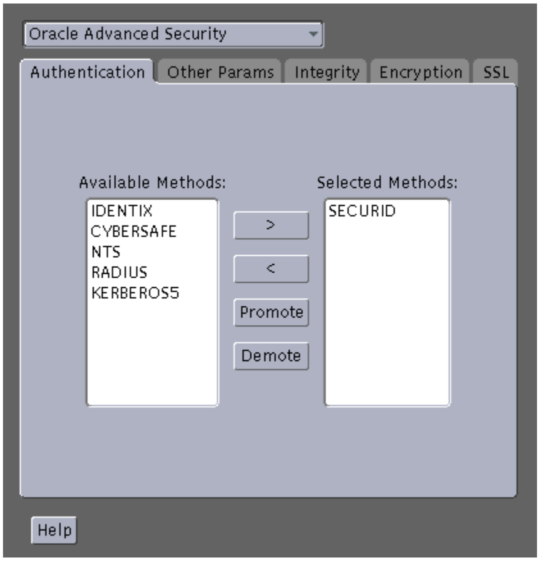
1. Log out and log in again as the Oracle owner.
2. Restart the listener.
3. Restart the Oracle server.

Task 5: Configure SecurID Authentication

Perform the following steps to configure SecurID authentication service:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the Authentication tab.



Oracle Advanced Security

Authentication | Other Params | Integrity | Encryption | SSL

Available Methods:

IDENTIX
CYBERSAFE
NTS
RADIUS
KERBEROS5

Selected Methods:

SECURID

5. From the Available Methods list, select SECURID.
6. Move SECURID to the Selected Methods list by clicking the right-arrow button [>].
7. Arrange the selected methods in order of desired 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, for SECURID to be the first service used, move it to the top of the list.

8. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entries:

   ```
   SQLNET.AUTHENTICATION_SERVICES=(SECURID)
   ```

Creating Users for SecurID Authentication

You create users for SecurID authentication by performing the following steps:

- Task 1: Assign a Card Using Security Dynamics sdadmin Program
- Task 2: Create an Oracle Server Account for the User
- Task 3: Grant the User Database Privileges

Task 1: Assign a Card Using Security Dynamics sdadmin Program

When the `sdadmin` tool asks for a login name when creating a new user, enter the same name you will use later to create the Oracle user.

> **More Information:** See the Security Dynamics documentation listed in "Related Publications" in the Preface.

If you want the user to be able to specify a new PIN to the card using the Oracle tools, choose the option that allows the user to make up his or her own PIN. If you do not allow this, the user will have to use the Security Dynamics tools to generate a PIN if the card is in new-PIN mode. Activate the user on the Oracle server. The Oracle server should already be registered as a SecurID client.

Task 2: Create an Oracle Server Account for the User

You can create an Oracle server account using SQL*Plus connected as a user with the CREATE USER database privilege. Enter the following to create an account:

```
SQL> CONNECT system/manager
SQL> CREATE USER os_authent_prefix username IDENTIFIED EXTERNALLY
```

The OS_AUTHENT_PREFIX Oracle initialization parameter has a default value of OPS$. The user name should be the same as the name you assigned to the card in Task 1: Assign a Card Using Security Dynamics sdadmin Program.

Note: Because user names can be long and Oracle user names are limited to 30 characters, Oracle Corporation strongly recommends that OS_AUTHENT_PREFIX be set to a null value as follows:

```
OS_AUTHENT_PREFIX=""
```

At this point, an Oracle user with *username* should not yet exist.

For example, suppose you have assigned a card to the user king, and OS_AUTHENT_PREFIX has been set to a null value (""). At this point, create an Oracle user account as follows:

```
SQL> CREATE USER king IDENTIFIED EXTERNALLY;
```

Task 3: Grant the User Database Privileges

Grant the user the required database privileges. At a minimum, the user should be granted the CREATE SESSION privilege, as in the following example:

```
SQL> GRANT CREATE SESSION TO king;
```

The user king can now connect to Oracle using the appropriate SecurID card.

More Information: For information on how to log on to an Oracle server after SecurID authentication has been installed and configured, see "Logging On to the Oracle Server" in this chapter.

Using SecurID Authentication

This section describes how to use SecurID authentication with the Oracle client tools. It assumes that you are already familiar with SecurID concepts, and that you have configured Oracle for use with the SecurID authentication.

This section contains the following topics:

- Logging On to the Oracle Server
- Assigning a New PIN to a SecurID Card
- Logging On to the Oracle Server

More Information: See "Related Publications" in the Preface.

Before using SecurID authentication to verify passwords, ensure that the following tasks have been completed:

- The SecurID authentication adapter has been installed and linked into the Net8 configuration.
- Oracle has been configured for use with the ACE/Server (that is, it can act as a SecurID client).
- The client and server have been configured with the necessary parameters so that database passwords can be verified by the central SecurID authentication server.
- Users have been configured for use with the SecurID authentication as described in "Task 5: Configure SecurID Authentication" in this chapter.

Logging On to the Oracle Server

SecurID authentication allows users to log on to the Oracle server with the PASSCODE that is generated by the SecurID card. The PASSCODE replaces the password in the Oracle connect statement.

There are two types of SecurID cards:

- Standard (model SD200)
- PINPAD (model SD520)

Depending on the type of card, you enter the PIN either:

- directly into the card, or
- as part of the Oracle connect statement

Using Standard Cards

The standard cards generate and display a PASSCODE. When logging in to Oracle, specify the user name, PIN, and current PASSCODE as follows:

```
sqlplus username/<pin><passcode>@net_service_name
```

For example, if the card is assigned to user king, the PIN is 3511, and the card shows the number 698244, log into Oracle using SQL*Plus as follows:

```
% sqlplus king/3511698244@oracle_database
```

or

```
% sqlplus king@oracle_database
% enter password: 3511698244
```

> **Note:** The Security Dynamics tools support the following characters as delimiters between the PIN and the PASSCODE:
>
> " " <tab> \ / ; :
>
> Do not use these characters, because Oracle interprets these characters differently.

Using PINPAD Cards

If you have a PINPAD card, you first have to enter the PIN on the card and generate a new PASSCODE. Use the PASSCODE to connect to Oracle as follows:

```
sqlplus username/passcode@net_service_name
```

For example, if the card is assigned to user king, first generate a PASSCODE by entering the PIN on the PINPAD card as described in the Security Dynamics documentation.

For example, if the generated PASSCODE is 698244, connect to Oracle using SQL*Plus as follows:

```
% sqlplus king/698244@oracle_dbname
```

Assigning a New PIN to a SecurID Card

If you are logging in for the first time or the administrator has put the card in the new-PIN mode, you must assign a PIN to the card. You can tell that this is the case if, while trying to connect to Oracle, you receive the following error message:

ORA-12681 "Login failed: the SecurID card does not have a pincode yet"

To assign a PIN to a card you connect to the Oracle Server using a special syntax. First select a PIN, which is typically four to eight digits long. Depending on the type of SecurID card you have, you might be able to use letters as well.

Old PIN Cleared

If you have cleared the old PIN, use the following the syntax while connecting to the Oracle database:

```
sqlplus username/+new_pin+tokencode@oracle_dbname
```

Note: You must add the two plus (+) characters in the connect string, because they tell Oracle that this is an attempt to assign a PIN to the card. Also, they separate the new PIN from the passcode.

You must also enclose the PIN / passcode combination in double quotes. Some Oracle tools such as SQL*Plus truncate the password string (PIN / passcode) just before the + character. Surrounding the password string (PIN / passcode) in double
quotes (" ") prevents the password string from being truncated.

For the tokencode, enter the card code that is currently displayed on the SecurID card's LCD. If you have a PINPAD card, do not enter the PIN on the card.

For example, if the card is assigned to user king, the new PIN is 45618, and the SecurID card currently displays number 564728, enter the following:

```
% sqlplus king/"+45618+564728"@oracle_dbname
```

Old PIN Not Cleared

If the old PIN was not cleared, use the following syntax while connecting to the database. Otherwise, the administrator must select the new PIN for you.

```
sqlplus username/+new_pin+old_pintokencode@oracle_dbname
```

For the tokencode, enter the card code that is currently displayed on the SecurID card. If you have a PINPAD card, do not enter the PIN on the card.

If the new PIN is accepted, you are connected to Oracle. The next time you want to connect to Oracle you should use the procedure described in "Logging On to the Oracle Server" in this chapter. If the new PIN is rejected, you receive the following error:

ORA-12688 "Login failed: the SecurID server rejected the new pincode"

Possible Reasons for PIN Rejection:

The following are possible reasons for why a PIN is rejected:

- The new PIN is less than 4 or more than 8 characters long.
- The PIN contains invalid characters. Valid characters are numeric digits, and for some SecurID cards, the letters a through z.
- You are not allowed to make up your own PIN. The Security Dynamics ACE/Server can be configured in such a way that you cannot make up your own PIN. If this is the case, you will have to use one of the Security Dynamics tools to generate a new PIN for your card.

Logging on When the SecurID Card is in "Next Code" Mode

As an additional safety step, the ACE/Server sometimes asks for the next card code, to ensure that the person who is trying to log in actually has the card in his or her possession. This is the case if you get the following error message when you try to log into Oracle:

ORA-12682, "Login failed: the SecurID card is in next PRN mode"

The next time you want to log on Oracle, you must specify the next two card codes. The syntax you use to log into Oracle depends on the kind of SecurID card you have, Standard or PINPAD.

Logging on with a Standard Card

If you have a standard card, specify the following:

1. the PIN
2. the current card code
3. a plus (+) character and the next card code

Steps 1, 2, and 3 above replace the password. The + character is important, because it separates the first card code (passcode) from the second one. Use the following syntax:

```
sqlplus <username>/ "pincodepasscode+next_passcode"@<net_service_name>
```

> **Note:** You must enclose the PIN / passcode / next passcode combination in double quotes. Some Oracle tools such as SQL*Plus truncate the password combination just before the plus (+) character. Surrounding the PIN and passcode in double quotes ("") prevents the password combination from being truncated.

For example, if the card is assigned to user king, the PIN is 3511, and the card first shows the number 98244 and the next number is 563866, enter the following:

```
% sqlplus king/"3511698244+563866"@oracle_database
```

This connects you to the Oracle server and puts the card back into normal mode. The next time you want to log in to the Oracle server, use the procedure described in "Logging On to the Oracle Server" in this chapter.

Logging on with a PINPAD Card

If you have a PINPAD card, perform the following steps to log on to the Oracle server:

1. Enter the PIN on the card to generate the first PASSCODE.
2. Clear the card's memory by pressing P and wait for the next PASSCODE.
3. Log on to the Oracle server with the two passcodes, separated by a + character as follows:

   ```
   sqlplus username/ "<first passcode+second passcode"@net_service_name
   ```

For example, if the card is assigned to user king, perform the following steps:

1. Enter the PIN on the PINPAD card to generate a passcode, such as 231003.
2. Clear the card's memory. The next displayed number might be 831234.
3. To log in, use the following syntax, entering the two passcodes generated in steps 1 and 2:

   ```
   % sqlplus king/"231003+831234"@oracle_dbname
   ```

This connects you to Oracle and puts the card back into normal mode. The next time you want to log in to Oracle, use the procedure described in"Logging On to the Oracle Server" on page 7-10.

Troubleshooting the Configuration of SecurID Authentication

If you experience problems while configuring SecurID authentication, verify the following:

- The services map should have an entry for the Security Dynamics ACE server. The service name is typically securid, but the SecurID administrator can choose any name.

  Use the SecurID tool `kitconts` (for ACE/Server 1.2.4) or `sdinfo` (for ACE/Server 2.0) to verify the name of the authentication service and the port numbers that SecurID is expecting to use. Verify that these port numbers match those in `/etc/services`, or the services map if you are using NIS.

  ACE/Server release 1.2.4 only: Verify that the `/var/ace/sdconf.rec` file is present on the machine running the Oracle server. Also verify that the permissions on the `/var/ace/sdconf.rec` file and the directory `/var/ace` are set so that the Oracle process can read and write in the directory.

  ACE/Server release 2.0 only: Make sure the ACE configuration data is in the `/var/ace` directory. Use of the VAR_ACE environment variable is not supported. Also make sure that the owner of the oracle executable can read and write the files in this directory.

- Check to see if the Oracle server machine is registered as a SecurID client. You can do this by using the Security Dynamics tool `sdadmin`.

- The user who is trying to connect to Oracle should be activated on the Oracle server, either as a direct user or as part of a group of users. Verify this using the SecurID tool `sdadmin`.

- Security Dynamics has developed a few logging facilities that can help you find problems. By using `sdadmin`, you can see a log of the recent system activities, including failed authentication with the reason for the failure. You can also use `sdlogmon` to get a similar log listing.

- Turn on tracing by adding the following line to the `sqlnet.ora` file on the Oracle side:

  ```
  trace_level_server = admin
  ```

Turning tracing on at the client side is less informative, because all interaction between the Oracle server and the ACE server happens at the Oracle server side of the Net8 connection. Be sure to turn off tracing when you have completed your check.

- Ensure that the user has been created in the Oracle database as an externally-identified user with the correct prefix (which defaults to OPS$). When connected as system, enter the following:

  ```
  SQL> SELECT * FROM all_users;
  ```

  to get a list of all database users.

- When you connect to Oracle as a non-externally identified user, the SecurID log file will indicate a warning. For example, if you connect as "system" as follows:

  ```
  sqlplus system/manager@oracle_dbname
  ```

  the SecurID log file displays the following:

  ```
  03/24/99   10:04  User not on client machinename
  ```

  This is not an error. Since the Oracle client and server negotiated to use SecurID because of the SQLNET.AUTHENTICATION_SERVICES line in the `sqlnet.ora` file, Oracle contacts the ACE/Server to validate "system." When validation fails, Oracle validates the password internally. If the password is valid, you are able to connect.

  The only way to eliminate the warning message is to disable SecurID authentication. To do so, change the `sqlnet.ora` file on the Oracle client to the following:

  ```
  SQLNET.AUTHENTICATION_SERVICES=(NONE)
  ```

  Setting this parameter to NONE disables the SecurID authentication adapter and you can no longer connect to Oracle using the SecurID card.

8

Configuring Identix Biometric Authentication

This chapter contains information on how to configure Oracle for use with Identix biometric authentication. It covers the following topics:

- Overview
- Architecture of the Biometric Authentication Service
- Prerequisites
- Enabling Biometric Authentication
- Administering the Biometric Authentication Service
- Authenticating Users with a Biometric Authentication Service
- Troubleshooting

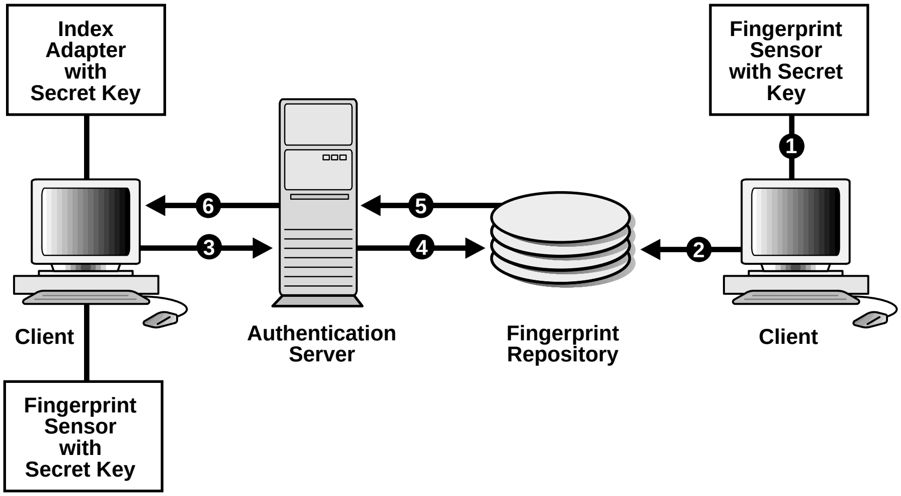
Overview

The Biometric Authentication Service uses Identix Biometric Authentication to provide tamper-proof biometric authentication of users using secret-key MD5 hashing, centralized management of biometrically identified users, and centralized management of those database servers that authenticate biometrically identified users.

This section describes how the Biometric Authentication Service works in a client-server environment.

Figure 8–1 shows the components and the configuration of the Biometric Authentication Service.

Figure 8–1 Typical Biometric Authentication Service Configuration


Index Adapter with Secret Key
Fingerprint Sensor with Secret Key
1
6
5
3
4
2
Client
Authentication Server
Fingerprint Repository
Client
Fingerprint Sensor with Secret Key

The Fingerprint Repository has one administrator who is responsible for enrolling multiple users’ fingerprint templates and defining the DEFAULT policy that is in force for all databases that subscribe to the fingerprint server for authentication.

The Fingerprint Security Service Administrator uses a desktop fingerprint scanner to read user fingerprints, convert them into fingerprint templates, and send them with measured accuracies to the Biometric Authentication Service. The Biometric Authentication Service stores the fingerprint templates in the Fingerprint Repository, an Oracle database. The measured accuracy of a fingerprint is an estimate of how reliable a comparison can be made between the stored fingerprint template and the user's fingerprint that is scanned later for authentication. The enrollment quality is expressed as a percent score between 0 and 100. For example, a user may have an enrollment quality of 72 percent.

- The Fingerprint Security Service Administrator also defines one security policy named DEFAULT for all of the database servers that accept biometrically identified users. The security policy is enforced for all clients serviced by that database server. It contains a secret key and three types of threshold levels for fingerprints: verification, false finger, and high security.
- At the client, before any authentication can occur, the Fingerprint Security Service Administrator stores the secret key in the fingerprint sensor for each client. The secret key stored in the fingerprint sensor is compared against the secret key stored in the security policy.
- At the client, in response to the user's request for authentication, the database server enforces on the client the set of values that it obtains from the DEFAULT security policy in its fingerprint server. The threshold levels (values) are as follows:
  - verification threshold
  - false finger threshold
  - high security threshold

  More information: See the Identix documentation for detailed information on the threshold levels.
- At the client, the biometric authentication service passes the user's fingerprint template and the threshold values to the sensor. The user's fingerprint is then scanned and the verification result, fingerprint template, secret key, and threshold values are used to generate a hash. The hash is sent to the server side adapter for comparison with the hash generated by the server side adapter.

Architecture of the Biometric Authentication Service

The Biometric Authentication Service consists of the following modules:

- The Biometric Manager, which the administrator uses to enter the security policy and fingerprints. In the remainder of this document, the Biometric Manager will also be referred to as the manager.
- The Biometric Authentication Server (fingerprint repository), which stores the security policies and fingerprint templates, is a specially configured version of

an Oracle database server. In this guide, the Biometric Authentication Server is also referred to as the authentication server.

- The Identix authentication adapters are used on both the clients and the database servers to communicate biometric authentication data between the authentication server and the clients in order to authenticate a database user.

Both the manager and the client-side adapter interface with Identix products: TouchSafe II Software Libraries, TouchSafe II Hardware Interface, TouchSafe II Desktop Sensor, TouchSafe III software libraries, and TouchSafe III desktop sensor.

> **More Information:** For a list of Identix documentation that describes these Identix products, see "Related Publications" in the Preface.

Administration Architecture

The Fingerprint Security Server Administrators use the manager to scan user fingerprints, measure the accuracy of the fingerprints, and establish security policies for database servers. The manager sends this information to the authentication server, which stores the data in the repository.

The administrator, or someone who can be trusted, uses the Identix TouchSafe II or TouchSafe III software to store the secret key on the TouchSafeII or TouchSafe III device. This key must match the key stored in the DEFAULT security policy before authentication can occur.

Authentication Architecture

Each user who wants to use the system must place a fingerprint on a TouchSafe II or TouchSafe II Desktop Sensor. The client-side adapter sends an authentication request to the server-side adapter which uses the previously enrolled fingerprint stored in the authentication server for comparison. For each authentication request from a client, the authentication server retrieves and sends the user’s fingerprint and the database server’s security policy back to the client-side adapter via the server-side adapter.

The user’s authentication request causes the client-side Oracle Advanced Security Identix authentication adapter to send the request to the server side biometric authentication adapter, which looks up the user’s fingerprint in the authentication server, which returns the stored fingerprint and the associated security policy.

Using threshold level values from the associated security policy, the client side adapter uses the TouchSafe II Software Libraries to set threshold values on the

TouchSafe II Desktop Sensor. It then prompts for the placing of the user’s finger on the TouchSafe II Desktop Sensor. The adapters on the client and the database server work together to compare the user’s fingerprint, the secret key, and the threshold levels against the administrator-entered security policy stored in the authentication server repository. If this data matches, the user is authenticated.

Prerequisites

- The Windows NT machine that is to become the manager PC must be running the Oracle Enterprise Manager 2.0 or above.
- Each Windows NT or Windows 95 machine that is to become a client PC must be running Net8.
- The authentication server and each database server must be running Oracle8 release 8.0.3 or higher or Oracle8*i*.
- Before proceeding with the Oracle Advanced Security installation, you must make sure that each Windows NT and Windows 95 client has Net8 connectivity with its associated database server.

Installing the TouchSafe II Encrypt Device Driver for Windows NT

The Biometric Manager installation process automatically installs the necessary TouchSafe II software and automatically configures the device if requested.

If during the installation of the Biometrics Manager, you chose not to allow the installer to set up your Identix TouchSafe II Device Driver, you can configure it manually as follows.

1. Change directory to `ORACLE_HOME\identix`.
   - If you are using the default IO port number 280 and the default Windows NT directory, go to Step 4.
   - If you are not using the default IO port number 280, go to Step 2.
   - If you are not using the default Windows NT directory `c:\winnt35\sytem32\drivers`, go to Step 3.
2. Modify the IoPortAddress parameter in `etsiint.ini` to the current TouchSafe II Encrypt I/O port setting. For example:

```
IoPortAddress = REG_DWORD 0x00000360  for I/O port 0x360
```

3. Modify the Windows NT directory setting in `etsiint.bat` with the Windows NT directory. For example:

```
copy etsiint.sys c:\winnt\system32\drivers
 -> copy etsiint.sys path\drivers
```

4. Run the batch file `etsiint.bat`.
5. Use the SetKey utility in the Identix demo program to set a hash key in Hex. Set the key to 7GF87SRG for example (do not use this value). Ensure that the hash key matches exactly the one set in the DEFAULT Security policy.
6. Re-boot the system, and the device driver will start to work.
7. To make sure the device driver is running, check the Device Control Panel after re-boot. The device ETSIINT should be started already.

Biometric Manager PC

Perform the following steps on the manager PC:

1. Install Oracle Enterprise Manager on both the Oracle server and the Oracle client.
2. Install the Identix hardware and the Identix driver firmware and configure the Identix variables and devices.

   More Information: See the Identix Readme file.

3. Install and test the Identix TouchSafe II (Encrypt) 2.0 or TouchSafe III.

   More Information: See "Installing the TouchSafe II Encrypt Device Driver for Windows NT"in this chapter and the platform-specific installation documentation.

   Follow the instructions in the Identix manual to verify that the module works with the Identix demonstration program. The demonstration program must work on the PC before any other Oracle products can be loaded onto the PC. See the Identix Readme file for additional information.

Client PC

Perform the following steps on each client PC:

1. Install the Identix hardware and the Identix driver firmware and configure the Identix variables and devices. See the Identix Readme file for additional information.
2. Install and test the Identix TouchSafe II (Encrypt) 2.0 or TouchSafe III. Follow the instructions in the Identix manual to verify that the module works with the Identix demonstration program. The demonstration program must work on the PC before any other Oracle products can be loaded onto the PC.

   More Information: See "Installing the TouchSafe II Encrypt Device Driver for Windows NT" in this chapter and the platform-specific installation documentation.

3. Install the Oracle Advanced Security Identix authentication adapter.

   More Information: See the platform-specific documentation. See also the Identix Readme file.

Database Server

The biometric authentication adapter must be installed on each database that uses biometric services for its authentication. Install the biometric authentication adapter following the instructions in the platform-specific documentation.

Note: Do not install the adapter on the database storing the fingerprint repository.

Biometric Authentication Service

The Biometric Authentication Service is the database that stores both the user and fingerprint information. The database can be any Oracle 8.0.3 or later production database. It should be on a secure, trusted system with strict security and access controls. The Identix adapter should not be installed on this database.

Enabling Biometric Authentication

To configure the Biometric Authentication Service, perform the following tasks.

Task 1: Configure the Database Server that is to become the Authentication Server

Task 2: Configure Identix Authentication

Task 3: Establish a Net Service Name for the Fingerprint Repository Server

Task 4: Verify that the Address of the Database Server is Accessible to the Client

Task 5: Configure the Biometric Manager PC

Task 1: Configure the Database Server that is to become the Authentication Server

Perform the following steps to configure the database server:

1. Connect to the database server as SYSTEM / MANAGER (or whatever your system password is).
2. Test the connection by connecting as:

   ```
   ofm_admin/ofm_admin
   ```

Task 2: Configure Identix Authentication

Perform the following tasks to configure Identix authentication:

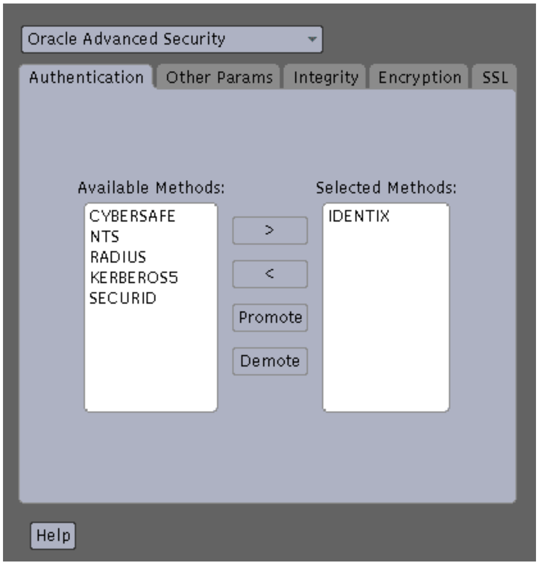
- Configure an Authentication Method and Fingerprint Server on the Client and Server
- Configure the User Name, Password, and Fingerprint Method
- Configure the Initialization Parameter File
- Configure the oracle.ini File

Unless otherwise indicated, you can configure Identix authentication either by using the Net8 Assistant, or by modifying the `sqlnet.ora` file with any text editor.

Configure an Authentication Method and Fingerprint Server on the Client and Server

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the Authentication tab.



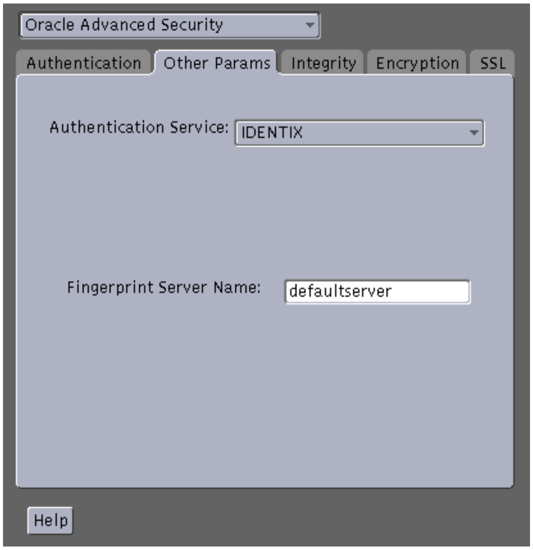
Oracle Advanced Security

Available Methods:
CYBERSAFE
NTS
RADIUS
KERBEROS5
SECURID

Selected Methods:
IDENTIX

5. From the Available Methods list, select IDENTIX.
6. Move IDENTIX to the Selected Methods list by clicking the right-arrow button [>].
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 IDENTIX to be the first service used, put it at the top of the list.

8. Click the Other Params tab.



Authentication Service: IDENTIX

Fingerprint Server Name: defaultserver

9. From the Authentication Service list, select IDENTIX.
10. In the Fingerprint Server Name box, enter the name of the fingerprint server you want to use.
11. Choose File > Save Network Configuration.

    The `sqlnet.ora` file updates with the following entries:

```
SQLNET.AUTHENTICATION_SERVICES=(IDENTIX)
SQLNET.IDENTIX_FINGERPRINT_DATABASE=SERVICE_NAME
```

Configure the User Name, Password, and Fingerprint Method

Use a text editor to set the following parameters in the sqlnet.ora file,

```
sqlnet.identix_fingerprint_database_user=ofm_client
sqlnet.identix_fingerprint_database_password=password
sqlnet.identix_fingerprint_method=oracle
```

where *username* is the well-known user name ofm_client, and *password* is the well-known password ofm_client.

Note: The `samples` directory contains a file that shows how to set these parameters.

Note: The ofm_client user name and password are set up by running `nauicat.sql`. You should not change ofm_client.

Configure the Initialization Parameter File

Add the following parameters to the initialization parameters file:

```
REMOTE_OS_AUTHENT = false
OS_AUTHENT_PREFIX = ""
```

Note: The local naming configuration file (`tnsnames.ora`) on the database server should contain the service name of the fingerprint repository. If they are on the same database, use the following with the service name:

```
(security=(authentication_service=none))
```

Configure the oracle.ini File

Set the USERNAME parameter in the Oracle section of the `oracle.ini` file. This parameter sets the name of the database user with which the client connects to the database.

Task 3: Establish a Net Service Name for the Fingerprint Repository Server

Establish a net service name for the fingerprint repository server.

More Information: See the *Net8 Administrator's Guide* for information on net service names

Task 4: Verify that the Address of the Database Server is Accessible to the Client

Verify that the address of the database server is accessible to the client.

More Information: See the *Net8 Administrator's Guide* for information on verifying the address of the database server.

Task 5: Configure the Biometric Manager PC

Configure the manager PC with a net service name to connect to the authentication server.

More Information: See *Net8 Administrator's Guide* for information on net service name configuration

Administering the Biometric Authentication Service

Perform the following tasks to administer the Biometric Authentication Service using the Biometric Manager.

More Information: See the Identix documentation.

Create a Hashkey on Each of the Clients:

Use the Identix Setkey utility to configure a hexadecimal hashkey on each of the clients, such as FF30EE. The key must be the same for each client and must match the DEFAULT Policy hashkey. This key can range from one to thirty-two hexadecimal digits.

Create a User for Biometric Authentication:

1. Use the Windows NT User Manager to create a user name on the client.

   The user name must match the name used in the next step.

2. On the database server, restart the database and create an Oracle server account for the user. Use SQL*Plus if using the Oracle Enterprise Manager or SQL*Plus connected as a user with the CREAT USER database privilege.

   Enter the following to create an account:

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

3. OS_AUTHENT_PREFIX is an Oracle server initialization parameter. The default value for OS_AUTHENT_PREFIX is OPS$. The user name in this step should match the user name created at the client. If you reset OS_AUTHENT_PREFIX parameter, you must stop and restart the database.

Note: Oracle user names are limited to 30 characters and user names can be long, so Oracle Corporation strongly recommends that OS_AUTHENT_PREFIX be set to a null value as follows:

```
OS_AUTHENT_PREFIX=""
```

Note: An Oracle user with a user name should not yet exist.

For example, if you create the user king, and set OS_AUTHENT_PREFIX to a null value (""), use SQL*Plus to create an Oracle user account using the following syntax:

```
SQL> CREATE USER king IDENTIFIED EXTERNALLY;
```

At the minimum, grant the user the CREATE SESSION privilege as follows:

```
SQL> GRANT CREATE SESSION TO king;
```

Use the Biometric Manager to enroll the user in the Biometric Authentication Service.

The user king can now be biometrically authenticated to Oracle.

> **More Information:** For information on creating users identified externally, see *Oracle8i Administrator’s Guide* and *Oracle8i Distributed Database Systems.*
>
> For information on the Biometric Authentication Service and on storing the secret key in the client, see the Identix documentation.

Authenticating Users with a Biometric Authentication Service

Before you authenticate a user, ensure that the Biometric Authentication Service has been installed and configured and the steps in "Administering the Biometric Authentication Service" in this chapter have been executed.

Perform the following steps to authenticate users with a Biometric Authentication Service:

1. Log on as the user assigned by the database administrator.
2. If you are using TouchSafe II, set the system environment variable. The following variable is based on the 10 port setting on the TouchSafe II firmware:

   ```
   ETSII_IOPORT = 0X280
   ```

   > **Note:** The TouchSafe III device does not use the ETSII_IOPORT environment variable. Instead, it uses the `tn3com.ini` file to set the port and baud rate.

3. Enter the following to launch SQL*Plus:

   ```
   sqlplus
   ```

4. Enter the name of the database server at the SQL*Plus prompt:

   ```
   SQL>connect /@net_service_name
   ```

   where *net_service_name* is Net8 net service name.
5. The Net8 Native Authentication dialog box appears followed by a beep sound.

   > **Note:** On some systems, this dialog box is displayed behind the current window. The beep alerts you when it is displayed.

6. Click OK in the Net8 Native Authentication dialog box.
7. When a message appears telling you to place your finger on the desktop fingerprint sensor, use the same finger that you and the administrator entered into the authentication server repository.
8. Remove your finger at the prompt. Another prompt tells you whether you have been authenticated.

9. If authentication fails and the message, "Access Denied," appears try one of the following recovery methods:

- Restart the authentication process.

  More Information: See "Authenticating Users with a Biometric Authentication Service" in this chapter.

- Have the security administrator lower the threshold value to 80.
- Have the security administrator re-enroll you.

  More Information: See the Biometric Manager online help for task oriented procedures.

Troubleshooting

Check the following if you encounter any problems installing or using Identix biometric authentication.

- Ensure that the Identix Set Key utility hash key exactly matches the Biometric Manager DEFAULT Policy hash key.
- The NT user name must exactly match the externally defined user name in the database server and the user name used when adding the user with the Biometric Manager.
- Domain naming must be consistent. For example, if the local naming configuration file (tnsnames.ora) uses .world as an appendix to the service name, then the profile (sqlnet.ora) must reflect this naming convention for the service name. For example:

```
TNSNAMES.ORA
biometrics.world = (DESCRIPTION =
                     (ADDRESS_LIST =
                      (ADDRESS =
                        ...
SQLNET.ORA
sqlnet.identix_fingerprint_database=biometrics.world
```

- It is possible to use one database for both the biometric authentication service and the production database; however, this is not recommended. If you do this, add the following line of code to the local naming configuration file (`tnsnames.ora`) on the server and on each PC client.

```
(connect_data =
    (service_name = service_name)
    (security = (Authentication_service = NONE))
```

9

Configuring DCE GSSAPI Authentication

DCE Generic Security Services Application Programming Interface (GSSAPI) authentication enables you to use DCE authentication even if you do not use other components of the Oracle DCE Integration product in your environment.

This chapter describes how to configure and use DCE GSSAPI authentication.

Note: Check the platform-specific installation documentation to ensure that Oracle Advanced Security supports Oracle DCE integration for your platform.

Note: If you are already using Oracle DCE Integration, you do not also have to use the DCE GSSAPI authentication adapter. The Oracle DCE Integration product, described in Part II, "Oracle DCE Integration", includes DCE authentication.

Note: The instructions in this chapter assume that you are familiar with DCE terminology. For more information about DCE, see:

- Part II, "Oracle DCE Integration", in this guide
- the operating system-specific DCE administration guide
- the documentation listed in "Related Publications"in the Preface

Configuring DCE GSSAPI Authentication

To configure DCE GSSAPI authentication you follow these four general steps, each of which is explained below:

Task 1: Create the DCE Principal

Task 2: Configure the New DCE Principal and Enable DCE GSSAPI Authentication

Task 3: Set up the Account for Authenticating to the Database

Task 4: Connect to an Oracle Server using DCE GSSAPI Authentication

Task 1: Create the DCE Principal

To create the DCE Principal used by the Oracle server to validate authentication, enter the commands below shown in **bold** typeface. These instructions assume the Oracle server principal is named oracle_server.

Enter the following commands on the database server.

```
% su
  password:    (root password is not echoed)
# dce_login cell_admin cell_admin_password
# rgy_edit
Current site is: registry server at
    /.../cellname/subsys/dce/sec/master
rgy_edit=> do p
Domain changed to: principal
rgy_edit=> add oracle_server
rgy_edit=> do a
Domain changed to: account
rgy_edit=> add oracle_server -g none -o none -pw oracle_server_
password -mp cell_admin_password
rgy_edit=> ktadd -p oracle_server -pw oracle_server_password
rgy_edit=> quit
bye
```

Task 2: Configure the New DCE Principal and Enable DCE GSSAPI Authentication

The following instructions assume that the Oracle server principal is named oracle_server. This must be a fully qualified name, including the cell name.

Add the following lines to the `sqlnet.ora` file.

```
SQLNET.AUTHENTICATION_GSSAPI_SERVICE=/.../cellname/oracle_server
SQLNET.AUTHENTICATION_SERVICES=(DCEGSSAPI)
```

Task 3: Set up the Account for Authenticating to the Database

Create the DCE principal used by the Oracle client to connect to the database. The following instructions assume the Oracle client principal is named oracle.

Enter the following:

```
% dce_login cell_admin cell_admin_password
% rgy_edit
Current site is : registry server at /.../cellname/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_client_password -mp
cell_admin_password
rgy_edit=> quit
bye
```

Create the Oracle database user account. The following instructions show how to use SQL*Plus to do this.

Enter the following:

```
sqlplus
SQL> connect internal
Connected
SQL> create user "/.../CELLNAME/ORACLE" identified externally;
Statement processed.
SQL> grant connect to "/.../CELLNAME/ORACLE";
Statement processed.
SQL> exit
```

Note: The Oracle client principal name must be a fully qualified principal (including full cell designation), must be in uppercase, and must be enclosed within quotes.

Task 4: Connect to an Oracle Server using DCE GSSAPI Authentication

The following instructions assume the Oracle server principal is oracle_server, the Oracle client principal is oracle, and the database service name is sales.

1. If your DCE authentication is not already encapsulated into the operating system authentication, log in as follows:

   ```
   % dce_login oracle_client_principal oracle_client_password
   ```

   For example:

   ```
   % dce_login oracle oraclnt
   ```

2. Connect to the Oracle database using DCE GSSAPI authentication as follows:

   ```
   % sqlplus /@<database_service_name>
   ```

   For example:

   ```
   %sqlplus /@sales
   ```

10

Configuring Secure Socket Layer Authentication

This chapter provides information on how to use the Secure Socket Layer (SSL) protocol. It contains the following topics:

SSL In an Oracle Environment

Secure Sockets Layer (SSL) is an industry standard protocol designed by Netscape Communications Corporation for securing network connections. SSL provides 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 for encryption and integrity and provides public key authentication based on the SSL standard.

You can use the Oracle Advanced Security SSL feature to secure communications between clients and servers. Specifically, you can use SSL to authenticate:

- Any client or server to one or more Oracle servers
- An Oracle 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.

> **More Information:** For more information on authentication methods, see Chapter 1, "Introduction to Oracle Advanced Security."

You can use SSL in one 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

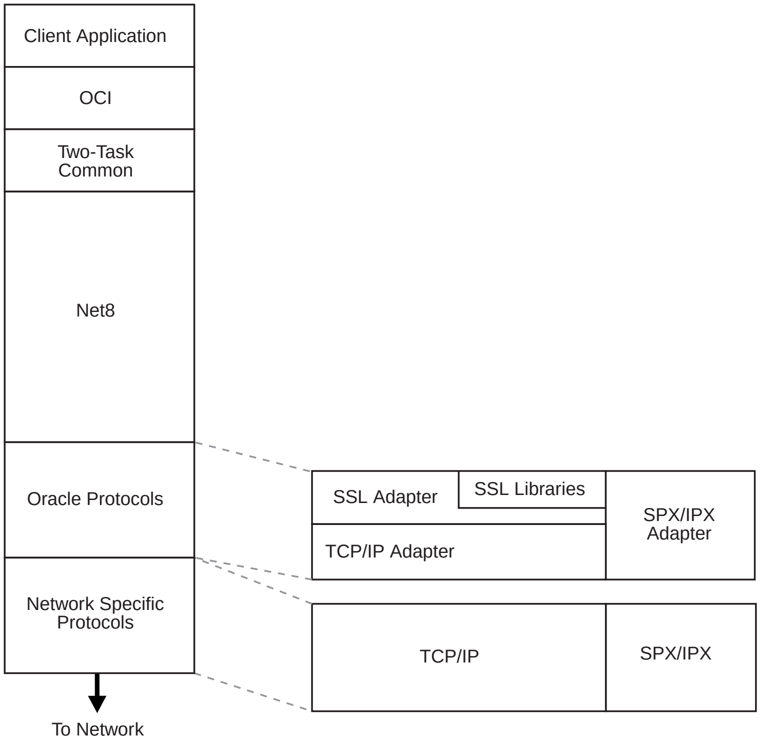
More Information: For a full explanation of SSL, see the Internet Engineering Task Force document, *The SSL Protocol*, Version 3.0.

For important security concepts and terminology, see the Glossary.

Architecture of SSL in an Oracle Environment

In an Oracle environment, SSL operates at the Oracle Protocols layer using TCP/IP as illustrated in Figure 10–1.

Figure 10–1 SSL Architecture in an Oracle Environment



Components of SSL in an Oracle Environment

The components of SSL in an Oracle environment include the following, each of which is described below:

- Certificate
- Certificate Authority (CA)
- Wallet

Certificate

A certificate ensures that the entity's identity information is correct and that the public key actually belongs to that entity. A certificate is created when an entity's public key is signed by a trusted identity, that is, a certificate authority (CA), described more fully in this section.

A certificate contains the entity's name, public key, a serial number, and an expiration date. It can contain information about the privileges associated with the certificate. Finally, it contains certificate chain information.

When an entity receives a certificate, either its own certificate from a CA or a certificate from another entity, it verifies that certificate is a **trusted certificate**, that is, that it is issued by a trusted certificate authority. A certificate is valid until it expires.

Certificate Authority (CA)

A certificate authority is a trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are. The certificate authority verifies the entity's identity and grants a certificate, signing it with the certificate authority's private key.

Different CAs may have different identification requirements when issuing certificates. One certificate authority may want to see a user's driver's license, another may want the certificate request form to be notarized, yet another may want fingerprints of the person requesting a certificate.

The certificate authority publishes its own certificate which includes its public key. Each network entity has a list of such 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 from different CAs.

Note: Oracle Advanced Security automatically installs trusted certificates from VeriSign, RSA, and GTE CyberTrust by default when you create a new wallet.

For information on adding certificates, see "Creating a New Wallet" in Chapter 18, "Using Oracle Wallet Manager."

For information on adding trusted certificates, see "Managing Trusted Certificates" in Chapter 18, "Using Oracle Wallet Manager."

Wallet

A wallet is an abstraction used to store and manage authentication data such as keys, certificates, and trusted certificates that are 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 8.1.6, also installs Oracle Wallet Manager release 2.0 and Oracle Enterprise Login Assistant release 1.0.

More Information: For information on the Oracle Wallet Manager, see Chapter 18, "Using Oracle Wallet Manager."

How SSL Works in an Oracle Environment: The SSL Handshake

At the beginning of their communication, the client and server perform an SSL handshake that includes the following important tasks:

- The client and server establish which **cipher suites** to use.
- The server sends its certificate to the client. 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. The server verifies that the client's certificate was signed by a trusted CA.
- The client and server exchange key material using public key cryptography, and, from this material, they each generate a session key. All subsequent communication between client and server is encrypted and decrypted by using this set of session keys and the negotiated cipher suite.

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

1. The user initiates a Net8 connection to the server by using SSL.
2. SSL performs the handshake between client and server.
3. If the handshake is successful, the server verifies that the user has the appropriate **authorization** to access the database.

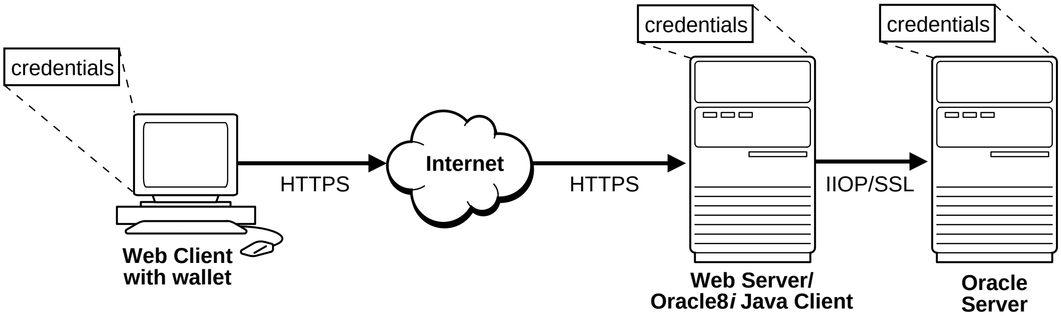
> **More Information:** For a full explanation of SSL, see the Internet Engineering Task Force document *The SSL Protocol*, Version 3.0.

SSL Beyond an Oracle Environment

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

Figure 10–2 shows an example of using SSL to secure connections between Oracle and non-Oracle entities, beginning over the Internet and proceeding to an Oracle server. In this example, a Web server runs as an Oracle8*i* Java client. It receives messages over **HTTPS** (HTTP secured by SSL), and sends **CORBA** requests to the Oracle server via a servlet over **IIOP**/SSL (IIOP secured by SSL). Note that, in this example, the Web server passes its own certificate, and not the Web client's, to the Oracle server.

Figure 10–2 Connecting to an Oracle Server over the Internet



> **More Information:** For information on using and configuring IIOP / SSL, see *Oracle8i Enterprise JavaBeans and CORBA Developer's Guide*.

SSL Combined with Other Authentication Methods

You can combine the features of SSL with other authentication methods supported by Oracle Advanced Security, such as Kerberos, SecurID, RADIUS, or Identix.

> **More Information:** For information on authentication methods, see Chapter 1, "Introduction to Oracle Advanced Security."

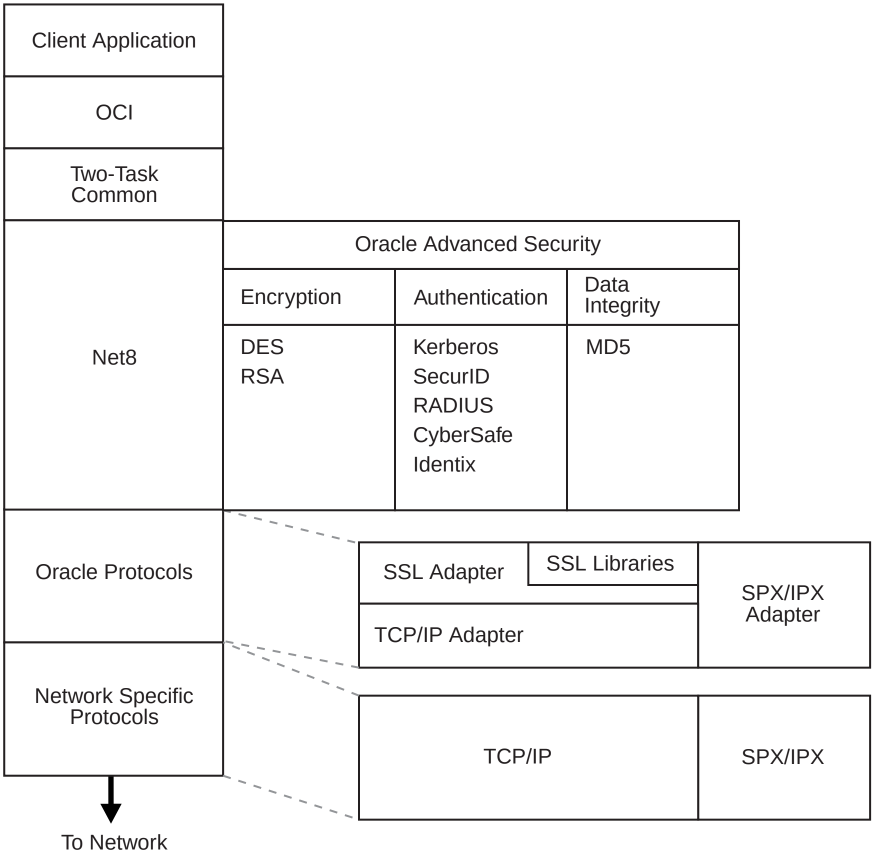
This section discusses the following topics:

- Architecture of SSL Combined with Other Authentication Methods
- Using SSL Combined with Other Oracle Authentication Methods

Architecture of SSL Combined with Other Authentication Methods

As Figure 10–3 illustrates, Oracle Advanced Security operates at the session layer on top of SSL which uses TCP/IP at the transport layer.

Figure 10–3 SSL in Relation to Oracle Advanced Security

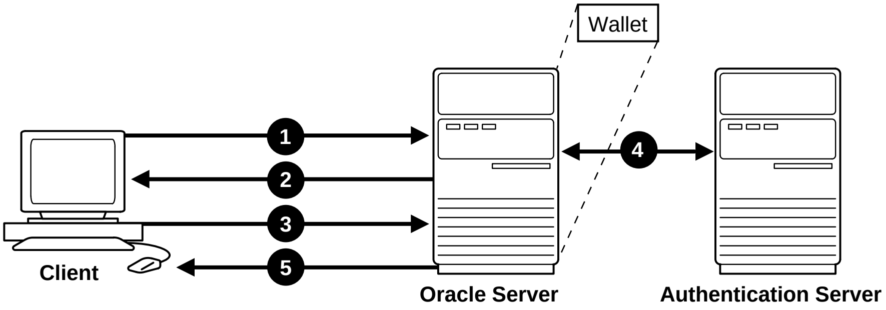


More Information: For more information on stack communications in an Oracle networking environment, see *Net8 Administrator's Guide.*

Using SSL Combined with Other Oracle Authentication Methods

Figure 10–4 shows a scenario in which SSL is used in combination with another authentication method supported by Oracle Advanced Security. In this scenario, server authentication uses SSL, and client authentication uses an authentication method supported by Oracle Advanced Security, such as Kerberos, SecurID, Identix, and Radius.

Figure 10–4 SSL in Relation to Other Authentication Methods



1. The client seeks to connect to the Oracle 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" in this chapter.
3. Once the SSL handshake is successfully completed, the user seeks access to the database.
4. The Oracle server exchanges the user's authentication information with the authentication server.
5. Upon validation by the authentication server, the Oracle server grants access and authorization to the user.

The user, at the client, can now access the Oracle server securely using SSL.

Warning: You can use SSL encryption in combination with another Oracle Advanced Security authentication method. When you do this, you must disable any non-SSL encryption to comply with government regulations prohibiting double encryption. If you do not do this, the connection fails.

You cannot use SSL authentication with Oracle Advanced Security encryption.

For information on how to disable Oracle Advanced Security encryption, see Chapter 2, "Configuring Data Encryption and Integrity."

Issues When Using SSL

Consider the following issues when using SSL:

- SSL cannot be proxied through traditional application level firewalls, such as the CERN proxy server.
- SSL use enables authorizations to be retrieved from a LDAP-based directory service. Client-side SSL authentication is required in order to manage enterprise users and their privileges in a directory.
- Because SSL does authentication and encryption, from a performance standpoint, it is slower than the standard Net8 TCP/IP transport.
- The Oracle Advanced Security SSL feature does not work with versions of Oracle earlier than Oracle8*i*.
- Each SSL authentication mode requires unique configuration settings, as described in the following section.

Enabling SSL

To enable SSL, perform the general tasks in the following list:

- 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 install Oracle Advanced Security, the Oracle Universal Installer installs SSL, Oracle Wallet Manager, and Oracle Enterprise Login Assistant on your system.

> **More Information:** See the Oracle8*i* installation documentation for your platform.

Task 2: Configure SSL on the Client

To configure SSL on the client, perform the tasks in the following list:

- Specify Required Client Configuration (Wallet Location)
- Set the SSL Cipher Suites on the Client (Optional)
- Set the Required SSL Version (Optional)
- Set SSL as an Authentication Service (Optional)
- Create a Net Service Name that Uses TCP/IP with SSL in the Connect Descriptor

> **Note:** There are two ways to configure a parameter:
>
> - Statically, by setting the parameters in the `sqlnet.ora` file, as described in "Specify Required Client Configuration (Wallet Location)".
> - Dynamically, by setting the parameters in the security subsection of the Net8 address.

> **More Information:** For the dynamic parameter names, see Appendix B, "Authentication Parameters".

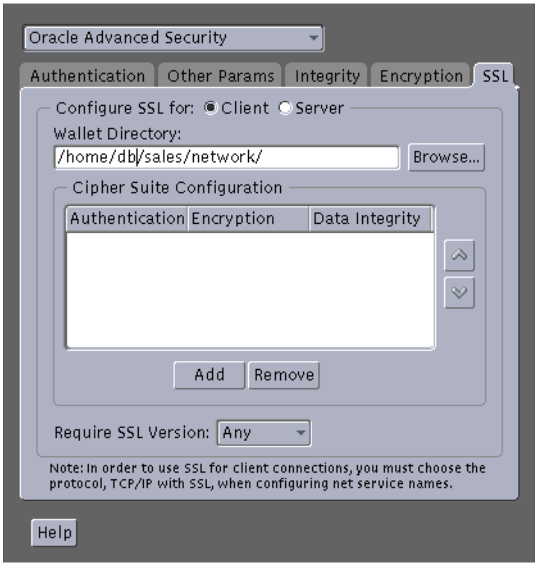
Specify Required Client Configuration (Wallet Location)

To specify required configuration parameters for the client:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.

2. In the navigator’s pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the SSL tab.



Oracle Advanced Security

Configure SSL for: ● Client ○ Server

Wallet Directory: /home/db/sales/network/

Cipher Suite Configuration

Authentication	Encryption	Data Integrity

Require SSL Version: Any

Note: In order to use SSL for client connections, you must choose the protocol, TCP/IP with SSL, when configuring net service names.

5. Select Configure SSL for Client.
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.

Note: You must later specify this same directory when you create a new wallet.

---

7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entries:

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

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.

The SSL_CIPHER_SUITES parameter sets the cipher suites SSL uses.

When you install Oracle Advanced Security, several SSL cipher suites are set for you by default. Setting one or more cipher suites yourself overrides the other default cipher suites set during installation. For example, if you use Net8 Assistant 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.

The following two tables list the available SSL cipher suites supported in both the domestic and export editions of Oracle Advanced Security. These cipher suites are set by default when you install Oracle Advanced Security. These tables also list the authentication, encryption, and data integrity types each cipher suite uses.

Table 10–1 Oracle Advanced Security Domestic Edition 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
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

Table 10–2 Oracle Advanced Security Export Edition Cipher Suites

Cipher Suite	Authentication	Encryption	Data Integrity
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4 40	MD5
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

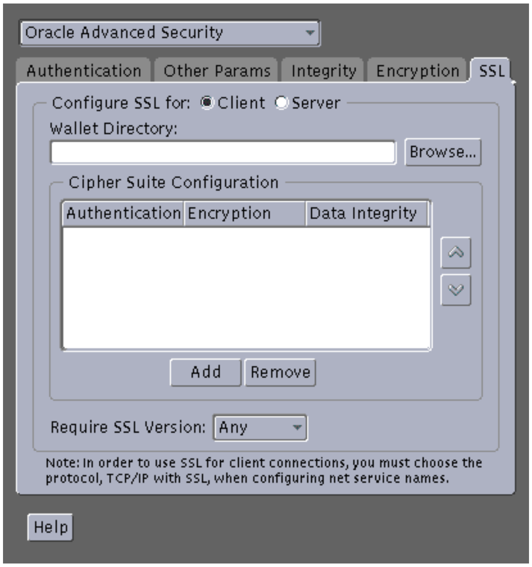
Warning: If you use SSL in conjunction with another authentication method supported by Oracle Advanced Security, you must disable any non-SSL encryption to comply with government regulations prohibiting double encryption. If you do not do this, the connection fails.

For information on how to disable Oracle Advanced Security encryption, see "Negotiating Encryption and Integrity" in Chapter 2, "Configuring Data Encryption and Integrity."

To specify cipher suites for the client:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the SSL tab.



Oracle Advanced Security

Configure SSL for: ● Client ○ Server

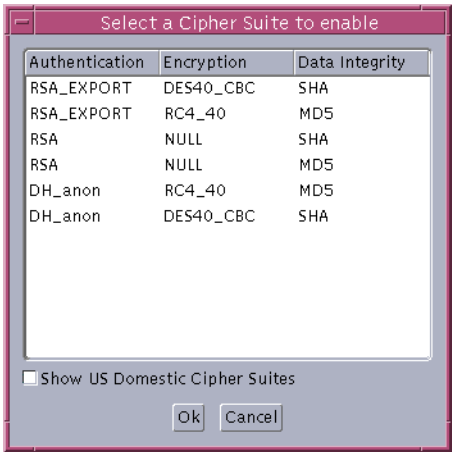
Wallet Directory:

Require SSL Version: Any

Note: In order to use SSL for client connections, you must choose the protocol, TCP/IP with SSL, when configuring net service names.

5. Select Configure SSL for Client.

6. Click the Add button. A secondary dialog box listing available cipher suites appears.

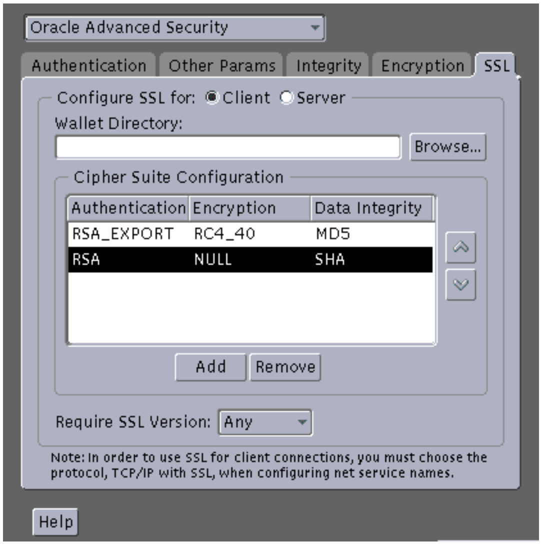


Select a Cipher Suite to enable

Authentication	Encryption	Data Integrity
RSA_EXPORT	DES40_CBC	SHA
RSA_EXPORT	RC4_40	MD5
RSA	NULL	SHA
RSA	NULL	MD5
DH_anon	RC4_40	MD5
DH_anon	DES40_CBC	SHA

□ Show US Domestic Cipher Suites

7. Add a suite to the Cipher Suite Configuration list by selecting a suite and clicking OK. The Cipher Suite Configuration list updates with the cipher suite.



Oracle Advanced Security

Configure SSL for: ● Client ○ Server

Wallet Directory:

Cipher Suite Configuration

Authentication	Encryption	Data Integrity
RSA_EXPORT	RC4_40	MD5
RSA	NULL	SHA

Require SSL Version: Any

Note: In order to use SSL for client connections, you must choose the protocol, TCP/IP with SSL, when configuring net service names.

8. Use the up and down arrow buttons to prioritize the cipher suites. The order you set determines the order in which the client negotiates with other entities

on which cipher suite to use.

9. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry:

   ```
   SSL_CIPHER_SUITES= (SSL cipher suite1 [,SSL cipher suite2])
   ```

Set the Required SSL Version (Optional)

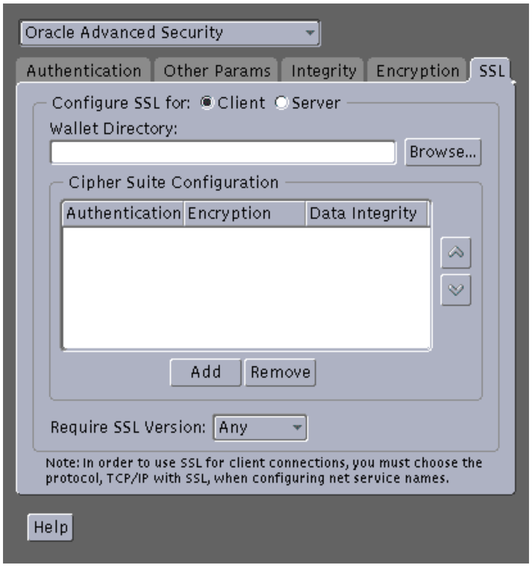
You can set the SSL_VERSION parameter in the `sqlnet.ora` file. The parameter defines the version of SSL that must run on the machines with which the client communicates. You can require those machines to use SSL 3.0, or any existing or future versions. The default setting for this parameter in `sqlnet.ora` is 0; in Net8 Assistant it is Any.

To set the SSL version for the client:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the SSL tab.



Configure SSL for: ● Client ○ Server

Require SSL Version: Any

Note: In order to use SSL for client connections, you must choose the protocol, TCP/IP with SSL, when configuring net service names.

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 enforce.

7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry:

   ```
   SSL_VERSION={ 0 | 3.0 }
   ```

Set SSL as an Authentication Service (Optional)

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

You must set this parameter only if both of the following two 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 or SecurID.

  and

- You are not using Net8 Assistant 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, identix,securid)
```

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

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.

> **More Information:** See the *Net8 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 server and on the Oracle client for all SSL parameters except the location of the Oracle wallet. To configure SSL on the server, perform the following tasks.

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

> **Note:** There are two ways to configure a parameter:
>
> - Statically, by setting the parameters in the `sqlnet.ora` file, as described in "Specify Required Client Configuration (Wallet Location)".
> - Dynamically, by setting the parameters in the security subsection of the Net8 address.

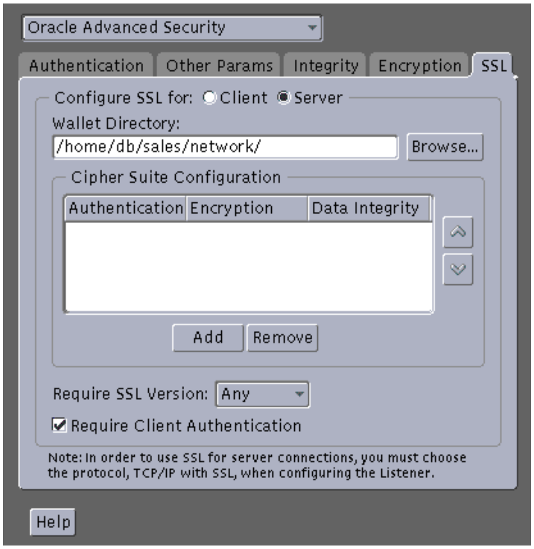
More Information: For the dynamic parameter names, see Appendix B, "Authentication Parameters".

Specify Server Configuration (Wallet Location)

To specify required configuration parameters for the server:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the SSL tab.



Oracle Advanced Security

Configure SSL for: ○ Client ● Server

Wallet Directory: /home/db/sales/network/

Cipher Suite Configuration

Authentication	Encryption	Data Integrity

Require SSL Version: Any

☑ Require Client Authentication

Note: In order to use SSL for server connections, you must choose the protocol, TCP/IP with SSL, when configuring the Listener.

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.

Note: You must later specify this same directory when you create a new wallet.

7. Choose File > Save Network Configuration.

   The `sqlnet.ora` and `listener.ora` files update with the following entry. allowing for secure connections over SSL:

   ```
   OSS.SOURCE.MY_WALLET =
    (SOURCE=
     (METHOD=File)
     (METHOD_DATA=
      (DIRECTORY=<wallet_location>)))
   ```

8. Add the following line to the listener.ora file to have the database authenticate the client rather than the listener:

   ```
   SSL_CLIENT_AUTHENTICATION=FALSE
   ```

   SSL_CLIENT_AUTHENTICATION is set to TRUE by default. Since the database authenticates the client, it is not necessary to have the listener authenticate the client. Therefore, Oracle Corporation recommends setting this parameter to FALSE. Decisions to set this parameter to TRUE are at the discretion of your security administrator.

> **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 Net8 Assistant or by modifying the sqlnet.ora file.
> - Later, you use the Oracle Wallet Manager as described in "Creating a New Wallet" in Chapter 17.

Set the SSL Cipher Suites on the Server (Optional)

The SSL_CIPHER_SUITES parameter sets the cipher suites SSL uses.

When you install Oracle Advanced Security, several SSL cipher suites are set for you by default. Setting one or more cipher suites yourself overrides the other default cipher suites set during installation. For example, if you use Net8 Assistant to add the cipher suite SSL_RSA_WITH_RC4_128_SHA, all other cipher suites in the default setting are removed.

You can prioritize the cipher suites. When the server negotiates with clients over 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 effect. For example, triple-DES encryption is stronger than DES. See Chapter 2, "Configuring Data Encryption and Integrity."
- 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.

For example, in the case of an Oracle Call Interface (OCI) user, the server requires the client to authenticate itself. In this case, you cannot 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.

Note: In Oracle Advanced Security version 8.1.6, 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 "Set SSL Client Authentication (Optional)"in this chapter.

You would typically prioritize cipher suites starting with the strongest and moving to the weakest. The following two tables list the available SSL cipher suites supported in both the domestic and export editions of Oracle Advanced Security. These tables also list the authentication, encryption, and data integrity types each cipher suite uses.

Table 10–3 SSL Cipher Suites in Domestic Edition of Oracle Advanced Security

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 10–3 SSL Cipher Suites in Domestic Edition of Oracle Advanced Security

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

Table 10–4 SSL Cipher Suites in Export Edition of Oracle Advanced Security

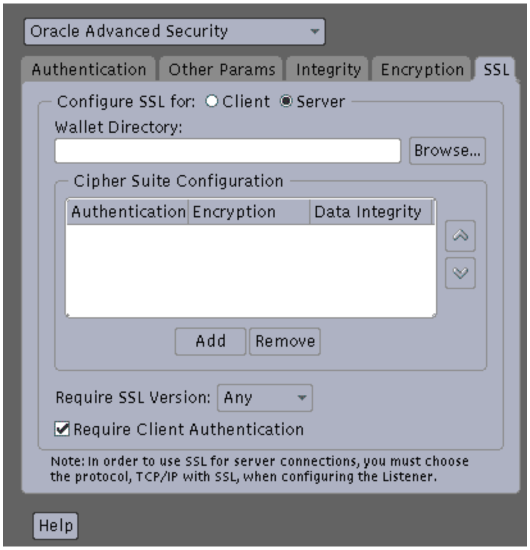
Cipher Suite	Authentication	Encryption	Data Integrity
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4 40	MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA	DES40 CBC	SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	DH anon	RC4 40	MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH anon	DES40 CBC	SHA

To specify cipher suites for the server:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator’s pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the SSL tab.



Configure SSL for: ○ Client ● Server

Wallet Directory:

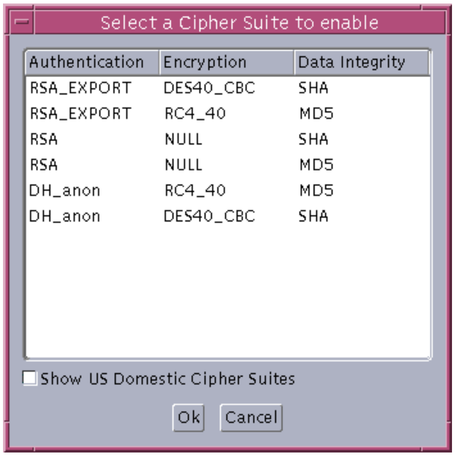
Require SSL Version: Any

☑ Require Client Authentication

Note: In order to use SSL for server connections, you must choose the protocol, TCP/IP with SSL, when configuring the Listener.

5. Select Configure SSL for Server.

6. Click the Add button. A secondary dialog box listing available cipher suites appears.

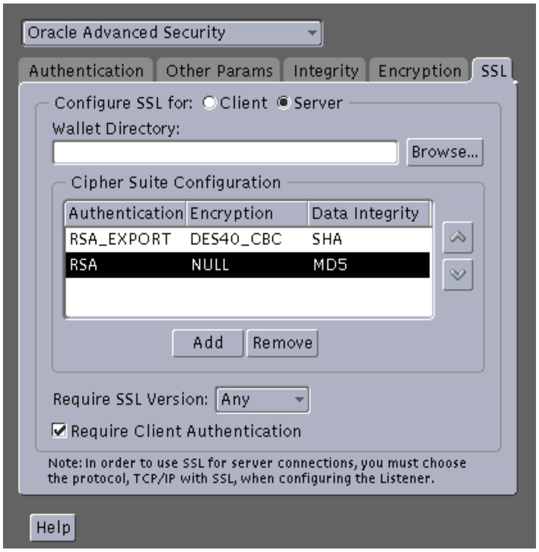


Select a Cipher Suite to enable

Authentication	Encryption	Data Integrity
RSA_EXPORT	DES40_CBC	SHA
RSA_EXPORT	RC4_40	MD5
RSA	NULL	SHA
RSA	NULL	MD5
DH_anon	RC4_40	MD5
DH_anon	DES40_CBC	SHA

□ Show US Domestic Cipher Suites

7. Add a suite to the Cipher Suite Configuration list by selecting a suite and clicking OK. The Cipher Suite Configuration list updates with the cipher suite.



Oracle Advanced Security

Configure SSL for: ○ Client ● Server

Wallet Directory:

Cipher Suite Configuration

Authentication	Encryption	Data Integrity
RSA_EXPORT	DES40_CBC	SHA
RSA	NULL	MD5

Require SSL Version: Any

☑ Require Client Authentication

Note: In order to use SSL for server connections, you must choose the protocol, TCP/IP with SSL, when configuring the Listener.

8. Use the up and down arrow buttons to prioritize the cipher suites. The order you set determines the order in which the client negotiates with other entities

on which cipher suite to use.

9. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry:

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

Set the Required SSL Version (Optional)

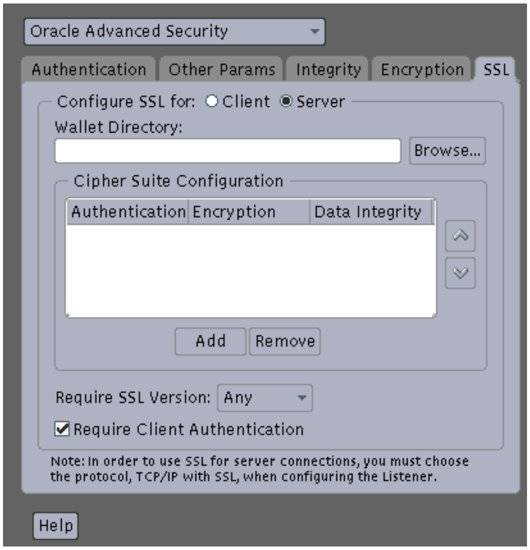
You can set the SSL_VERSION parameter in the `sqlnet.ora` file. This parameter defines the version of SSL that must run on the machines with which the client communicates. You can require those machines to use SSL 3.0, or any existing or future versions. The default setting for this parameter in `sqlnet.ora` is 0; in Net8 Assistant it is Any.

To set the SSL version on the server:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the SSL tab.



Configure SSL for: ○ Client ● Server

Wallet Directory:

Require SSL Version: Any

☑ Require Client Authentication

Note: In order to use SSL for server connections, you must choose the protocol, TCP/IP with SSL, when configuring the Listener.

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 enforce.
7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry, listing the version in order of priority used:

   ```
   SSL_VERSION={ 0 | 3.0 }
   ```

Set SSL Client Authentication (Optional)

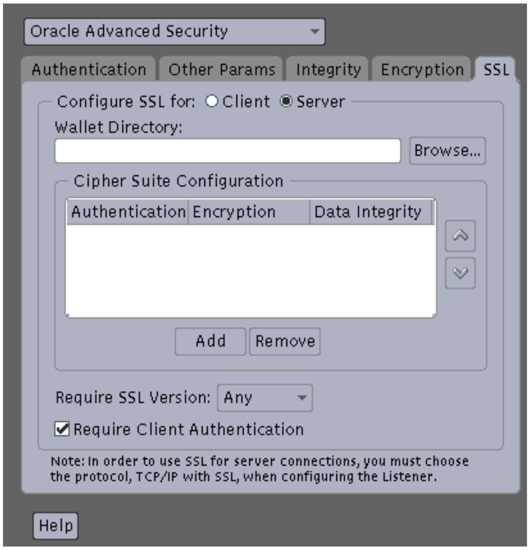
The SSL_CLIENT_AUTHENTICATION parameter in the `sqlnet.ora` file controls whether the client is authenticated using SSL. The default value is TRUE.

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 Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.
4. Click the SSL tab.



Configure SSL for: ○ Client ● Server

Wallet Directory:

Require SSL Version: Any

☑ Require Client Authentication

Note: In order to use SSL for server connections, you must choose the protocol, TCP/IP with SSL, when configuring the Listener.

5. Select Configure SSL for Server.
6. Deselect Require Client Authentication.
7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry:

   ```
   SSL_CLIENT_AUTHENTICATION=FALSE
   ```

Set SSL as an Authentication Service (Optional)

The SQLNET.AUTHENTICATION_SERVICES parameter sets the SSL authentication service.

You must 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 or SecurID.
- You are not using Net8 Assistant to make any configuration changes.

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

```
SQLNET.AUTHENTICATION_SERVICES = (BEQ, TCPS, selected_method_1, selected_method_2)
```

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

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 Net8 clients and 2482 for client connections to Oracle8*i* JServer.

> **More Information:** See the *Net8 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
```

11

Choosing and Combining Authentication Methods

This chapter describes how to use conventional user name and password authentication even if you have configured another authentication method. It also describes how to configure the network to use one or more authentication methods using Oracle Advanced Security and how to set up more than one authentication method on a client or on a server.

This chapter covers the following topics:

- Connecting with User Name and Password
- Disabling Oracle Advanced Security Authentication
- Configuring Oracle For Multiple Authentication Methods

Connecting with User Name and Password

To connect to an Oracle server using a user name and password when an Oracle Advanced Security authentication method has been configured, disable the external authentication.

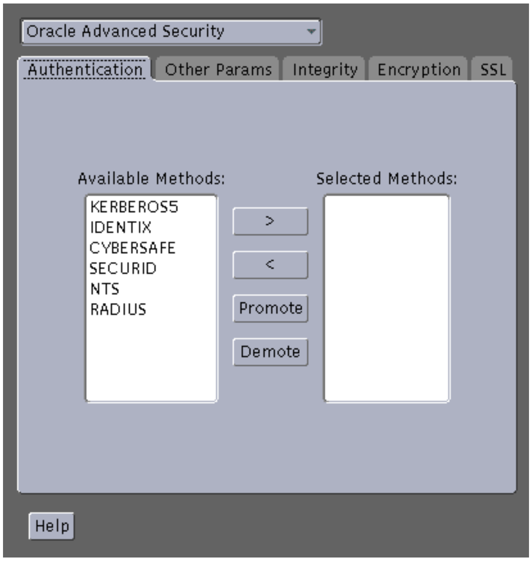
Disabling Oracle Advanced Security Authentication

Perform the following steps to disable authentication methods:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Authentication tab.



5. Move an authentication method from the Selected Method list to the Available Methods list by selecting a method and clicking the left arrow [<].
6. Repeat until all methods are removed from the Selected Methods area.
7. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry:

   ```
   SQLNET.AUTHENTICATION_SERVICES = (NONE)
   ```

A user can now connect to a database using the following user name and password format:

```
% sqlplus username/password@net_service_name
```

For example:

```
% sqlplus scott/tiger@emp
```

Configuring Oracle For Multiple Authentication Methods

Many networks use more than one authentication method on a single security server. For this reason, Oracle Advanced Security allows you to configure your network so that Oracle clients can use a specific authentication method and Oracle servers can accept any method specified.

This section describes how to set up Oracle servers and clients to use multiple authentication methods.

Set up multiple authentication methods on both client and server machines either by using the Net8 Assistant, or by using any text editor to modify the `sqlnet.ora` file.

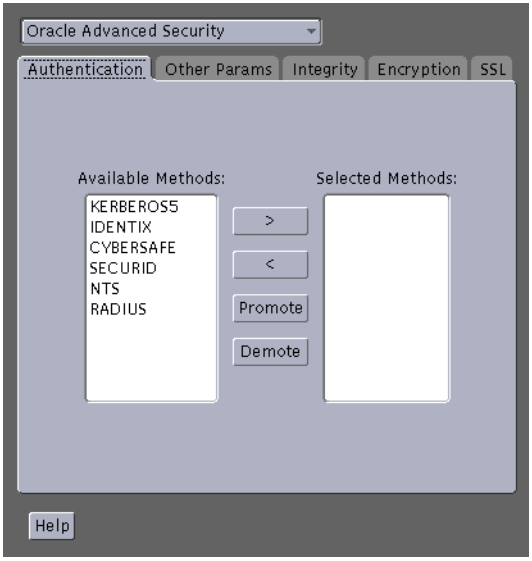
The following instructions apply to both clients and servers.

To disable authentication methods:

1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the navigator's pane, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed pages appear.

4. Click the Authentication tab.



5. Select a method listed in the Available Methods list.
6. Move the method to the Selected Methods list by clicking the right arrow.
7. Repeat until you have added all of the required methods to the Selected Methods list.
8. Arrange the selected methods in order of desired use. To do this, select a method in the Selected Methods list, then click Promote or Demote to position it in the list.
9. Choose File > Save Network Configuration.

   The `sqlnet.ora` file updates with the following entry, listing the selected authentication methods:

   ```
   SQLNET.AUTHENTICATION_SERVICES =
   (RADIUS|CYBERSAFE|KERBEROS5|SECURID|IDENTIX)
   ```

Part II

Oracle DCE Integration

The following chapters describe Oracle Distributed Computing Environment (DCE) Integration.

- Chapter 12, "Overview of Oracle DCE Integration"
- Chapter 13, "Configuring DCE for Oracle DCE Integration"
- Chapter 14, "Configuring Oracle for Oracle DCE Integration"
- Chapter 15, "Connecting to an Oracle Database in DCE"
- Chapter 16, "DCE and Non-DCE Interoperability"

Note: Check the platform-specific installation documentation to verfify that Oracle Advanced Security supports Oracle DCE integration on your platform.

12

Overview of Oracle DCE Integration

This chapter briefly describes the Distributed Computing Environment (DCE) and the Oracle DCE Integration product.

This chapter contains the following topics:

- System Requirements
- Backward Compatibility
- Overview of Distributed Computing Environment (DCE)
- Overview of Oracle DCE Integration

> **More Information:** See the list of related books and papers in "Related Publications" in the Preface.

System Requirements

Oracle DCE Integration requires Net8 and Oracle8*i*. It enables Oracle applications and tools to access Oracle8*i* servers in a distributed computing environment.

Oracle DCE Integration is based on the Open Software Foundation (OSF) DCE V1.1 and above, and will be compatible with OSF's future DCE releases.

OSF has merged with another standards group, X/OPEN, to form The Open Group. This group will continue to support DCE.

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, the 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 2.3.2 or later server in order to connect to a database.

Overview of Distributed Computing Environment (DCE)

The Distributed Computing Environment (DCE) from the Open Software Foundation (OSF) is a set of integrated network services that work 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.

The following topics are described in this section:

- Components of Oracle DCE Integration
- Flexible DCE Deployment
- Release Limitations

More Information: See "Related Publications" in the Preface.

Overview of Oracle DCE Integration

Oracle DCE Integration enables use of Oracle tools and applications to access Oracle8*i* servers in a DCE environment.

Components of Oracle DCE Integration

Oracle's DCE Integration product consists of DCE Communication / Security and DCE CDS Native Naming.

DCE Communication/Security

DCE Communication / Security includes the following:

- **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 referred to as external authentication to the database. It is also known as 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. For example, users have a range of choices from no protection to full encryption for each connection, with a guarantee that no data has been 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 Net8. They provide message integrity and data encryption services in non-DCE environments, allowing administrators to ensure that all network traffic is protected against unauthorized viewing or modification, regardless of the start or end point.

DCE CDS Native Naming

The DCE CDS Native Naming component includes naming and location transparency.

DCE Integration registers Oracle8*i* connect descriptors in the DCE Cell Directory Service (CDS), allowing them to 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 Oracle8*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)

More Information: For more information about DCE CDS Native Naming, see the following:

- To configure DCE to use CDS naming, see Chapter 13, "Configuring DCE for Oracle DCE Integration".
- To configure Oracle clients and servers to use CDS, see Chapter 14, "Configuring Oracle for Oracle DCE Integration".
- For information on how Oracle Native Naming works with other Oracle name services, see the *Net8 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) described in this part of the guide.
- 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. Configuration of the CDS Native Naming adapter is described in Chapter 13, "Configuring DCE for Oracle DCE Integration" and Chapter 14, "Configuring Oracle for Oracle DCE Integration" in this guide.
- You can use only DCE authentication services by using the DCE GSSAPI authentication method described in Chapter 9, "Configuring DCE GSSAPI Authentication" of this guide. This option requires OSF DCE 1.1.

Release Limitations

The following are limitations in the current release (8.1.6) of Oracle Advanced Security:

- Only one listener address that uses the DCE protocol is allowed per 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 Multi-Threaded Server (MTS).

13

Configuring DCE for Oracle DCE Integration

This chapter describes what you need to do to configure DCE to use Oracle DCE Integration after Oracle DCE Integration has been successfully installed.

This chapter contains the following topics:

- 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

More Information: See the list of related books and papers in "Related Publications" in the Preface.

The following is a list of tasks with examples you need to follow to configure DCE to use DCE Integration. The tasks assume that a DCE cell has been configured and the machines being used are part of that cell.

As the DCE cell administrator, you need to perform the following tasks:

- 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

First, add server principals using a procedure like the one below:

```
% 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, you just created a DCE principal named oracle. The principal has a corresponding account with password *password*. The account does not belong to any DCE group or DCE profile.

You only need to do this once after DCE Integration has been installed. Also, you only need to do this procedure for the Oracle database server, not for the client.

Task 2: Install the Key of the Server into a Keytab File

In this step by step procedure, you install the key of the server into a keytab file: `dcepa.key`. This keytab file contains the password of the principal under which the Net8 listener starts. The Net8 listener reads this file to authenticate itself to DCE. You only need to do this once after DCE Integration has been installed. Also, you only need to do this procedure for the Oracle database server, not for the client.

Note: Remember to substitute the correct full pathname for the $ORACLE_HOME variable. If the specified directories do not already exist, you must create it before running the command. Enter the following to create the directories.

```
mkdir $ORACLE_HOME/dcepa
mkdir $ORACLE_HOME/dcepa/admin
```

Enter the following commands to generate the keytab file.

```
% 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
```

Task 3: Configure DCE CDS for Use by Oracle DCE Integration

The /.:/subsys/oracle/names directory contains objects that map Net8 service names to connect descriptors, which are used by the CDS naming adapter.

The /.:/subsys/oracle/service_registry directory also contains objects that map the service name in DCE addresses to the network endpoint which is used by both DCE protocol adapter clients and servers.

Create Oracle Directories in the CDS Namespace

Perform the steps in this section after installing DCE Integration for the first time in a cell.

```
% 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: Create the directories on all CDS replicas.

Give Servers Permission to Create Objects in the CDS Namespace

Perform the following steps 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
```

Load Oracle Service Names into CDS

Load Oracle service names into the Cell Directory Service.

More Information: For instructions on how to configure clients and load Oracle service names into CDS, see Chapter 14, "Configuring Oracle for Oracle DCE Integration".

14

Configuring Oracle for Oracle DCE Integration

This chapter discusses how to configure Oracle and Net8 to use Oracle DCE Integration after it has been successfully installed. The following sections describe the tasks you must perform for servers and clients.

- DCE Address Parameters
- Configuring the Server
- Creating and Naming Externally-Authenticated Accounts
- Setting up DCE Integration External Roles
- Connecting to an Oracle Database as SYSDBA or SYSOPER Using DCE
- Configuring the Client
- Configuring Clients to Use DCE CDS Naming

DCE Address Parameters

DCE addresses in the `listener.ora` and `tnsnames.ora` configuration files are defined by DCE parameters. The parameters consist of both mandatory and optional fields, which are described below:

```
ADDRESS=(PROTOCOL=DCE)(SERVER_PRINCIPAL=server_name)(CELL_NAME=cell_name)
(SERVICE=dce_service_name))
```

DCE address parameters and definitions are listed in Table 14–1.

Table 14–1 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
```

or

SERVICE=*dce_service_name*

provided that `CELL_NAME=cell_name` is also specified.

You can also specify the following:

SERVICE=*dce_service_name*

in which case the cell name defaults to the local cell. However, this way of specifying service names only works well if you are working within a single cell.

Note: The *dce_service_name* in the service field might not be the same as the service name used by Net8. The service name used by Net8 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.

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.

Configuring the Server

To configure a server for DCE Integration, you need to configure the following Net8 files with DCE address and parameter information:

- Listener configuration file (`listener.ora`) must be configured with DCE address information for all servers.
- Profile (`sqlnet.ora`) and `protocol.ora` must be configured for servers in distributed systems that need to make database link connections to other servers.

Note: Use the Net8 Assistant to create the necessary configuration files. For explanations of the configuration files, see the *Net8 Administrator's Guide.*

For a database server to receive connections from Net8 clients in a DCE environment, there must be a Net8 listener active on the server platform. A listener listens for connections on a network address that is defined in the listener configuration file, `listener.ora`.

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/oracle8))
```

Creating and Naming Externally-Authenticated Accounts

To use DCE authentication for logging onto the Oracle database, you must create database accounts that are authenticated externally.

To enable secure external authentication, do the following:

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 allowed:

   ```
   mts_dispatchers="dce, 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 allow Oracle access across cell boundaries. The way you define users depends on whether they are connecting within a single cell or across cell boundaries.

   > **Note:** The privileges shown in the remainder of this section are the minimum access privileges necessary. The actual set of privileges needed depends upon the instance and/or application.

   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;
   ```

Note: The entire CELL_NAME/SERVER_PRINCIPAL string must be 15 characters or less.

For example:

```
SQL> CREATE USER "CELL1/ORACLE" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL1/ORACLE";
```

If connecting to the database across multiple cells, specify both the *cell_name* and the *server_principal*.

```
SQL> CREATE USER "CELL_NAME/SERVER_PRINCIPAL" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL_NAME/SERVER_PRINCIPAL";
```

Attention: You must enclose the externally-identified account name in double quotes, 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";
```

Note: When using the above format, set the following parameter in protocol.ora 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. For example, 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 be as follows:

```
SQL> SELECT * FROM oracle.emp
```

If a user wants to access tables in an another account created for connections across cells, the command might be:

```
SQL> SELECT * FROM "CELL1/ORACLE".emp
```

More Information: For more information on external authentication, see *Oracle8i Distributed Database Systems.*

Setting up DCE Integration External Roles

This section explains the steps you follow to set up external roles for DCE integration, and how to connect to an Oracle database as SYSOPER or SYSDBA with DCE credentials.

Perform the following steps to set up external roles for DCE Integration:

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 14–2 provides a list of the syntax components.

Table 14–2 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

Note: For more details on external roles see the *Oracle8i Administrator's Guide.*

4. DCE authenticate to a user who is a member of a DCE group by performing a `dce_login` and a `klist` command.

The following is sample output from the dce_login and klist commands:

```
% dce_login oracle
Enter Password:
 % klistDCE
 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 is sample output showing a listing of 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
   ```

Connecting to an Oracle Database as SYSDBA or SYSOPER Using DCE

To connect to an Oracle database as SYSOPER or SYSDBA with DCE credentials, perform the following steps:

1. Create DCE groups that map to Oracle DBA and OPERATOR roles. DCE group names should adhere to the syntax presented in "Setting up DCE Integration External Roles" in this chapter. Add the externally authenticated user "oracle" as a member of the group or groups.

```
$ 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 explained in "Creating and Naming Externally-Authenticated Accounts" in this chapter.

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: 1998-12-04-10:28:22
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 1998-12-04-00:28:22 to 1998-12-04-10:28:22
Server: dce-rgy@dce.dlsun685.us.oracle.com
        valid 1998-12-04-00:28:22 to 1998-12-04-10:28:22
Server: dce-ptgt@dce.dlsun685.us.oracle.com
        valid 1998-12-04-00:28:26 to 1998-12-04-02:28:26
Client: dce-ptgt@dce.dlsun685.us.oracle.com     Server:
krbtgt/dce.dlsun685.us.o
racle.com@dce.dlsun685.us.oracle.com
        valid 1998-12-04-00:28:26 to 1998-12-04-02:28:26
Client: dce-ptgt@dce.dlsun685.us.oracle.com     Server:
dce-rgy@dce.dlsun685.us.
oracle.com
        valid 1998-12-04-00:28:27 to 1998-12-04-02:28:26
```

Note: List output shows the DCE group membership of Oracle.

5. Connect to the Oracle database as SYSBDA or SYSOPER. For example, enter the following:

```
SQL> connect /@oradce as SYSDBA
```

Configuring the Client

To configure a client for DCE Integration, you need to configure the following Net8 files with DCE address and parameter information, as described in this section:

- `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.

> **More Information:** See "Configuring Clients to Use DCE CDS Naming" in this chapter.

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 are as follows:

- DCE.AUTHENTICATION=*dce_secret*
- DCE.PROTECTION=*pkt_integ*
- DCE.TNS_ADDRESS_OID=1.3.22.1.5.1
- DCE.LOCAL_CELL_USERNAMES=TRUE

> **Note:** The default for DCE.LOCAL_CELL_USERNAMES is now TRUE. (It was set to FALSE in the DCE Integration 2.1.6 release.)

Configuration parameters are not case-sensitive: you can enter them in either uppercase or lowercase.

> **Note:** If the DCE.AUTHENTICATION entry is not specified, cell-wide default authentication is used.
>
> If the DCE.PROTECTION entry is not specified, cell-wide default protection is used.

DCE.AUTHENTICATION

The DCE.AUTHENTICATION parameter is optional. It indicates the authentication value to be used for each DCE RPC. The client's DCE_AUTHENTICATION value must be the same as the server's DEC_AUTHENTICATION value. The options are as follows:

Option	Description
NONE	No authentication
DCE_SECRET	DCE shared-secret key authentication (Kerberos)
DCE_SECRET	Default authentication level and recommended value
DEFAULT	Cell default

Note: It is recommended that DCE_SECRET be used for this parameter.

DCE.PROTECTION

DCE.PROTECTION is an optional field that specifies the data integrity protection levels for data transmission. The client's DCE_PROTECTION level must be equal to or greater than the server's DCE_PROTECTION level. The options are as follows:

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 DCE.TNS_ADDRESS_OID as follows:

```
DCE.TNS_ADDRESS_OID=1.3.22.1.x.x
```

> **More Information:** For information on how to determine if you must include this parameter and how to specify it, see "Modify the CDS Attributes File and Restart the CDS" in this chapter.

DCE.LOCAL_CELL_USERNAMES

DCE.LOCAL_CELL_USERNAMES is an optional parameter that defines the format used to specify the principal name (username) either with or without the cell name.

> **Note:** The choice you make for this parameter should be determined by whether users are making connections across cells, and if so, whether you have naming conventions that assure that users in different cells do not have duplicate names.

The options are as follows:

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

Configuring Clients to Use DCE CDS Naming

Clients typically use CDS to resolve Oracle service names to addresses. Follow the instructions in this section to configure CDS.

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.

> **More Information:** For instructions on how to install and configure the CDS Naming Adapter, see the DCE Integration installation instructions and "Task 3: Configure DCE CDS for Use by Oracle DCE Integration" in Chapter 13, "Configuring DCE for Oracle DCE Integration".

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.

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 with the following format to the `/opt/dcelocal/etc/cds_attributes` file.

   ```
   1.3.22.1.5.1    TNS_Address    char
   ```

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.

> **Note:** The first four digits of the TNS_Address attribute value, 1.3.22.1.x.y, are fixed under DCE-naming conventions.

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
```

> **Note:** 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 the CDS on the machine.

   The command to restart CDS can vary from platform to platform. For example, on IBM AIX, you can use `smit` to restart the CDS as follows.

   1. Enter the following:

      ```
      smit DCE
      ```
   2. Select Restart DCE/CDS Daemons.
   3. Select List.
   4. Select all CDS daemons available.

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`, containing service names or aliases and addresses. A sample file is shown below. The local `tnsnames.ora` file is used to map service names to addresses for use by Net8.

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. However, they are often the same.

The keyword value pair PROTOCOL=DCE is mandatory. It appears in the address section of a listener configuration file, `listener.ora`, and in the address section of a local naming configuration file, tnsnames.ora. It must be the same in both places.

The DCE parameter SERVER_PRINCIPAL is optional in a local naming configuration file, `tnsnames.ora`.

The DCE parameter SERVICE is mandatory. The value given for the DCE parameter (SERVICE= *dce_service_name*) must be the same in the listener configuration file, `listener.ora`, and the local naming configuration file, `tnsnames.ora`.

The Oracle parameter SID is mandatory. It identifies the Oracle system ID; each SID value must be unique on a node. This parameter is strictly local and is not used in DCE CDS.

More Information: For information on the local naming configuration file, `tnsnames.ora`, see the *Net8 Administrator's Guide*.

Load Oracle Connect Descriptors into CDS

A separate utility called `tnnfg` is provided with Oracle DCE Integration to load connect descriptors into CDS.

To load the Oracle service names or aliases 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)
```

Note: Enter the full pathname of the tnsnames.ora file in the previous command.

Also, ensure that the sqlnet.ora file exists in the same directory as the tnsnames.ora file.

This procedure loads the service names in tnsnames.ora into DCE CDS.

Note: 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.

Delete or Rename the tnsnames.ora File

If you are using SQL*Net 2.2 or earlier, after having loaded the tnsnames.ora file into DCE's CDS, Oracle Corporation recommends that you rename the file to tnsnames.bak, for example, or delete it. Otherwise, tnsnames.ora might be searched instead of CDS to resolve the service name to an address.

If you are using SQL*Net 2.3 or Net8, 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 in "Modify the sqlnet.ora File to Resolve Names in CDS" in this chapter.

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 Net8 you are using.

SQL*Net Release 2.3 and Later and Net8

For a client or server to use the DCE CDS Naming, the administrator needs to do the following:

- Ensure that the CDS Naming Adapter has been installed on that node
- 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 some reason, the next name resolution service is used, and so forth.

Connect to Oracle Servers in DCE

More Information: For information on how to connect to Oracle databases in a DCE environment, see Chapter 15, "Connecting to an Oracle Database in DCE".

15

Connecting to an Oracle Database in DCE

This chapter explains how to connect to an Oracle database after having installed Oracle DCE Integration and having configured both DCE and Oracle to use Oracle DCE Integration.

This chapter covers the following topics:

- Starting the Listener
- Connecting to an Oracle Database Server in the DCE Environment

Starting the Listener

Perform the following steps to start the listener.

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 to start the listener:

   ```
   % dce_login oracle orapwd
   % lsnrctl start LSNR_DCE
   ```

2. Verify that the server has registered its binding handler with rpcd as follows:

   ```
   % 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, enter the following:

   ```
   % cdscp show object "/.:/subsys/oracle/service_registry/dce_svc"
   ```

   This shows you the mapping in the CDS namespace that the listener has chosen for the endpoint. For example:

   ```
               SHOW
             OBJECT    /.../subsys/oracle/service_registry/dce_svc
                 AT   1995-05-15-17:10:52
   RPC_ClassVersion = 0100
            CDS_CTS = 1995-05-16-00:05:01.221106100/aa-00-04-00-3e-8c
            CDS_UTS = 1995-05-16-00:05:01.443343100/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

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
```

More Information: See *Oracle8i Distributed Database Systems*.

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 Net8 net service name.

For example:

```
% sqlplus scott/tiger@ORADCE
```

16

DCE and Non-DCE Interoperability

This chapter describes how clients outside DCE can connect to Oracle servers in DCE and how a local naming configuration file, `tnsnames.ora`, can be used for name lookup when CDS is accessible.

This chapter covers 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 Oracle and Net8 procedures to connect to an Oracle server in DCE.

Note: In this case, DCE security would not be available to clients. Also, service names would be located and resolved to network addresses in a tnsnames.ora file on the client, not using the CDS name server.

Following are samples of `listener.ora` and `tnsnames.ora` files as they would need to 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 two Oracle parameter files are needed for successful client/server communications. Create and modify these files using any text editor.

The parameter files are described in the following sections:

- The listener.ora File
- The tnsnames.ora File

The listener.ora File

The `tnsnames.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 laid out on a separate line, so it is easy to see the file's structure. This is the recommended format. If you must edit a `listener.ora` file manually, 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 onto 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 too. 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/oracle7)
         )
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/oracle7/network/trace
TRACE File_LISTENER=listener.trc
LOG_DIRECTORY_LISTENER=/usr/prod/oracle7/network/log
LOG_FILE_LISTENER=listener.log
```

The tnsnames.ora File

This file resides on both the client and the server nodes. It provides a list of the service names and addresses of all services on the network.

The following `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)
          )
          )
```

A user who wants to access the DB1 database 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 (in the context of Native Naming adapters) of the tnsnames.ora file 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 Net8

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 III

Oracle8*i* Security/Directory Integration

Part III of the *Oracle Advanced Security Administrator's Guide* presents Oracle's directory and security integration product, which provides single sign-on functionality in a client/server environment.

The following chapters describe how to set up enterprise user security in your Oracle environment:

- Chapter 17, "Managing Enterprise User Security"
- Chapter 18, "Using Oracle Wallet Manager"
- Chapter 19, "Oracle Enterprise Login Assistant"
- Chapter 20, "Using Oracle Enterprise Security Manager"

17

Managing Enterprise User Security

Enterprise user security provides single sign-on functionality in a client-server environment.

This chapter covers topics in the following sections:

Overview of Enterprise User Security

Administrators today must manage complex user information, keeping it current and secure. These tasks become all the more challenging with increased use of technology and a high user turnover in enterprises. For example, in a typical enterprise, each user can have multiple accounts on different databases. This means too many passwords for users to remember, and too many accounts for administrators 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 change the same day in order to guard against misuse of that user's old or unused accounts and privileges. However, in a large enterprise, with user accounts and passwords distributed over multiple databases, an administrator may not be able make all the changes as expeditiously as good security requires.

Enterprise user security management, introduced in Oracle8*i* Release 8.1.6, addresses these user, administration, and security challenges by centralizing storage and management of user-related information in an LDAP-compliant directory service. Now, when an employee changes jobs, the administrator needs to modify information in only one location, namely, the directory. This centralization lowers the cost of administration and makes the enterprise more secure.

Oracle8*i* enterprise user security provides single sign-on to Oracle8*i* using interoperable X.509 v3 certificates over Secure Sockets Layer (SSL) v3. It supports the following LDAP-compliant directory services:

- Oracle Internet Directory Release 2.0.5 or later
- Microsoft's Active Directory

It also provides a tool, Oracle Enterprise Security Manager, to create user entries in the directory.

This chapter discusses important concepts related to enterprise user security management in the following sections:

- About Directories
- Elements of Enterprise User Security Management
- How Enterprise User Security Management Works

About Directories

A directory is a way of organizing information so that you can find it easily. It lists objects—for example, people, books in a library, merchandise in a department store—and gives details about each one. A telephone book is a familiar enter of directory, a card catalog in a library is another, and a department store catalog still another.

In a computerized environment, a directory is a specialized database that stores collections of information about objects. The information in such a directory might represent any resources that require management—for example, employee names, titles, and security credentials, information about e-commerce partners, or about shared network resources such as conference rooms and printers.

Some of the key concepts for understanding directories are discussed in the following sections:

- Entries
- Distinguished Names and Directory Information Trees
- Naming Contexts
- Authorization and Access Control

Entries

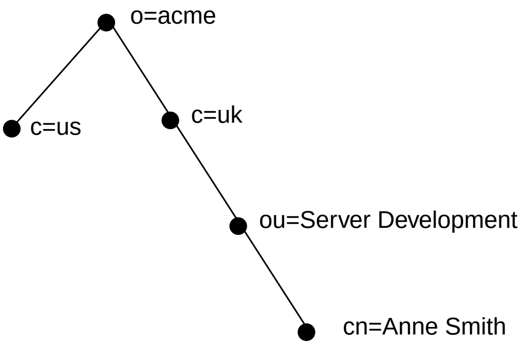
In a directory, each collection of information about an object is called an *entry*. Just as a typical telephone directory includes entries for people, an online directory might include entries for employees, conference rooms, e-commerce partners, or shared network resources such as printers.

Distinguished Names and Directory Information Trees

Each entry in a directory is uniquely identified by a *distinguished name (DN)*. The distinguished name tells you exactly where the entry resides in the directory's hierarchy, called a *Directory Information Tree (DIT)*.

To understand the relation between a distinguished name and a Directory Information Tree look at Figure 17–1.

Figure 17–1 A Directory Information Tree



The DIT in Figure 17–1 is structured along geographical and organizational lines. The branch on the right represents the entry for Anne Smith who works in the organization (`o`) Acme, in an organizational unit (`ou`) named Server Development, in the country (`c`) of Great Britain (uk).

The DN for this "Anne Smith" entry is:

```
cn=Anne Smith,ou=Server Development,c=uk,o=acme.
```

Note that the conventional format of a distinguished name places the lowest DIT component at the left, then follows it with the next highest component, thus moving progressively up to the root.

Naming Contexts

A directory divides its information into units called directory naming contexts. A directory naming context is a subtree that resides entirely on one server. It must be contiguous, that is, it must begin at an entry that serves as the top of the subtree, and extend downward to either leaf entries or references to subordinate naming contexts. It can range in size from a single entry to the entire DIT.

With Oracle8*i* Release 8.1.6, as we will see later in this chapter, you choose one or more naming contexts to contain Oracle enterprise information. We call these administrative contexts, and in each one, a container may be created to hold Oracle enterprise information. This container is called an Oracle Context.

Authorization and Access Control

Authorization is the process of ensuring that a user reads or updates only the information for which that user has privileges. When directory operations are attempted within a directory session, the directory server ensures that the user has the required permissions to perform those operations. Otherwise, the operation is disallowed. Through this mechanism, the directory server protects directory data from unauthorized operations by directory users. This mechanism is called access control.

Access control policies are captured in **Access Control Lists (ACLs)**. Typically, a given ACL is associated with each directory object and governs the access policies for that object.

ACLs specify the following:

- The users who can access objects in the directory
- The objects each user can access
- The access rights, or what the user can do with the object—for example, read or write

Elements of Enterprise User Security Management

There are a few main types of directory entries that relate to enterprise user security management, and these are discussed in the following sections:

- Enterprise Users
- Enterprise Roles
- Enterprise Domains
- Administrative Context
- Oracle Context

Enterprise Users

An **enterprise user** one that is defined and managed in a directory. Each enterprise user has a unique identity across an enterprise.

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**. A global role is managed in a directory, but its privileges are contained within a single database. An enterprise role is thus a container of global roles. For example, the enterprise role CLERK could contain the global role HRCLERK with its unique privileges on the Human Resources database, and the ANALYST role with its unique privileges on the Payroll database.

An enterprise role can be granted to or revoked from one or more enterprise users. For example, you could grant the enterprise role CLERK to a number of enterprise users who hold the same job. This information is protected in the directory, and only you, as the administrator, can manage users and grant and revoke their roles.

A user can be granted local roles and privileges in a database in addition to enterprise roles.

Note: The database obtains a user's global roles when the user logs in. If you change a user's global roles, 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, allocates enterprise roles to users and manages enterprise security.

Oracle Context

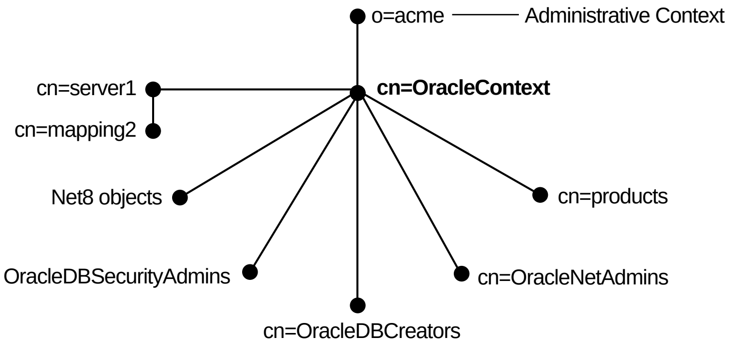
An **Oracle Context** (`cn=OracleContext`) is a special entry in the directory that contains various Oracle entries to support directory naming and enterprise user security. An Oracle Context contains three administrative groups, a products subtree, and, possibly, server and Net8 objects.

Administrative Context

The administrative context is the location for an Oracle Context. It can be any directory entry. During directory access configuration, which is completed with the Net8 Configuration Assistant during or after installation, you select an administrative context.

An administrative context, an Oracle Context, and its subtrees are shown in Figure 17–2.

Figure 17–2 An Oracle Context



WARNING: Do not modify the ACLs for the objects contained in an Oracle Context. Doing so breaks the security configuration for these objects.

See Also: Chapter 20 for a full discussion of Oracle Enterprise Security Manager.

Each of the three administrative groups is created with its associated ACL. The user who creates the Oracle Context with the Net8 Configuration Assistant automatically becomes the first member of these groups. The three administrative groups in an Oracle Context are:

Administrative Group	Description
OracleNetAdmins	Members of the OracleNetAdmins group (`cn=OracleNetAdmins,cn=OracleContext`) have `create`, `modify`, and `read` access to Net8 objects and attributes. The Net8 Configuration Assistant establishes these access rights for this group 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 or the OracleNetAdmins group.
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. The Net8 Configuration Assistant 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 Oracle Context. 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 The Net8 Configuration Assistant 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.

See Also: *Net8 Administrator's Guide* for instructions on adding members to the OracleNetAdmins group

In addition to the three administrative groups, an Oracle Context may also include other types of objects, for example:

Object	Description
Database server object	A database server object (represented as `cn=server1` in Figure 17–2 on page 17-7) contains information about a database server. It is created by the Oracle Database Configuration Assistant during the database installation and can be added later by members of the OracleDBCreators group by using Oracle Enterprise Security Manager. A database server object is the parent of database level *mapping objects* that contain mapping information between full or partial DNs and Oracle shared schema names. Database level mapping objects are created by the database administrator by using Oracle Enterprise Security Manager.
Net service name object	Net service name objects can be created during the database installation by using the Net8 Assistant. The can be created later by members of the OracleNetAdmins group.

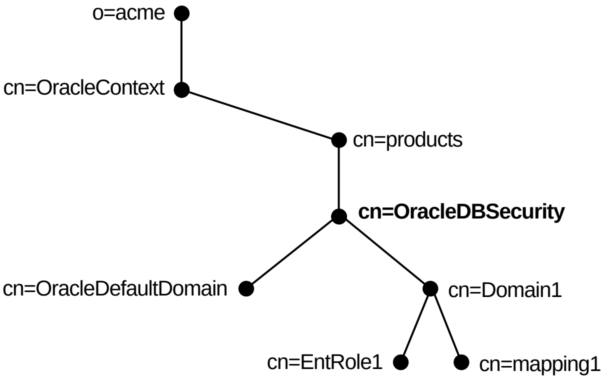
The Oracle Context also contains the database security subtree under the `cn=products` and `cn=OracleDBSecurity` container objects. This subtree contains enterprise domains, which are the groups of database servers and enterprise roles.

During database installation, a default enterprise domain (`cn=OracleDefaultDomain`) is established. The domain administrator can later add additional enterprise domains (represented in Figure 17–3 on page 17-10 as `cn=Domain1`) by using Oracle Enterprise Security Manager.

WARNING: Do not remove the default enterprise domain (cn=OracleDefaultDomain). It is required when registering a database by using the Oracle Database Configuration Assistant.

See Also: "Creating an Enterprise Domain" on page 20-15.

Figure 17–3 An Enterprise Domain



An enterprise domain subtree includes the following objects:

Enterprise role object

Contains information about global roles for each server and enterprise roles for the domain. These are created and managed by the domain administrator by using Oracle Enterprise Security Manager.

See Also: Creating an Enterprise Role within an Enterprise Domain on page 20-17.

Mapping objects

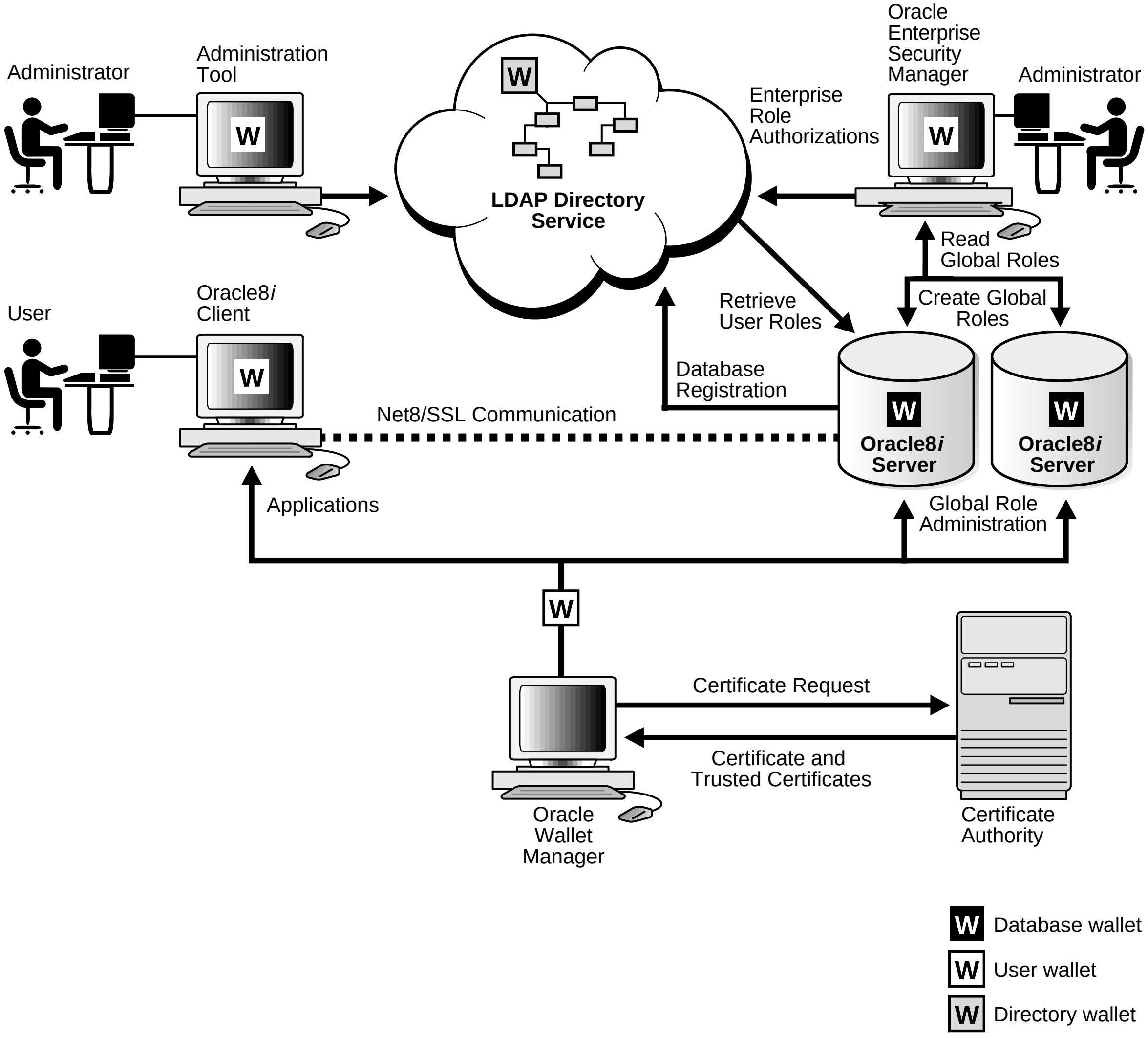
Each mapping object contains mapping information between a full or partial DN and an Oracle database user name. Mapping objects are created by the domain administrator for a particular domain. Mapping objects, as we saw earlier on page 17-9, also reside under server objects and are created by the database administrator for a particular database.

See Also: "Mapping an Enterprise User to a Shared Schema" on page 17-16.

Architecture of Enterprise User Security Management

Refer to Figure 17–4 on page 17-11 to see the relationships among the various components of this product.

Figure 17–4 Overview of Enterprise User Security



How Enterprise User Security Management Works

This section summarizes the interactions between the various components in enterprise user security management. In this summary, we are assuming that the user already has a wallet and that the user’s authentication to the database takes place by using SSL.

The enterprise user security management process looks like this:

1. The directory or other administrator, using Net8 Configuration Assistant, selects the administrative context in the directory and creates an Oracle Context.
2. A member of the OracleDBCreators group, using the Oracle Database Configuration Assistant, registers the database with the directory.
3. An administrator, using Oracle Enterprise Security Manager, sets up enterprise users and enterprise roles in the directory and domains.
4. An end user seeks to log in to the database.
5. The database retrieves from the directory the user’s enterprise roles, and authorizes any global roles that they contain that apply to that database.

User/Schema Separation

In most cases, users do not need *their own* accounts or schemas in a database. Rather, they typically need to access only *application* schemas. For example, suppose that users John, Firuzeh, and Jane need to access the Payroll schema on the Finance database. They do not need to create objects in the database, and therefore do not need their own schemas. They merely need access to the Payroll application.

Oracle8*i* Release 8.1.6 supports mapping many enterprise users stored in a directory to shared schema on an individual database. This separation of users from schemas reduces the cost of user administration by reducing the number of user accounts. It means that you do not need to create an account for a user—that is, a user schema—in multiple databases, in addition to creating the user in the directory. Instead, you can create a user in one location, namely, the directory, and “point” the user to a shared schema that other enterprise users can also access. For example, if John, Firuzeh and Jane 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 need to create only a single schema on each database—let’s call them SALES_APPLICATION and FINANCE_APPLICATION respectively—that all three users can access.

Setting Up User/Schema Separation

To configure User/Schema Separation, the local administrator or privileged user of a database needs to create at least one database schema in the database. Enterprise users can then be mapped to this schema.

Consider the process in the following example in which an enterprise security administrator creates a shared schema, then maps users to it.

1. The enterprise security administrator creates a shared schema called EMPLOYEE and the global role HRMANAGER on the HR database.
2. The administrator then uses Oracle Enterprise Security Manager to create and manage enterprise users and enterprise roles in the directory. For example, the administrator creates an enterprise user Jane and an enterprise role named MANAGER. The administrator then assigns the global role HRMANAGER to the enterprise role MANAGER.
3. The administrator then assigns enterprise roles to enterprise users in the directory. For example, the administrator assigns the enterprise role MANAGER to Jane.
4. The administrator then uses Oracle Enterprise Security Manager to map the user Jane in the directory to the shared schema EMPLOYEE on the HR database.

Now, when Jane 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 Jane and Scott automatically use the EMPLOYEE schema when connecting to the HR database, but each may get a different role.

User/Schema Separation Functionality and SSL

User/Schema Separation functionality relies on SSL for authentication to the database. SSL authentication occurs as follows:

1. Prior to connecting to a database, an enterprise user opens his or her wallet by providing a password.
2. When connecting, Oracle Advanced Security performs an SSL handshake with the database, during which it passes the user’s certificate to the server. This handshake authenticates the user to the server.

3. The database extracts the user’s DN from the user’s certificate. It then looks up the DN in the database.

   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 to a database schema. The database may find:

   - 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 per 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 allows multiple enterprise users sharing 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.

   - No mapping at all

4. 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 allows the user to log on.

5. The database then maps the user to the associated schema.

   For example, suppose that Jane is trying to connect to the HR database. Suppose, further, that the HR database does not find Jane’s DN. The HR database then looks up the Jane’s DN in the directory. The directory has a mapping of Jane to the shared schema EMPLOYEE and returns this schema. The database now logs Jane in and points her to the EMPLOYEE schema.

6. The database retrieves from the directory the global roles for this database for this user.

   The database also retrieves from its own tables any local roles and privileges associated with the database schema to which the user was 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. You do this by not registering the database in the directory by using the Oracle Database Configuration Assistant. If this configuration is set, then the database uses only local roles to determine what the user can access. This allows customers to use SSL for client authentication, without having to manage user privileges centrally. This configuration does not work with mapped users and shared schemas.

See Also:

- Chapter 10 for information on SSL
- Chapter 18 for information on wallets and Oracle tools for managing them

Continuing our example, suppose that the enterprise role MANAGER contains the global roles CLERK on the Payroll database and ANALYST on the HR database. When Jane, who has the enterprise role MANAGER, connects to the HR database, she uses the schema EMPLOYEE on that database. Her privileges on that database are determined by:

- The global role ANALYST
- Any local roles and privileges associated with the EMPLOYEE schema on the HR database

When she connects to the Payroll database, her privileges are determined by:

- The global role CLERK
- 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 application_user 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

To load entries one at a time, you can use Oracle Enterprise Security Manager. To load large numbers of entries, use other LDAP processes such as the Oracle Internet Directory bulkload tool.

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, contained in a user's X.509 certificate, to a database schema that the user will access. You create a mapping object by using Oracle Enterprise Security Manager. This mapping can be either:

- Full DN (entry-level) mapping
- Partial DN (subtree-level) mapping

See Also: "User/Schema Separation Functionality and SSL" on page 17-13.

When determining the schema to which it should connect the user, the database uses the following precedence rules:

1. It first looks for that schema locally.
2. If it does not find it, then it looks in the directory. Within the directory, it first looks under the *server* object, first for a full DN mapping, then for a partial DN mapping.
3. If it does not find a mapping object under the server object, then it looks under the *domain* object, first for a full DN mapping, then for a partial DN mapping.
4. If it does not find a mapping object in the domain object, then the database refuses the connection.

If there are some privileges that need to be granted to a certain group of users, you can do this by granting roles and privileges to a database schema. Everyone sharing such schema gets these local roles and local privileges in addition to their personal enterprise roles. However, you must exercise caution when doing this, because everyone who points to this shared schema can exercise the privileges assigned to this schema.

Summary

- User-schema separation eliminates the need to have a dedicated database schema on each database for every enterprise user. A single shared schema is sufficient.
- Each enterprise user can be mapped to a shared schema on each database the user needs to access. The user is then pointed to the shared schema when the user connects to a database.
- User-schema separation lowers the cost of managing users in an enterprise.

Current User Database Links

Current user database links are an Oracle Advanced Security feature.

Oracle8*i* supports a particular type of database link, called a current user database link. This link allows you to connect to a second database as another user, with that user's privileges. At the same time, it does not require that user's credentials be stored in the database link definition.

For example, a current user database link allows Jane, a user of the Accounts Payable database, to access the Human Resources database by connecting as Scott, and using Scott's credentials.

For Jane 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. Jane, 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 Human Resources databases, you would use the following command in each database:

```
CREATE USER Scott IDENTIFIED GLOBALLY AS 'CN=Scott,OU=Sales,C=US,O=Acme'
```

Note that the syntax for creating this kind of schema is slightly different from the syntax for creating a shared schema presented in "Creating a Shared Schema" on page 17-16. In this case, the schema is Scott’s alone. In order for the current user database link to work, the schema created for Scott is not shared with other users.

Current user database links operate only between databases within a single enterprise domain, and only if that domain is trusted. You specify a domain as trusted by using Oracle Enterprise Security Manager. If you want to specify as untrusted a database that is part of a trusted enterprise domain, you use the PL/SQL package DBMS_DISTRIBUTED_TRUST_ADMIN. To obtain a list of trusted servers, you use the TRUSTED_SERVERS view.

See Also:

- *Oracle8i Distributed Database Systems* for additional information about current user database links
- *Oracle8i SQL Reference* for more information on syntax
- *Oracle8i Supplied PL/SQL Packages Reference* for information on the PL/SQL package DBMS_DISTRIBUTED_TRUST_ADMIN
- *Oracle8i Reference* for information about the TRUSTED_ SERVERS view.

Oracle Enterprise User Security Components

Oracle enterprise user security incorporates administration tools discussed in the following sections:

- Oracle Wallet Manager
- Oracle Enterprise Login Assistant
- Oracle Enterprise Security Manager
- Oracle Internet Directory

Oracle Wallet Manager

Oracle Wallet Manager is a stand alone Java application that wallet owners and security administrators use to manage and edit the security credentials in their Oracle wallets. These tasks include:

- Generating a public-private key pair and creating a certificate request for submission to a certificate authority (CA)
- Installing a certificate for the entity
- Configuring trusted certificates for the entity
- Opening a wallet to enable access to PKI-based services
- Creating a wallet that can be later opened using the Oracle Enterprise Login Assistant

See Also: Chapter 18, for detailed information on how to use this application.

Oracle Enterprise Login Assistant

Use Oracle Enterprise Login Assistant to open and close a user wallet in order to enable or disable secure SSL based communications for an application. This tool provides a subset of the functionality provided by Oracle Wallet Manager.

> **See Also:** Chapter 19, for detailed instructions on how to use this application.

Oracle Enterprise Security Manager

Use Oracle Enterprise Security Manager to:

- Create new enterprise domains and manage their security
- Manage the administration groups in the Oracle Context

> **See Also:** Chapter 20, for detailed instructions on how to use this application.

Oracle Internet Directory

Oracle Internet Directory is a Lightweight Directory Access Protocol (LDAP) v3 directory service implemented as an application on the Oracle8*i* database. It enables the retrieval of information about dispersed users and network resources.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for detailed instructions on how to use this application.

Installing and Configuring Enterprise User Security

This section gives you an overview of how to set up enterprise user security. The required steps are as follows:

- Task 1: Install or Identify a Certificate Service
- Task 2: Install and Configure a Directory Service
- Task 3: Install and Configure One or More Databases
- Task 4: Configure Database Clients
- Task 5: Install and Configure Oracle Enterprise Security Manager
- Task 6: Create and Configure Enterprise Users
- Task 7: Log In as the Enterprise User

Task 1: Install or Identify a Certificate Service

Oracle Wallet Manager requires you to have a certificate authority in your environment. The CA must be able to process PKCS#10 certificate requests in Base 64 format, and return X509v3 certificates, also in Base 64 format.

> **See Also:** Chapter 18 for a fuller description of certificate authorities and Oracle Wallet Manager

Task 2: Install and Configure a Directory Service

1. Install an LDAP v3-compliant directory service. Oracle8*i* Release 8.1.6 supports the following:
   - Oracle Internet Directory Release 2.0.5 or later
   - Microsoft's Active Directory
2. Set up the directory for two-way SSL authentication. This involves creating an Oracle wallet for the directory.

   If you are using Oracle Internet Directory, use Oracle Wallet Manager and Oracle Directory Manager to create a wallet for the directory and to configure SSL. If you are using Microsoft Active Directory or Novell Directory Service, see the product-specific documentation for instructions for enabling two-way SSL authentication.
3. Start the directory.
4. Create directory naming contexts as desired, and, beneath them in the DIT, create desired administrative contexts.

   For example, if your naming context is dc=my_company,`dc=com`, you may choose as an administrative context `dc=us,dc=my_company,dc=com`.
5. If you are using Oracle Internet Directory, use the Net8 Configuration Assistant to create Oracle Contexts as desired. The Oracle schema is already installed. You can also create Oracle Contexts later during a database installation.

   If you are not using Oracle Internet Directory, you can use the Net8 Configuration Assistant later to install the Oracle schema and desired Oracle Contexts.

See Also:

- "Administrative Context" on page 17-6
- The chapter on configuring naming methods in *Net8 Administrator's Guide* for instructions on installing the Oracle directory schema into the directory and creating an Oracle Context
- *Oracle Internet Directory Administrator's Guide*—if you are using Oracle Internet Directory— for instructions on using Oracle Wallet Manager and Oracle Directory Manager to create a wallet for the directory

Task 3: Install and Configure One or More Databases

This involves performing the tasks described in the following subsections:

- Install Oracle8i Release 8.1.6 Database Software
- Set Up Directory Access for ORACLE_HOME
- Use Oracle Database Configuration Assistant to Register the Database in the Directory
- Configure Net8 for Listener and Database SSL Support
- Create and Open a Database Wallet
- Perform Database Logout for Extra Security
- Verify Database Installation

Install Oracle8*i* Release 8.1.6 Database Software

Use the Oracle Universal Installer to install Oracle8*i* Release 8.1.6 databases.

Before installation, obtain from the directory administrator:

- A directory login name and password. The login name must belong to someone in the OracleDBCreators group. The user who created the Oracle Context is automatically a member of this group.
- Either of the following:
  - An administrative context that holds the Oracle Context that serves as the location in the directory for this database's future entry
  - An administrative context in which to create a new Oracle Context

During installation, select Oracle8*i* Enterprise Edition and the Custom installation type with Oracle Advanced Security.

> **See Also:** Oracle8*i* Release 8.1.6 installation documentation for your platform for detailed instructions on how to install and create a database.

Set Up Directory Access for ORACLE_HOME

You may complete this during or after the installation of Oracle8*i* Release 8.1.6. If so then you do not need to do it again.

To configure directory service access configuration by using the Net8 Configuration Assistant:

1. Run the Net8 Configuration Assistant.
2. Choose the Directory Service Access configuration option.
3. Perform directory access configuration for a server option, and continue.

   If your directory does not yet have the required Oracle schema, you are prompted to install it in the directory. If there is no Oracle Context set up under an administrative context, you are prompted to create the Oracle Context.

> **Note:** If you do not install clients, and ORACLE_HOME is set to a database server ORACLE_HOME, then you will need at least one new TNS_ADMIN directory with a `sqlnet.ora` file. That `sqlnet.ora` file must have *no* wallet location. This ensures that SSL uses the default location of the wallet for the operating system user.

> **See Also:** The chapter on configuring naming methods in *Net8 Administrator's Guide* for instructions on setting up directory access for the database.

Use Oracle Database Configuration Assistant to Register the Database in the Directory

Oracle Database Configuration Assistant asks if you want to register the database in the directory. Choose Yes. This creates a database service object underneath your chosen Oracle Context in the directory.

Configure Net8 for Listener and Database SSL Support

Net8 needs to be configured for SSL on both the Listener and the Database. The Listener must have a listening endpoint which is configured for the TCP/IP with SSL protocol, and the location of the Database's wallet must be specified. You do this using the Net8 Configuration Assistant as described in "Enabling SSL" on page 10-10. The suggested wallet locations for a database are:

- On UNIX:

  `/etc/ORACLE/WALLETS/DATABASES/`*`database_name`*

- On Windows NT:

  `C:\WINNT\Profiles\DATABASES\`*`database_name`*

Create and Open a Database Wallet

This involves performing the tasks described in the following subsections:

- Create a Wallet
- Open the Wallet

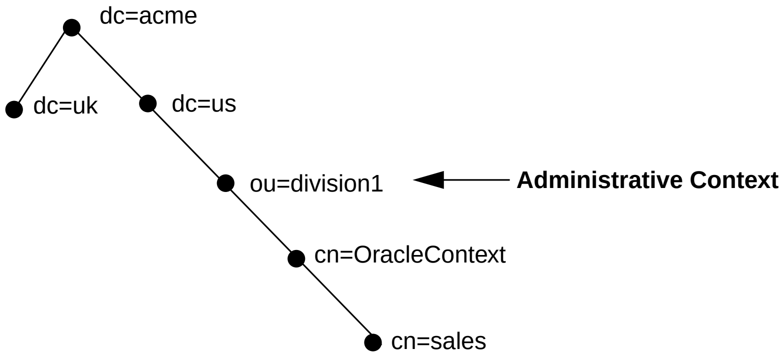
Create a Wallet To do this:

1. Create a directory for the wallet according to the location you specified in "Configure Net8 for Listener and Database SSL Support" on page 17-24 when using the Net8 Assistant.
2. Run Oracle Wallet Manager to create a new wallet for the database. Enter:

   ```
   owm
   ```

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 DN of the database exactly. The DN of the database is: `cn=`*`database_name`*`,cn=OracleContext,cn=`*`administrative_context`* and is found in the initialization parameter file in the parameter RDBMS_SERVER_DN.

   Be sure to match the case of the characters exactly. For example, if the global database name chosen during installation is `sales.us.acme.com`, and the administrative context selected within Net8 Configuration Assistant is `ou=division1,c=us,o=acme,` then the DIT for this DN would look like this:



The full DN of the database that you would type into Oracle Wallet Manager would be:

```
cn=sales,cn=OracleContext,ou=division1,c=us,o=acme
```

Note: `cn=OracleContext` must be included in the DN immediately after the simple database name.

4. Send the certificate request to your certificate authority.
5. Get the certificate text for the CA *trusted* certificate. The CA trusted certificate is sometimes known as a root key certificate.

   The certificate text includes the lines BEGIN CERTIFICATE and END CERTIFICATE and the text within these two lines.
6. Paste this certificate into the database wallet using the Oracle Wallet Manager Import Trusted Certificate function.
7. Get the certificate text for the *database* certificate. The certificate text includes the lines BEGIN CERTIFICATE and END CERTIFICATE and the text within these two lines.
8. Paste the certificate text into the certificate using the Oracle Wallet Manager Import User Certificate function.

Open the Wallet In order for users to access the database using two-way SSL authentication, the database wallet must be open, and the listener must be running. To open the wallet and run the listener:

1. Shut down the listener. The listener needs to read the database's open wallet, so the database must log on before the listener can be started. Enter:

   ```
   lsnrctl stop
   ```

2. In the 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.
3. Save the wallet to the directory you set up when you performed the task described in "Create a Wallet" on page 17-24.
4. Start the Listener. Enter:

   ```
   lsnrctl start
   ```

   > **See Also:** Chapter 18 for detailed instructions on how to create a wallet.

5. The database wallet will now remain open, and the database will be able to participate in authenticated communications using SSL.

Perform Database Logout for Extra Security

> **Important: If the database will be shut down for an extended period of time, perform a database logout and close the wallet for extra security.**

To log out of the database:

1. Stop the listener. Enter:

   ```
   lsnrctl stop
   ```

2. Start Oracle Wallet Manager. Enter:

   ```
   owm
   ```

3. Clear the Autologin check box.
4. Save your changes.

5. Restart the listener. Enter:

   ```
   lsnrctl start
   ```

Verify Database Installation

Follow the steps listed below to verify that the database was successfully installed.

1. Verify that there is a `cwallet.sso` file located in the database wallet directory. If there is no `cwallet.sso` file in the directory, 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 the LSNRCTL START command is successful and that the listener process is listening on the SSL port designated in the `listener.ora` file. Enter:

   ```
   lsnrctl status
   ```

3. Verify that there is an `ldap.ora` file located in `$ORACLE_HOME/network/admin`.

   If there is no `ldap.ora` file, then the Net8 Configuration Assistant failed. Verify that the ORACLE_HOME is set and TNS_ADMIN is not set. Rerun Net8 Configuration Assistant.
4. Use the directory administration tool to verify that a database entry exists under the Oracle Context you specified when you ran the Net8 Configuration Assistant. 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 try to register the database again. Enter:

   ```
   dbassist
   ```

5. Modify the database.

Task 4: Configure Database Clients

Once you have installed Oracle8*i* clients, configure Net8 on the client. Do this by using the Net8 Configuration Assistant.You may complete this during or after installation of the Oracle8*i* Release 8.1.6.

Because you will be using an LDAP directory service for enterprise security, you may also want to use Net8 directory naming. Net8 directory naming allows the client to connect to the database using the database entry registered with the directory by Oracle Database Configuration Assistant. Or you can use one of the other Net8 naming methods, such as Local naming (`tnsnames.ora` file), to configure a net service name for the database.

Enable clients to connect and authenticate to a database by using SSL. Use the Net8 Assistant to configure SSL. Do not enter a wallet location.

See Also:

- *Net8 Administrator's Guide*
- Chapter 10 for information on configuring SSL

Note: Wallets for specific users are set up when you create enterprise users. See Chapter 20 for instructions on creating enterprise users.

Task 5: Install and Configure Oracle Enterprise Security Manager

See: Installing and Configuring Oracle Enterprise Security Manager on page 20-3.

Once you have installed and configured Oracle Enterprise Security Manager, set up an enterprise domain. The following table lists the tasks you perform when you set up an enterprise domain, and points you to the corresponding instructions:

Task	Instructions
Create an enterprise domain.	"Creating an Enterprise Domain" on page 20-15
Make the enterprise domain trusted / untrusted. Only a trusted domain allows current user database links between member databases.	"Creating an Enterprise Domain" on page 20-15
Create enterprise roles in the domain.	"Creating an Enterprise Role within an Enterprise Domain" on page 20-17
Use Oracle Enterprise Security Manager to make the database a member of the desired enterprise domain.	"Adding a Database to an Enterprise Domain" on page 20-16 "Removing a Database from an Enterprise Domain" on page 20-23

Task	Instructions
Create global roles on the databases. The SQL*Plus command is: `CREATE ROLE` *`rolename`* `IDENTIFIED GLOBALLY`	*Oracle8i SQL Reference*
Assign a global role to each enterprise role.	"Creating an Enterprise Role within an Enterprise Domain" on page 20-17

Task 6: Create and Configure Enterprise Users

The following procedures assume that you have successfully installed all of the enterprise user security components. You should also have started the directory and the Oracle Enterprise Security Manager.

Create a new enterprise user by performing tasks described in the following subsections:

- Add a New Enterprise User to the Directory
- Create a User Wallet
- Create the User on the Databases
- Authorize the User

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

You may want to populate the directory with users before using Oracle Enterprise Security Manager.

See Also:

- "Administering Enterprise Databases, Domains, and Users" on page 20-9 for instructions on 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

Create a User Wallet

> **See:** Chapter 18 for instructions on creating a wallet by using Oracle Wallet Manager.

Create the User on the Databases Use Oracle Enterprise Manager or SQL*Plus to add a user to the desired database. For example:

```
CREATE USER fred IDENTIFIED GLOBALLY AS
'cn=frederick,ou=division1,c=us,o=oracle'
```

The user name must match the DN that is found in the user's wallet and in the directory.

Alternatively, to create a shared schema to which multiple enterprise users may be mapped, you could enter:

```
CREATE USER guest IDENTIFIED GLOBALLY AS '';
```

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, created in "Create the User on the Databases" on page 17-30. This step is optional.

- Enterprise role authorization

  Use Oracle Enterprise Security Manager to create and grant enterprise roles to the enterprise user in the directory. In order to do this, you must first set up an enterprise domain.

> **See Also:**
>
> - Task 5: Install and Configure Oracle Enterprise Security Manager on page 17-28 for information about setting up an enterprise domain
> - "Administering Enterprise Users" on page 20-24

Task 7: Log In as the Enterprise User

This involves two tasks:

- Open the User Wallet
- Connect to the Database

Open the User Wallet

In "Task 6: Create and Configure Enterprise Users" on page 17-29, you created a wallet for the enterprise user. The enterprise user must now open this wallet in order to log in to the database. Use Oracle Enterprise Login Assistant to open or close the wallet.

Opening a user wallet generates a single sign-on file and allows authentication to the SSL adapter.

To open a user wallet:

1. Launch Oracle Enterprise Login Assistant by typing the command:

   ```
   elogin
   ```

2. Enter the wallet password when you are prompted by the system.
3. Enable Autologin, then save your changes.

   Your wallet will now remain open, and you will have authenticated communications using the SSL adapter.

4. Set ORACLE_HOME.
5. If the ORACLE_HOME is set to a *server* ORACLE_HOME, then you must set the TNS_ADMIN environment variable to point to the directory where you placed the `sqlnet.ora` file that you created in "Task 4: Configure Database Clients" on page 17-27.

   If you have a separate client ORACLE_HOME, then you do not need to set the TNS_ADMIN environment variable.

To close a user wallet:

In the Oracle Enterprise Login Assistant, click Autologin > Logout to disable authentication with the SSL adapter.

> **See Also:** Chapter 19 for instructions on using Oracle Enterprise Login Assistant.

Connect to the Database

You have the option of doing this by using SQL*Plus. To do this, launch SQL*Plus and enter:

```
CONNECT /@connect_identifier
```

where *connect_identifier* is the net service name you set up in "Task 4: Configure Database Clients" on page 17-27. If you are using Net8 directory naming, the connect identifier is the simple database name.

Troubleshooting Enterprise User Login

This section contains a list of potential problem areas and ways to alleviate them.

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, 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, then 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.

TNS Lost Connection

This error may indicate that you attempted to configure a domestic cipher suite. Run the Net8 Assistant again, and be sure that you select the "Show domestic cipher suites" radio 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 in the wallet that is used to connect does not match the DN in the CREATE USER statement for any schema in the database, nor does it match the DN in any relevant mapping.

Check the case, spaces, and spelling of the distinguished name of the user that was created in the database. Enter:

```
SELECT external_name FROM user_users;
```

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.

18

Using Oracle Wallet Manager

Security administrators use Oracle Wallet Manager to manage public-key security credentials on Oracle clients and servers. Oracle Wallet Manager is used to create wallets that can be later opened by using either the Oracle Enterprise Login Assistant or the Oracle Wallet Manager.

> **See Also:** Chapter 19, for information on how to open and close wallets for secure SSL communications by using Oracle Enterprise Login Assistant.

The topics are covered in the following sections:

- Overview
- Security Concepts
- Using Oracle Wallet Manager with Oracle Application Server
- Starting Oracle Wallet Manager
- Managing Wallets
- Managing Certificates

Overview

Public-key cryptography requires entities, that want to communicate in a secure manner, to possess certain security credentials. This collection of security credentials is stored in a wallet.

Security credentials consist of a public/private key pair, a "user" certificate, a certificate chain, and "trusted" certificates. Each entity that participates in a public key system must have a public/private key pair. The public key for an entity is published in a user certificate so, for example, other entities that want to send it secure information can encrypt that information with the recipient entity’s public key. Another use for a public key is for an entity that receives a communication to validate the sender’s organizational affiliation—this is the most typical application in Oracle environments today. Oracle Wallet Manager generates public/private key pairs for clients and servers.

A certificate authority (CA) issues public key certificates. A certificate contains a unique serial number assigned to it by the CA, an algorithm identifier that identifies which algorithm was used to sign that certificate, the name of the CA that issued that certificate, a pair of dates between which the certificate is valid, the certificate user’s name, the entity’s public key, and the CA’s signature.

A trusted certificate, sometimes known as a root key certificate, typically belongs to a third party entity that is trusted to issue certificates. It is obtained in a secure manner and, operationally, does not need to be validated for its authenticity each time it is accessed. A client or a server uses a trusted certificate to validate that an entity is who it claims to be by verifying that entity’s certificate. Typically, certificate authorities you trust issue the user certificates. Oracle provides several default trusted certificates, so users do not have to install their own. These trusted certificates also enable servers to perform SSL authentication to clients who have wallets containing only trusted certificates.

Clients and servers use these credentials to access secure services, such as SSL, using public key cryptography. A wallet also represents a storage facility that is location and type transparent once it is opened.

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 certificate authority (CA).
- Installing a certificate for the entity.

- Configuring trusted certificates for the entity.
- Opening a wallet to enable access to PKI-based services.
- Creating a wallet that can be later opened by using either the Oracle Enterprise Login Assistant or the Oracle Wallet Manager.

Security Concepts

General security concepts and their associated definitions are as follows:

Table 18–1 Security Concepts and Definitions

Concept	Definition
Authentication	The recipient of an authenticated message can be certain of the message's origin (its sender). Authentication reduces the possibility that another person has impersonated the sender of the message.
Authorization	The set of privileges available to an authenticated entity.
Certificate	An ITU x.509 v3 standard data structure that securely binds an identity to a public key. A certificate is created when an entity's public key is signed by a trusted identity: a certificate authority. This certificate ensures that the entity's information is correct and that the public key actually belongs to that entity.
Certificate Authority	An application that creates public key certificates with a high level of assurance and trust in this function.
Confidentiality	A function of cryptography. Confidentiality guarantees that only the intended recipient(s) of a message can view the message (decrypt the ciphertext).
Cryptography	The act of writing and deciphering secret code resulting in secure messages.
Decryption	The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).
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.

Table 18–1 Security Concepts and Definitions

Concept	Definition
Encryption	The process of disguising the contents of a message (plaintext) and rendering it unreadable (ciphertext) to anyone but the intended recipient.
Identity	A user who is certified as being the entity it claims to be.
Integrity	The guarantee that the contents of the message received were not altered from the contents of the original message sent.
Non-repudiation	Undeniable proof of the origin, delivery, submission, or transmission of a message.
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 the recipient's private key. When public-key encryption is used with symmetric key encryption, DES for example, only the DES key is encrypted with the recipient's public key. The message itself is encrypted with a DES key: this is more efficient.
Public/Private Key Pair	A mathematically related set of two numbers where one is called the private key and the other is called the public key. The two numbers are related, but it is mathematically infeasible to derive the private key from the public key. Public keys are typically made widely available, while private keys are available only to their owners. Data encrypted with a public key can only be decrypted with its associated private key and vice versa. Data encrypted with a public key cannot be decrypted with the same public key.
Trusted Certificate	A trusted certificate, sometimes known as a root key certificate, is a third party identity that is trusted. The trust is used when an identity is being validated as the entity it claims to be. Typically, the certificate authorities you trust issue user 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.

Table 18–1 Security Concepts and Definitions

Concept	Definition
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.
Wallet Resource Locator	A wallet resource locator (WRL) provides all the necessary information to locate the wallet. It is a path to an operating system directory which contains a particular wallet.
WRL	See Wallet Resource Locator.
X.509	The public key certificate can be created in various data formats. The X.509 v3 format from ITU is a widely used format.

Using Oracle Wallet Manager with Oracle Application Server

When using the Oracle Application Server (OAS), you must install the Oracle Wallet Manager on a primary node and on each remote node in a multi-node configuration. After you install the product on each node you must then copy the wallet from the primary node to each of the remote nodes.

Starting Oracle Wallet Manager

Refer to your platform-specific documentation for instructions on how to start Oracle Wallet Manager.

Managing Wallets

This section gives you detailed instructions on 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.

This section covers topics in the following subsections:

- Creating a New Wallet
- Opening an Existing Wallet
- Closing a Wallet
- Saving Changes
- Saving the Open Wallet to a New Location
- Saving in System Default
- Deleting the Wallet
- Changing the Password
- Using Auto Login

Creating a New Wallet

Create a new wallet as follows:

1. Click Wallet > New from the menu bar.

   The New Wallet dialog box is displayed.

2. Read the recommended guidelines for creating a password, then enter a password in the Wallet Password field.
3. Re-enter that password in the Confirm Password field.
4. Click OK to continue.
5. An Alert displays, informs you that a new empty wallet has been created, and prompts you to decide whether you want to create a certificate request: see “Creating a Certificate Request”.

   If you Click Cancel, you are returned to the Oracle Wallet Manager main window. The new wallet you just created is displayed in its left pane. The certificate has a status of Empty, and the wallet displays its default trusted certificates.

6. Click 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.

> **Note:** Because an Oracle wallet contains a user's credentials that can be used to authenticate the user to multiple databases, it is especially important to choose a strong password for the wallet. If a malicious user guesses the password to a user's wallet, then the malicious user could access all the databases that the user can access.
>
> Oracle Corporation recommends that you choose a password that is not too short, is not easily guessed, and has some complexity (such as including a symbol or numeric character). A reasonably strong password has at least six characters, and contains at least one symbol or number (so that it is not a word that can be found in the dictionary), for example, `gol8fer`. Also, it is prudent security practice for users to change their passwords periodically, such as once a month, or once a quarter.

Opening an Existing Wallet

Open a wallet that already exists in the file system directory as follows:

1. Click Wallet > Open from the menu bar.

   The Select Directory dialog box displays.

2. Navigate to the correct directory location in which the wallet is located.

3. Click to select the directory.

4. Click OK.

   The Open Wallet dialog box displays.

5. Enter the wallet password in the Wallet Password field.

6. Click OK.
7. A message at the bottom of the window displays the message “Wallet opened successfully”.
8. You are returned to the Oracle Wallet Manager main window. The wallet’s certificate and its trusted certificates are displayed in the left pane.

Closing a Wallet

Close an open wallet as follows.

1. Click Wallet > Close to close the open wallet in the currently selected directory.
2. A message at the bottom of the window confirms that the wallet is closed.

Saving Changes

Save your changes to the current open wallet as follows.

1. Click Wallet > Save to save changes to the current open wallet.
2. 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. Click Wallet > Save As.

   The Select Directory dialog box displays.
2. Select the directory location in which to save the wallet.
3. Click OK.

> **Note:** You will get the following message if a wallet already exists in the selected directory: "A wallet already exists in the selected path. Do you want to overwrite it?". Click Yes to overwrite the existing wallet, or click No to save the wallet to another directory.

4. 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. This will make the current open wallet the wallet that will be used by SSL.

1. Click Wallet > Save in System Default.
2. A message at the bottom of the window confirms that the wallet was successfully saved in the system default wallet location.

Deleting the Wallet

Delete the current open wallet as follows:

1. Click Wallet > Delete.
2. The Delete Wallet dialog box appears.
3. Review the displayed wallet location to verify you are deleting the correct wallet.
4. Enter the wallet password.
5. Click OK.
6. A dialog panel appears to inform you that the wallet was successfully deleted.

Note: Any open wallet in an application memory will remain in memory until the application exits. Therefore, deleting a wallet that is currently in use using Oracle Wallet Manager will not immediately affect system operation.

Changing the Password

A password change becomes effective immediately. The wallet is saved to the currently selected directory encrypted with the new password.

Change the password on the current open wallet as follows:

1. Click Wallet > Change Password.

   The Change Wallet Password dialog box appears.
2. Enter the existing wallet password.

3. Enter the new password. Remember to follow the password guidelines.
4. Re-enter the new password.
5. Click OK.

   A message at the bottom of the window informs you 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 is open in memory.

You need to enable Auto Login if you want single sign-on to multiple Oracle databases.

Enabling Auto Login

Enable Auto Login as follows:

1. Click Wallet from the menu bar.
2. Click the check box next to the Auto Login menu item.
3. A message at the bottom of the window displays "Autologin enabled."

Disabling Auto Login

Disable Auto Login as follows:

1. Click Wallet from the menu bar.
2. Click the check box next to the Auto Login menu item.
3. 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 explains how to manage both kinds of certificate, and does so 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 certificate authority. Several trusted certificates are installed by default when you create a new wallet.

Managing User Certificates

Managing user certificates involves performing the following tasks:

- Creating a Certificate Request
- Exporting a User Certificate Request
- Importing the User Certificate into the Wallet
- Removing a User Certificate from a Wallet

Creating a Certificate Request

The actual certificate request becomes part of the wallet. You can reuse any certificate request to obtain a new certificate. However, you can not edit an existing certificate request, so only store a correctly filled out certificate request in a wallet.

Create a PKCS #10 certificate request as follows:

1. Click Operations > Create Certificate Request.

   The Create Certificate Request dialog box displays.

2. Enter the following information:

Table 18–2 Create a Certificate Request Fields and Definitions

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: for example, Finance.
Organization	Optional—Enter the name of the identity’s organization, for 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. Recommendation—Enter the full state name, because some certificate authorities do not accept two–letter abbreviations.
Country	Mandatory—Click the drop down list to view a list of country abbreviations. Click to select the country in which the organization is located.
Key Size	Click the drop down box to view a list of key sizes to use when creating the public / private key pair.
Advanced	Click 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.

3. Click OK. An Oracle Wallet Manager dialog box informs you that a certificate request was successfully created. You can now either copy the certificate request text from the body of this dialog panel and paste it into an e-mail message that you send to a certificate authority, or you can export the certificate request to a file.

4. Click OK. You are returned to the Oracle Wallet Manager main window. The status of the certificate is changed to Requested.

Exporting a User Certificate Request

You save the certificate request in a file system directory when you elect to export a certificate request.

1. Click Operations > Export Certificate Request from the menu bar.

   The Export Certificate Request dialog box appears.

2. Enter the file system directory in which to save your certificate request, or navigate the directory structure under Folders.
3. Enter the name of the file to which you want to save your certificate request in the Enter File Name field.
4. Click 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.

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 or file you receive from the certificate authority. Include the lines Begin Certificate and End Certificate.
2. Click Operations > Import User Certificate from the menu bar.

   The Import Certificate dialog box appears.

3. Click the Paste the Certificate radio button, and click 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 click 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 wallet status changes to Ready.

Selecting a File that Contains the Certificate To select the file:

1. Click Operations > Import User Certificate from the menu bar.
2. Click the Select a file... certificate radio button, and click OK.

   The Import Certificate dialog box appears.
3. Enter the path or folder name of the certificate location.
4. Click to select the name of the certificate file (for example, `cert.txt`).
5. Click OK. A message at the bottom of the window displays to inform you that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the wallet status is changes to Ready.

Removing a User Certificate from a Wallet

1. Click Operations > Remove User Certificate.

   A dialog panel appears and prompts you to verify that you want to remove the user certificate from the wallet.
2. Click Yes. You are returned to the Oracle Wallet Manager main panel, and the certificate will display a status of Requested.

Managing Trusted Certificates

A trusted certificate is the certificate of the issuer of the certificate you requested. Managing trusted certificates consists of performing the tasks discussed in the following sections:

- 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, and GTE CyberTrust when you create a new wallet.

Pasting the Trusted Certificate To paste the trusted certificate:

1. Click Operations > Import Trusted Certificate from the menu bar. The Import Trusted Certificate dialog panel appears.
2. Click the Paste the Certificate radio button, and click 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 you received that also contained the user certificate. Include the lines Begin Certificate and End Certificate.
4. Paste the certificate into the window, and click OK. A message at the bottom of the window informs you that the trusted certificate was successfully installed.
5. Click OK.

   You are returned to the Oracle Wallet Manager main panel, and the trusted certificate is displayed at the bottom of the Trusted Certificates tree.

Selecting a File that Contains the Trusted Certificate To select the file:

1. Click Operations > Import Trusted Certificate from the menu bar. The Import Trusted Certificate dialog panel displays.
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. Click OK. A message at the bottom of the window displays to inform you that the trusted certificate was successfully imported into the wallet.
5. Click OK to dismiss the dialog panel. You are returned to the Oracle Wallet Manager main panel, and the trusted certificate is displayed at the bottom of the Trusted Certificates tree.

Removing a Trusted Certificate

Remove a trusted certificate from a wallet as follows:

1. Select the trusted certificate listed in the Trusted Certificates tree.
2. Click Operations > Remove Trusted Certificate from the menu bar.

A dialog panel displays and 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. Click Yes. The selected trusted certificate is removed from the Trusted Certificates tree.

Warning: Certificates that are signed by a trusted certificate are no longer verifiable when you remove that trusted certificate 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

Export a trusted certificate to another file system location as follows:

1. Click Operations > Export Trusted Certificate.

   The Export Trusted Certificate dialog box appears.

2. Enter the file system directory in which to save your trusted certificate, or click Browse to display the directory structure.
3. Enter the name of the file to which you want to save your trusted certificate.
4. Click OK. You are returned to the Oracle Wallet Manager main window.

Exporting All Trusted Certificates

Export all of your trusted certificates to another file system location as follows:

1. Click Operations > Export All Trusted Certificates. The Export Trusted Certificate dialog box appears.
2. Enter the file system directory in which to save your trusted certificates, or click Browse to display the directory structure.
3. Enter the name of the file to which you want to save your trusted certificates.
4. Click OK. You are returned to the Oracle Wallet Manager main window

Exporting a Wallet

Export a wallet to text-based PKI formats. Individual components will be formatted according to the following standards:

Component	Encoding Standard
Certificate chains	X509v3
Trusted certificates	X509v3
Private keys	PKCS5

19

Oracle Enterprise Login Assistant

Use the Oracle Enterprise Login Assistant to open and close existing wallets and enable / disable secure SSL based communications.

> **See Also:** Chapter 18, for instructions on managing wallets by using Oracle Wallet Manager.

This chapter covers topics in the following sections:

- About Oracle Enterprise Login Assistant
- Starting Oracle Enterprise Login Assistant
- Enabling Automatic Login
- Disabling Automatic Login
- Changing a Wallet Password

About Oracle Enterprise Login Assistant

Oracle Wallet Manager provides secure management of PKI[1] -based user credentials. Oracle Wallet Manager creates a private and public key pair for a user, and issues a PKCS[2] #10 certificate signing request which can be fulfilled by a certificate authority (CA). After the CA issues an X.509 certificate, the user can load the certificate into his wallet.

Oracle Wallet Manager also manages user trustpoints, the list of root certificates that the user trusts, and is pre-configured with root certificates from PKI vendors such as VeriSign and CyberTrust. Wallets are protected using password-based, strong encryption.

Typically, users do not need to access their wallets once the wallets have been configured. However, they can easily access their wallets using Oracle Enterprise Login Assistant, a very simple-to-use login tool that hides the complexity of a private key and certificate. Once users have securely opened their wallets by using Oracle Enterprise Login Assistant, they can connect to multiple databases over SSL, without providing additional passwords. This provides the benefit of strong authentication as well as single sign-on.

Starting Oracle Enterprise Login Assistant

Refer to your platform-specific documentation for instructions on how to start Oracle Enterprise Login Assistant.

Enabling Automatic Login

The Automatic Login feature of the Oracle Enterprise Login Assistant enables applications running on a server or a client to revalidate themselves to the other end without human intervention. Users can thus obtain single sign-on (SSO), using the credentials contained in their wallets, to authenticate to multiple applications over SSL.

To enable secure SSL based communications using the default wallet:

1. Click AutoLogin > Login from the menu bar.
2. The Login dialog box appears.

[1] Public Key Infrastructure

[2] Public Key Certificate Standard

3. Enter the wallet password.
4. Click OK.
5. The Oracle Enterprise Login Assistant then creates an obfuscated copy of the wallet on the file system.
6. You are returned to the Oracle Enterprise Login Assistant window. A message at the bottom of the window displays the message “Autologin enabled”.

Disabling Automatic Login

Use the Oracle Enterprise Login Assistant to disable single sign-on communications from server side applications to the client. Log out as follows.

1. Click AutoLogin > Logout from the menu bar.
2. A dialog box displays the warning “If you log out, your applications will no longer use the security credentials of your wallet”.
3. Click Yes to continue.
4. You are returned to the Oracle Enterprise Login Assistant window. A message at the bottom of the window displays the message “Autologin not enabled”.

Changing a Wallet Password

Change a wallet password according to company policy or whenever you think that a password has been compromised. Change a wallet password as follows.

1. Click AutoLogin > Change Password from the menu bar.
2. The Change Password dialog box appears.
3. Enter the existing password in the Old Password field.
4. Read the text that describes how to create more secure passwords.
5. Enter the new password in the New password field.
6. Enter the new password again in the Confirm password field.
7. Click OK to continue.

   A dialog box appears and displays the message “Password changed successfully”.

8. Click OK to dismiss this dialog box.

Note: This procedure changes the password only for the wallet that is stored in the default wallet location. It will not change the password for the current open wallet used for SSO communication.

20

Using Oracle Enterprise Security Manager

This chapter describes how an enterprise database administrator uses Oracle Enterprise Security Manager to administer database security in an enterprise domain of Oracle8*i* databases.

This information is organized into the following sections:

Introduction

Oracle Enterprise Security Manager is an administration tool that provides a graphical user interface to manage enterprise users, enterprise domains, databases, and enterprise roles that are held in a directory server.

Table 20–1 explains the terminology used when describing this tool and its features.

Table 20–1 Oracle Enterprise Security Manager Terms and Definitions

Term	Definition
Administrative Context	The location for an Oracle Context. It can be any entry in the directory.
Database Security Administrator	Has create, modify, and read access for enterprise user security. This type of administrator has permissions on all of the domains in the enterprise and is responsible for: ■ Administering the OracleDBSecurityAdmins and OracleDBCreators groups ■ Creating new enterprise domains ■ Moving databases from one domain to another within the enterprise
Database Administrator	Manages a specific database object and its related Net8 objects in the directory.
Database Installation Administrator	Also called a database creator. This type of administrator is in charge of creating new databases, and this includes registering each database in the directory by using the Database Configuration Assistant. This type of administrator has create and modify access to database service objects and attributes. This administrator can also modify the Default Domain.
Enterprise Domain	Directory construct that consists of databases and enterprise roles. A database should only exist in one enterprise domain at any given point in time.
Enterprise Domain Administrator	User(s) authorized to manage a specific enterprise domain including adding new enterprise domain administrators to this list of users.
Enterprise Role	Roles that determine access privileges on databases. Enterprise roles are stored in the directory and contain one or more global roles.
Enterprise User	User that is defined and managed in a directory. Each enterprise user has a unique identity across an enterprise and uses a wallet.

Term	Definition
Global Role	A role that is assigned to a user in a directory, but whose privileges are contained within a single database.

Installing and Configuring Oracle Enterprise Security Manager

The following instructions outline how to install Oracle Management Server and Oracle Enterprise Manager with the Oracle Enterprise Security Manager.

These installation steps are discussed in the following sections:

- Task 1: Install Oracle Enterprise Security Manager
- Task 2: Configure Oracle Enterprise Security Manager
- Task 3: Start Oracle Enterprise Security Manager
- Task 4: Log Into the Directory

Task 1: Install Oracle Enterprise Security Manager

You do this when you install Oracle Enterprise Manager. See the platform-specific installation documentation for Oracle Enterprise Manager.

Task 2: Configure Oracle Enterprise Security Manager

Oracle Enterprise Security Manager must be able to connect to databases published in the directory. Therefore, there should be a TNS alias for each database, and that alias should match the global name of the database and its common name in the directory.

Use the Net8 Configuration Assistant to create a `tnsnames.ora` file in `ORACLE_HOME/network/admin`, and create service names for the databases you want to manage. This is not necessary if all databases to be managed are set up to listen for incoming TCP connections on port 1521 (part of the default setup) and their global database names are exactly *hostname.domain*.

Use the Net8 Configuration Assistant to set up directory access. This creates an `ldap.ora` file on `ORACLE_HOME/network/admin`.

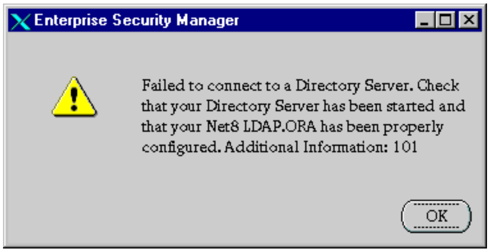
> **Note:** Oracle Enterprise Security Manager allows you to specify the Net8 service to connect to a database published in the directory.

Task 3: Start Oracle Enterprise Security Manager

To start Oracle Enterprise Security Manager, in the command line, enter

```
oemapp esm
```

If the `ldap.ora` file is not configured, you receive the following alert:



Failed to connect to a Directory Server. Check that your Directory Server has been started and that your Net8 LDAP.ORA has been properly configured. Additional Information: 101

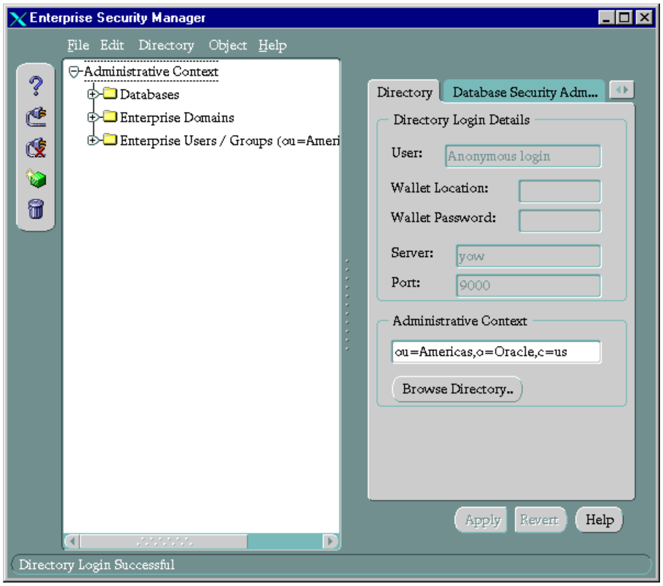
In this case, you can exit Oracle Enterprise Security Manager and run Net8 Configuration Assistant to set up directory access, then restart Oracle Enterprise Security Manager. Alternatively, you can:

- Manually connect to the directory server as described in "Task 4: Log Into the Directory" on page 20-6
- Configure the administrative context

If the ldap.ora file is properly configured, Oracle Enterprise Security Manager starts and automatically connects to the directory server.

- Windows NT: If you are using Microsoft's Active Directory, Oracle Enterprise Security Manager logs in to the directory using native authentication.
- UNIX: Oracle Enterprise Security Manager attempts to connect to the directory server by using SSL. If this fails, Oracle Enterprise Security Manager attempts to connect by using anonymous authentication.

On startup, Oracle Enterprise Security Manager looks something like this:



User: Anonymous login

Wallet Location:

Wallet Password:

Server: yow

Port: 9000

Administrative Context: ou=Americas,o=Oracle,c=us

Directory Login Successful

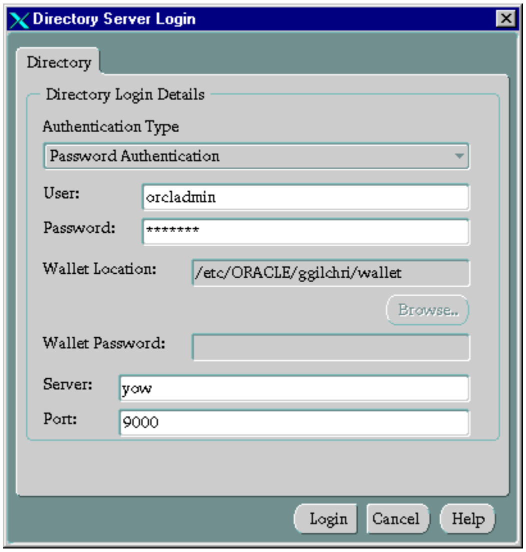
If the result of automatic login is not what you want, log out, then log back in again with the user name you want to use. To do this:

1. From the menu bar, choose Directory > Logout.
2. From the menu bar, choose Directory > Login. This displays the Directory Server Login dialog box.
3. Proceed to "Task 4: Log Into the Directory" for instructions on filling in the fields in this dialog box.

Task 4: Log Into the Directory

To log into the directory:

1. From the Oracle Enterprise Security Manager menu bar, choose Directory > Login.
2. Choose the authentication type. Options are:
   - Password Authentication: Uses simple authentication requiring a user DN and password.
   - SSL Client Authentication: Uses two-way SSL authentication in which both client and server use Oracle wallets containing digital certificates.
   - Native Authentication (Windows NT or Windows 2000 only): Relies on the operating system to determine how you log in.



Authentication Type: Password Authentication

User: orcladmin

Password: *******

Wallet Location: /etc/ORACLE/ggilchri/wallet

Wallet Password:

Server: yow

Port: 9000

3. If you are using SSL, enter the wallet location and the wallet password.
4. Enter the server and port number. If you are using SSL, be sure to enter the directory’s SSL port number.
5. Click OK.

Navigating Oracle Enterprise Security Manager

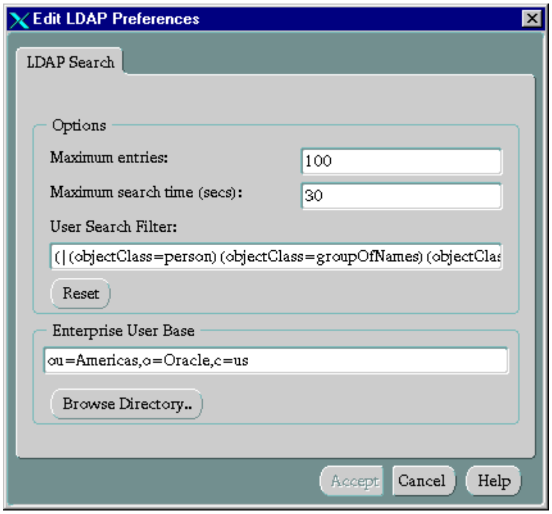
This section introduces some basic features of Oracle Enterprise Security Manager. It does so in the following sections:

- Changing a Search Base
- Browsing the Directory

Changing a Search Base

By default, when Oracle Enterprise Security Manager performs a search, it uses as its search base the administrative context you have already set. If you want to use a search base other than the configured administrative context, do the following:

1. On the menu bar, click Edit > Preferences. The Edit LDAP Preferences dialog box appears.



Edit LDAP Preferences

LDAP Search

Options

Maximum entries: 100

Maximum search time (secs): 30

User Search Filter: (|(objectClass=person)(objectClass=groupOfNames)(objectClas

Enterprise User Base

ou=Americas,o=Oracle,c=us

2. In the Enterprise Users Base field, enter the DN you want to use as the base of the search.

   You can also click Browse Directory to navigate to a directory object to use as the base of the search.

3. Click Accept.

Browsing the Directory

A Browse Directory button appears frequently as you use Oracle Enterprise Security Manager. Whenever you click a Browse Directory button, Oracle Enterprise Security Manager displays a dialog box that allows you to focus your search by specifying a naming context and directory search criteria. In each context, you use this dialog box in the same way.

For example, suppose you want to change the administrative context to `c=acme,c=us`. To do this, you would:

1. Navigate to the Oracle Enterprise Security Manager initial screen (Task 3), and click Browse Directory. The corresponding dialog box appears.
2. In the Naming Context field, you would enter `c=us`.
3. In the Directory Search Attributes field, in the Searchable Attribute Value column, in the object class row, you would enter `organization`. The entries for organizations in the U.S. would appear in the Directory Search Results: Directory Entry field.
4. In the Directory Search Results: Directory Entry field, you would select the directory entry that you want to use as the new administrative context, and click OK. This would return you to the Oracle Enterprise Security Manager initial screen. The administrative context you just specified would appear in the Administrative Context field.

Use the same steps when browsing for directory objects in other contexts, for example, when using the Edit LDAP Preferences dialog box to change the base of a search.

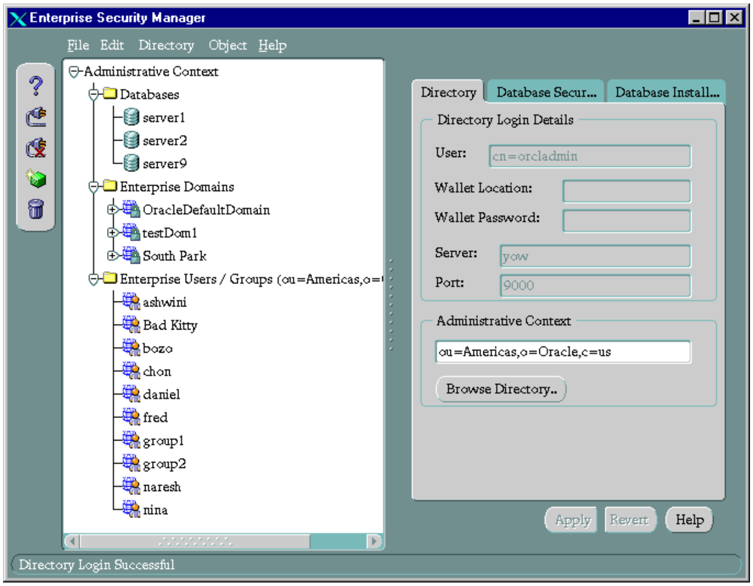
Administering Enterprise Databases, Domains, and Users

The following instructions assume you are running Oracle Enterprise Manager and have invoked the Oracle Enterprise Security Manager.

> **See Also:** "Task 3: Start Oracle Enterprise Security Manager" on page 20-4 for instructions on how to start the Oracle Enterprise Security Manager and on how to connect to a database.

Managing enterprise users involves working in the three top level nodes in the Oracle Enterprise Security Manager navigator pane. These three nodes are discussed in the following sections:

- Administering Databases
- Administering Enterprise Domains
- Administering Enterprise Users



- Administrative Context
  - Databases
    - server1
    - server2
    - server9
  - Enterprise Domains
    - OracleDefaultDomain
    - testDom1
    - South Park
  - Enterprise Users / Groups (ou=Americas,o=
    - ashwini
    - Bad Kitty
    - bozo
    - chon
    - daniel
    - fred
    - group1
    - group2
    - naresh
    - nina

Directory Login Details

User: cn=orcladmin

Wallet Location:

Wallet Password:

Server: yow

Port: 9000

Administrative Context

ou=Americas,o=Oracle,c=us

Directory Login Successful

Administering Databases

This section tells you how to manage user/schema separation for a database.

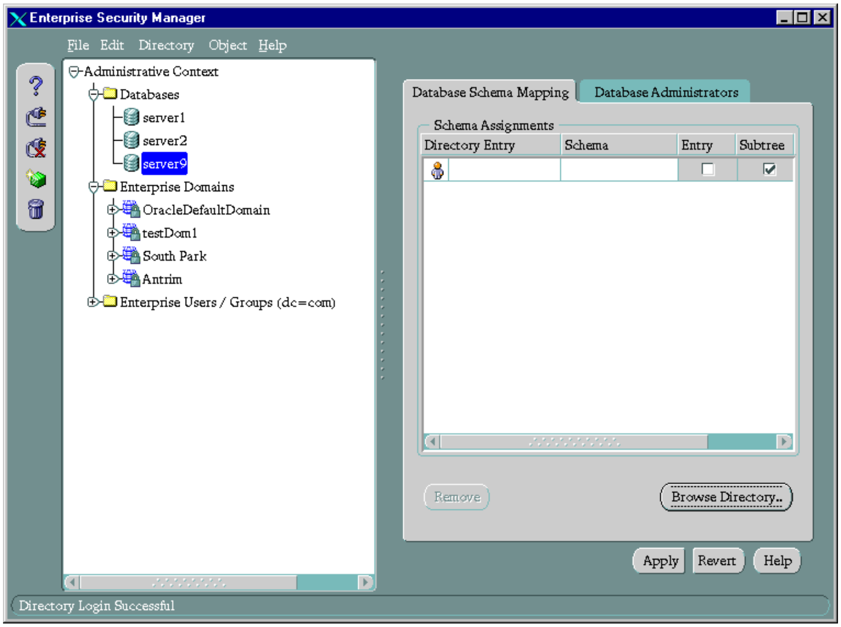
See Also:

- "User/Schema Separation" on page 17-12 for a conceptual overview of this feature
- "Managing Database Administrators" on page 20-29

Managing User/Schema Separation

To map an enterprise user to a database schema:

1. In the navigator pane, expand Administrative Context > Database.
2. Select the server on which you want to support user/schema separation. The corresponding tab pages appear in the right pane.

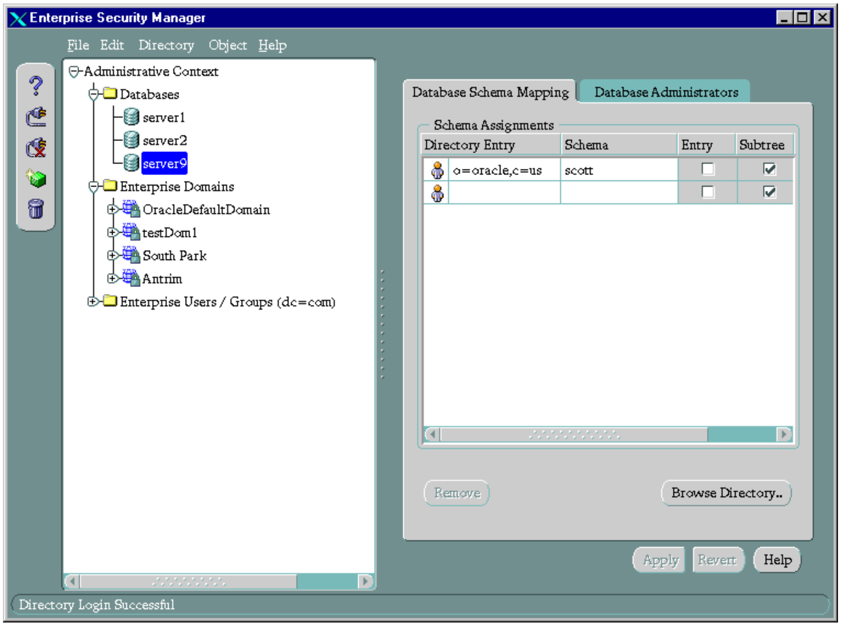


Directory Login Successful

3. In the Database Schema Mapping tab page, in the Schema Assignments window, in the Directory Entry column, enter either the full or partial DN of the entry that you want to map to a shared schema. You can also click the Browse Directory button to navigate to that DN.

4. In the same row, in the Schema column, enter the name of an existing schema on that database.

5. If this is a full DN, in the Entry column, select the check box. If this is a partial DN, in the Subtree column, select the check box.



Directory Entry	Schema	Entry	Subtree
o=oracle,c=us	scott	☐	☑
		☐	☑

Directory Login Successful

6. Click Apply. The database object is updated in the directory, and an empty row is added in the Schema Assignments window should you want to add more mappings.

Administering Enterprise Domains

There is initially one enterprise domain listed under the Enterprise Domains node in the Oracle Enterprise Security Manager navigator: Oracle Default Domain. Each enterprise domain you define in the LDAP directory is added under the Enterprise Domains node.

This section discusses topics in the following subsections:

- Managing User/Schema Separation
- Creating an Enterprise Domain
- Adding a Database to an Enterprise Domain
- Creating an Enterprise Role within an Enterprise Domain
- Assigning Enterprise Users to an Enterprise Role
- Removing a Database from an Enterprise Domain
- Deleting an Enterprise Domain

> **See Also:** "Managing Enterprise Domain Administrators" on page 20-30.

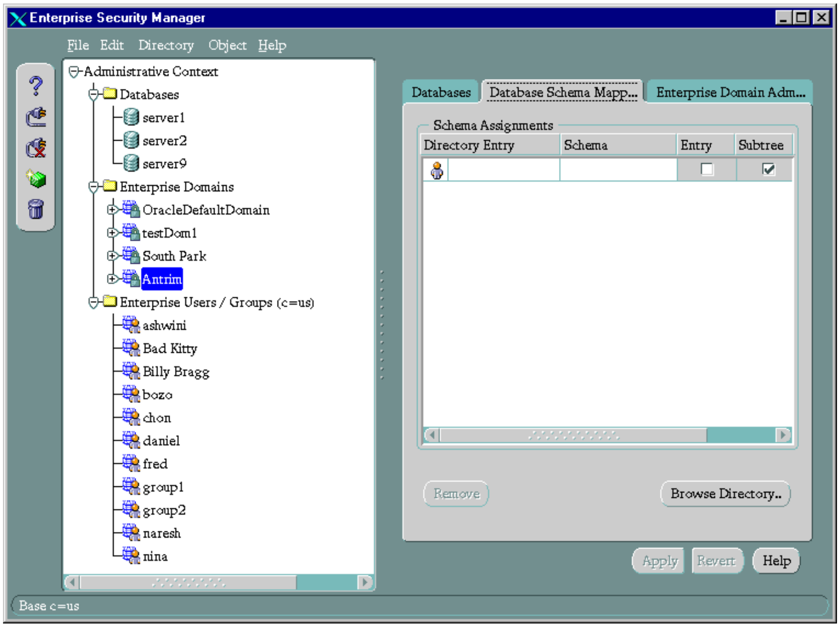
Managing User/Schema Separation

The section "Administering Databases" on page 20-10 discussed how you manage user/schema separation on an individual database. This section tells you how to manage user/schema separation on all the databases in a given domain.

To map an enterprise user to a database schema:

1. In the navigator pane, expand Administrative Context > Enterprise Domains.

2. Select the enterprise domain on which you want to support user/schema separation. The corresponding tab pages appear in the right pane.

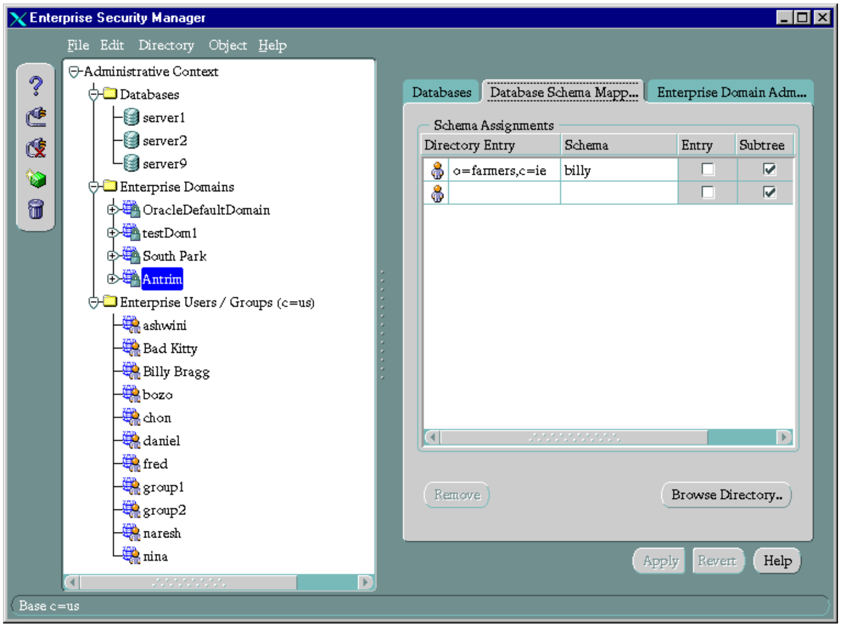


3. Select the Database Schema Mapping tab.

4. In the Schema Assignments window, in the Directory Entry column, enter either the full or partial DN that you want to map to a shared schema. You can also click the Browse Directory button to navigate to the DN.

5. In the same row, in the Schema column, enter the name of an existing schema supported by all the databases in the domain.

6. If this is a full DN, in the Entry column, select the check box. If this is a partial DN, in the Subtree column, select the check box.



Directory Entry	Schema	Entry	Subtree
o=farmers,c=ie	billy	☐	☑
		☐	☑

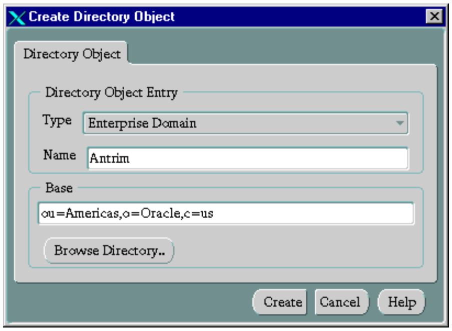
7. Click Apply. The database object is updated in the directory, and an empty row is added in the Schema Assignments window should you want to add more mappings.

Creating an Enterprise Domain

An enterprise domain contains databases and enterprise roles. You create a new enterprise domain by giving a name to the new enterprise domain and defining where the enterprise domain is to be located in the directory.

To create an enterprise domain:

1. Click Object > Create on the menu bar. The Create Directory Object dialog appears.



Create Directory Object

Directory Object

Directory Object Entry

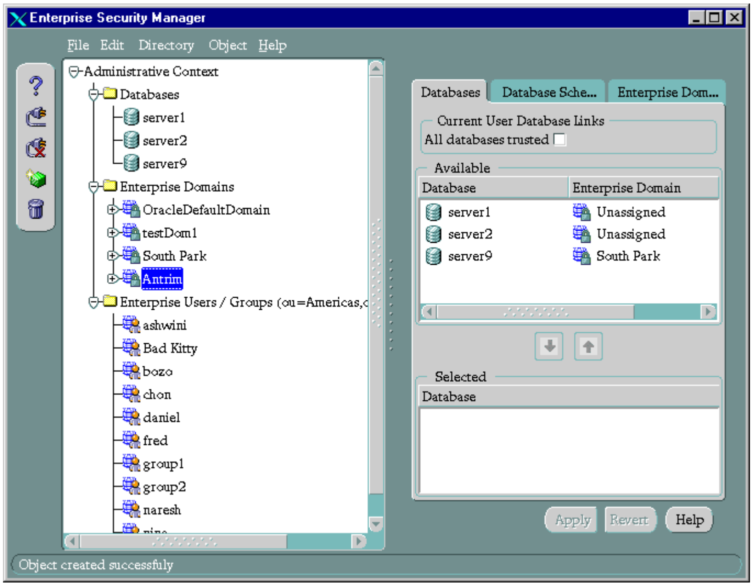
Type: Enterprise Domain

Name: Antrim

Base: ou=Americas,o=Oracle,c=us

2. In the Type menu, select Enterprise Domain.
3. In the Name field, enter the name of the enterprise domain you want to create.
4. In the Base field, Oracle Enterprise Security Manager fills in the name of the administrative context. If you want to use a different administrative context, you may change the values in this field. However, be careful to enter the name of a valid administrative context, that is, one that contains and Oracle Context.
5. Click Create.The enterprise domain you just created appears at the bottom of the Enterprise Domains node.

6. In the navigator pane, click the name of the enterprise domain you just created. The corresponding group of tab pages appear in the right pane.



All databases trusted □

Database	Enterprise Domain
server1	Unassigned
server2	Unassigned
server9	South Park

Object created successfuly

7. Select the All Databases trusted check box, if desired. This allows databases within the enterprise domain to have current user database links between each other.

Note: Individual database administrators still have the capability to configure their databases to not trust other databases.

You have now created an enterprise domain and can proceed to add databases to the enterprise domain you just created.

Adding a Database to an Enterprise Domain

At the completion of database installation, you directed Oracle Database Configuration Assistant to publish the database in the directory. Once you have created an enterprise domain, you can view a list of all databases registered in the directory, select a database from that list, and assign it to the enterprise domain you created.

A database should exist in only one enterprise domain at a time. Therefore, you should assign a database to an enterprise domain only if the database has a value of "Unassigned" on the Databases property page.

To assign a database to the enterprise domain:

1. In the navigator pane, expand Administrative Context > Enterprise Domains.
2. In the navigator pane, select the enterprise domain to which you want to add a database.
3. In the right pane, in the Available window, select a database name.
4. Click the down arrow to move the selected database to the Selected list.
5. Click Apply.

You have added a database to an enterprise domain and can proceed to create an enterprise role in that enterprise domain.

> **See Also:** "Use Oracle Database Configuration Assistant to Register the Database in the Directory" on page 17-23.

Creating an Enterprise Role within an Enterprise Domain

Once you have created an enterprise domain and added a database to it, you can create an enterprise role within the enterprise domain.

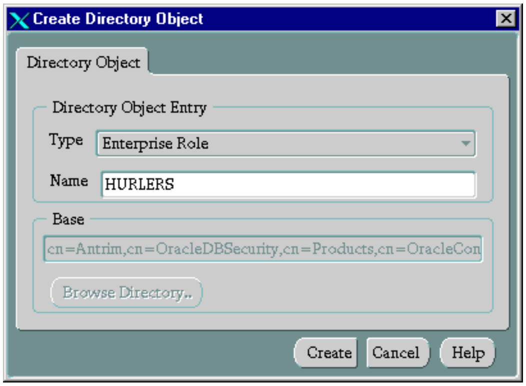
An enterprise role is a set of global roles that operate on multiple databases within an enterprise domain. An enterprise role is assigned to one or more enterprise users. The Enterprise Database Administrator uses these enterprise roles to assign sets of global roles on multiple databases to a selected user all at once.

You cannot create two enterprise roles with the same name within a single enterprise domain.You can, however, create enterprise roles with the same name in separate enterprise domains. Enterprise roles with the same name that exist in separate enterprise domains have no implied relationships.

> **Note:** The database obtains a user’s global roles when the user logs in. If you change a user’s global roles, those changes do not take effect until the next time the user logs in.

To create an enterprise role in an enterprise domain:

1. In the navigator pane, expand Administrative Context > Enterprise Domains and select the enterprise domain name. The corresponding group of tab pages appear in the right pane.
2. On the menu bar, click Object > Create. The Create Directory Objects dialog box appears.



Create Directory Object

Directory Object

Directory Object Entry

Type: Enterprise Role

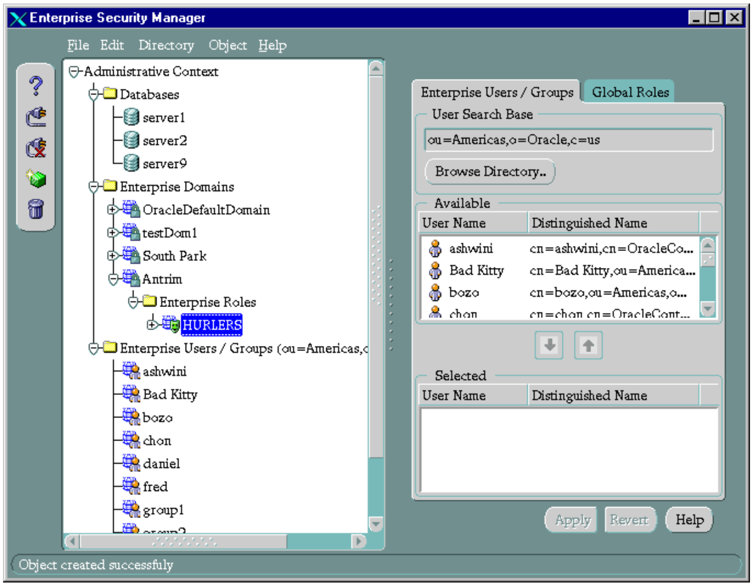
Name: HURLERS

Base: cn=Antrim,cn=OracleDBSecurity,cn=Products,cn=OracleCon

3. In the Type menu, select Enterprise Role.
4. In the Name field, enter the name of the enterprise role you want to create.

   Note that the directory base chosen for the new enterprise role derives from the currently selected enterprise domain. You cannot edit this value.
5. Click Create.
6. In the navigator pane, expand Enterprise Domains > *enterprise_domain_name* > Enterprise Roles.

7. In the navigator pane, in the Enterprise Domains subtree, select the name of the enterprise role you just created. The corresponding group of tab pages appear in the right pane.



User Search Base: ou=Americas,o=Oracle,c=us

Available

User Name	Distinguished Name
ashwini	cn=ashwini,cn=OracleCo...
Bad Kitty	cn=Bad Kitty,ou=America...
bozo	cn=bozo,ou=Americas,o...
chon	cn=chon,cn=OracleCont...

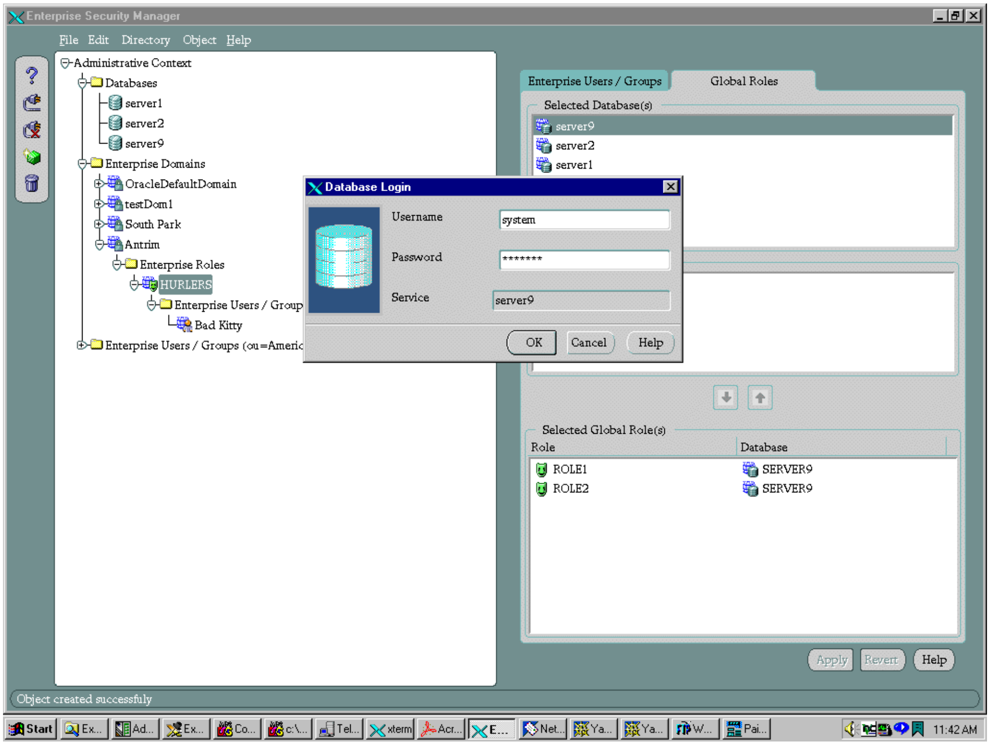
Selected

User Name	Distinguished Name

Object created successfuly

8. Select the Global Roles tab.

9. Select a database. The Database Login dialog box appears.



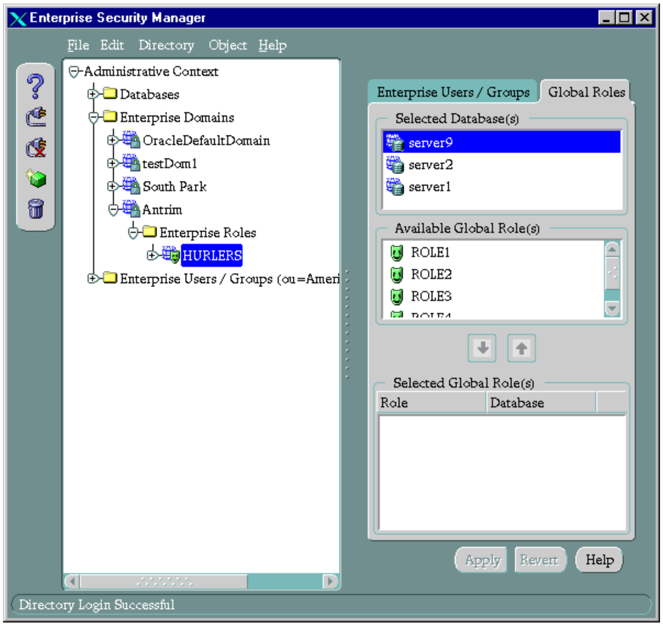
Username: system

Password: *******

Service: server9

Object created successfuly

10. Supply the correct information about the selected database. Click OK. The selected database roles appear in the Available Global Role(s) area.



Directory Login Successful

If no database service has been configured:

a. In the Global Roles tab page, in the Selected Databases field, right-click the database name.

b. Choose Reconnect.

c. Specify a new service name.

Note: Although Oracle Enterprise Security Manager provides this database configuration convenience, be sure to properly configure the Oracle Enterprise Security Manager Net8 client environment to support connectivity to databases as they are named in the directory server.

---

11. In the Available Role(s) field, select an available role.

12. Click the down arrow button to move the role into the Selected Global Role(s) area.

13. Repeat steps 9 through 12 for each database from which you want to select one or more roles.
14. Click Apply.

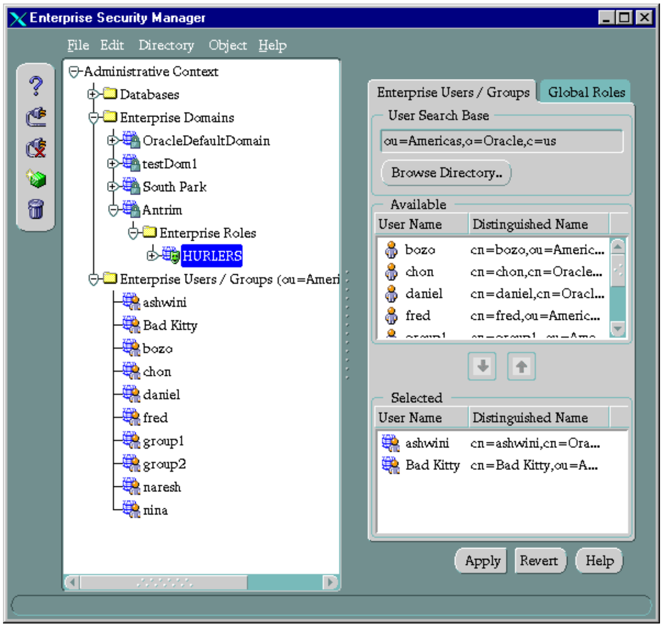
You have created an enterprise role within an enterprise domain of databases and can assign this enterprise role to any enterprise user.

Assigning Enterprise Users to an Enterprise Role

To assign an enterprise user to an enterprise role:

1. In the navigator pane, expand Administrative Context > Enterprise Domains > *enterprise_domain_name* > Enterprise Roles.
2. In the navigator pane, select the enterprise role. The corresponding group of tabs pages appears in the right pane.
3. In the right pane, select the Enterprise / Users Groups tab.
4. In the Available window, select the enterprise users you want to assign to the role.

5. Click the down arrow button. The enterprise users appear in the Selected window.



User Search Base: ou=Americas,o=Oracle,c=us

Available

User Name	Distinguished Name
bozo	cn=bozo,ou=Americ...
chon	cn=chon,cn=Oracle...
daniel	cn=daniel,cn=Oracl...
fred	cn=fred,ou=Americ...

Selected

User Name	Distinguished Name
ashwini	cn=ashwini,cn=Ora...
Bad Kitty	cn=Bad Kitty,ou=A...

6. Click Apply. The enterprise users appear under the Enterprise Role node in the navigator pane.

Removing a Database from an Enterprise Domain

1. In the navigator pane, expand Administrative Context > Enterprise Domains.
2. In the navigator pane, select the name of the enterprise domain from which you want to remove a database.
3. In the Databases tab page, in the Selected window, select the database you want to remove from the enterprise domain.
4. Click the up button to move the database from the Selected window to the Available window.
5. Click Apply.

Deleting an Enterprise Domain

You must first delete all enterprise roles from the enterprise domain before you can delete the selected enterprise domain. Otherwise a message window will display informing you that you need to delete one or more enterprise roles.

To delete an Enterprise Domain:

1. In the navigator pane, expand Administrative Contexts > Enterprise Domains.
2. In the navigator pane, select the name of the enterprise domain you want to delete.
3. Click the Delete Object button to the left of the navigator pane.
   A window asks you to confirm the deletion.
4. Click Yes.
5. The selected enterprise domain is removed from the enterprise domains tree.

Administering Enterprise Users

This section discusses the following tasks:

- Creating a New Enterprise User
- Granting an Enterprise Role to an Enterprise User
- Deleting an Enterprise User

Creating a New Enterprise User

Oracle Enterprise Security Manager allows you to create new enterprise users if the users do not already exist in the directory server. To do this.

1. From the menu bar, select Object > Create. The Create Directory Object dialog box appears.
2. In the Type menu, select Enterprise User.
3. In the Name field, enter the name of the enterprise user you want to create.
4. In the Base field, accept the default, or enter a new search base as described in "Browsing the Directory" on page 20-8.
5. Click Create. The enterprise user you created appears in the navigator pane under the Enterprise Users/Groups node. When you select the new enterprise user, the corresponding tab page appears in the right pane.

Note: In the above procedure, the directory user entry that Oracle Enterprise Security Manager creates is associated with only the `top` and `person` object classes. If you want to associate that user entry with other object classes, you must do so in an separate procedure.

Granting an Enterprise Role to an Enterprise User

Once you have created an enterprise user, you can assign enterprise roles to that user.

You can grant many enterprise roles to enterprise users, and these roles can exist in different enterprise domains. You can grant these roles in two ways:

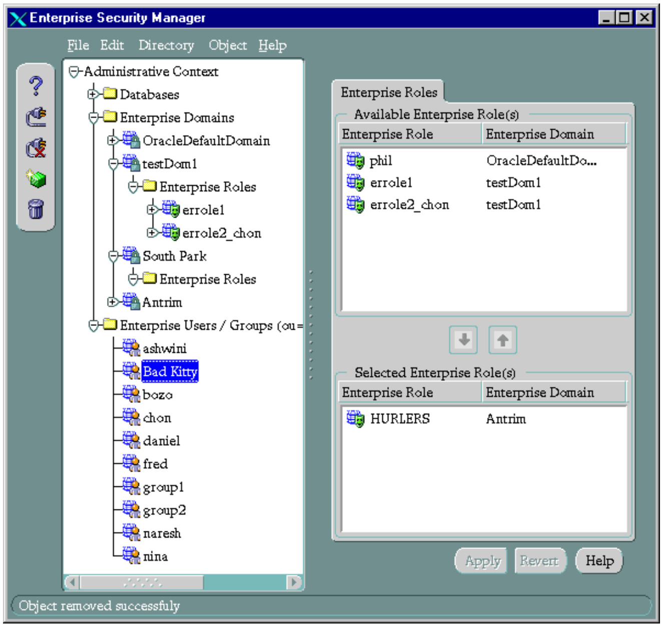
- Assign enterprise users to each enterprise role, as described in "Assigning Enterprise Users to an Enterprise Role" on page 20-22
- Grant one or more enterprise roles to each enterprise user, as described in this section

When a database needs to authorize a global user, it searches the directory for those enterprise role(s) within its enterprise domain that are granted to that user.

To grant an enterprise role to an enterprise user:

1. In the navigator pane, expand Administrative Context > Enterprise Users.

2. Select the name of an enterprise user. The corresponding group of tab pages appears in the right pane.



Available Enterprise Role(s)

Enterprise Role	Enterprise Domain
phil	OracleDefaultDo...
errole1	testDom1
errole2_chon	testDom1

Selected Enterprise Role(s)

Enterprise Role	Enterprise Domain
HURLERS	Antrim

Object removed successfuly

3. In the Available Enterprise Role(s) window, select an enterprise role you want to grant to the enterprise user.
4. Click the down arrow button.
5. The selected role is moved from the Available Role(s) list to the Selected Role(s) list.
6. Click Apply.

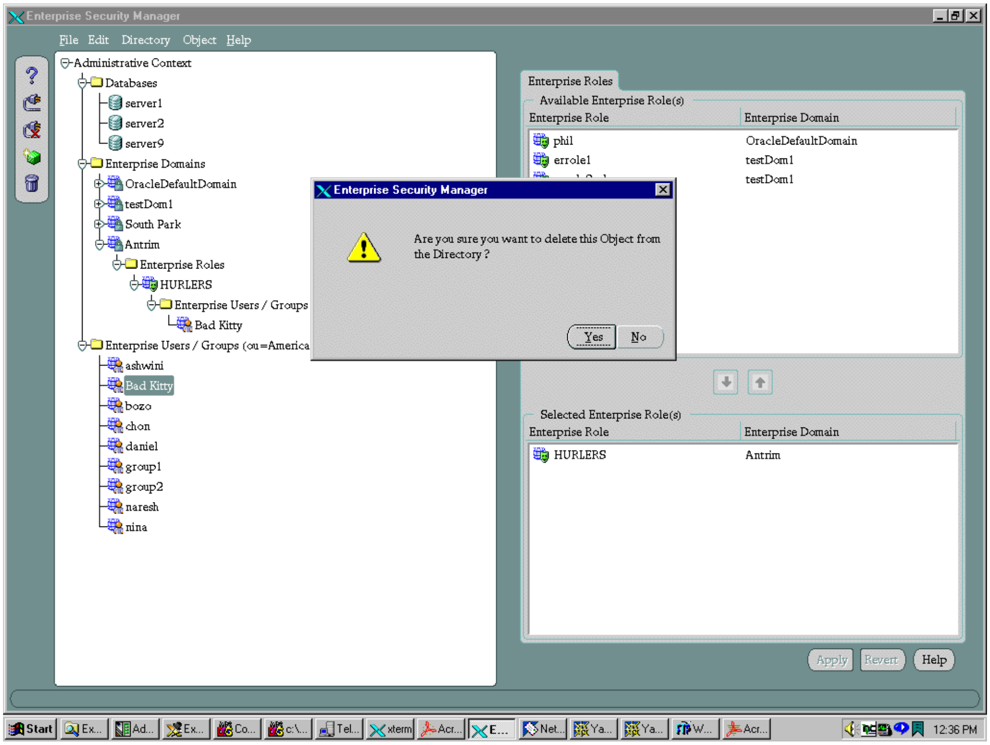
Deleting an Enterprise User

Note: You can delete an enterprise user only if that user has no enterprise roles.

---

1. Expand Administrative Context > Enterprise Users / Groups.
2. Select the enterprise user you want to delete.

3. On the menu bar, click Object > Delete. An alert asks you to confirm the deletion.



Enterprise Security Manager

Are you sure you want to delete this Object from the Directory ?

Enterprise Role	Enterprise Domain
phil	OracleDefaultDomain
errole1	testDom1
	testDom1

Enterprise Role	Enterprise Domain
HURLERS	Antrim

4. Click Yes. The enterprise user is deleted from the tree in the navigator pane.

Managing Security Administrators

See Also: "Oracle Context" on page 17-6.

Use Oracle Enterprise Security Manager to define various administrators as described in the following sections:

- Managing Database Security Administrators
- Managing Database Installation Administrators
- Managing Database Administrators
- Managing Enterprise Domain Administrators

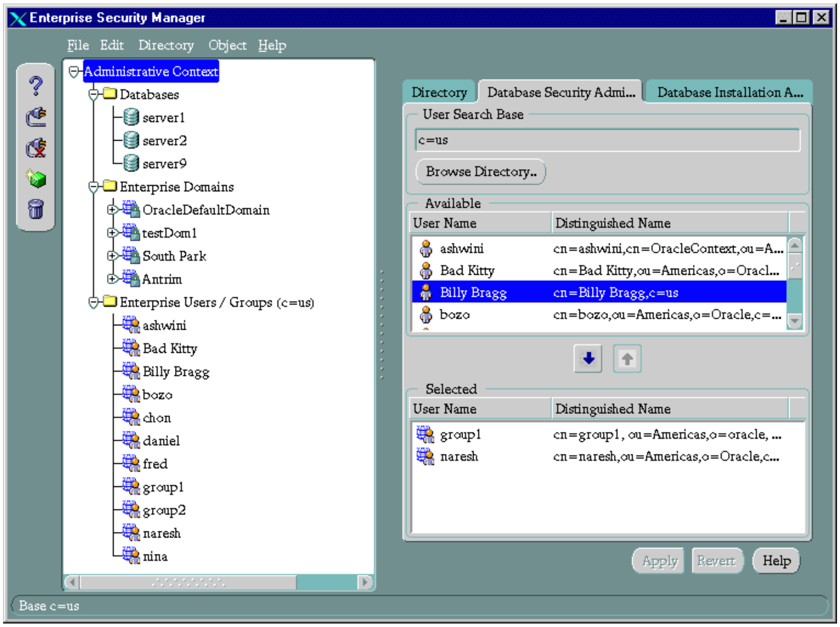
Managing Database Security Administrators

To manage database security administrators, you must be a member of the Database Security Administrators group.

To specify a user to be a database security administrator:

1. In the navigator pane, select Administrative Contexts.
2. In the right pane, select the Database Security Administrators tab.

   Enterprise user names appear in the Available field.



User Search Base: c=us

Available

User Name	Distinguished Name
ashwini	cn=ashwini,cn=OracleContext,ou=A...
Bad Kitty	cn=Bad Kitty,ou=Americas,o=Oracl...
Billy Bragg	cn=Billy Bragg,c=us
bozo	cn=bozo,ou=Americas,o=Oracle,c=...

Selected

User Name	Distinguished Name
group1	cn=group1, ou=Americas,o=oracle, ...
naresh	cn=naresh,ou=Americas,o=Oracle,c...

Base c=us

3. Select the enterprise user you want to specify as an administrator.
4. Click the down arrow to move the user to the Selected window.
5. Click Apply.

Managing Database Installation Administrators

To manage database installation administrators, you must be a member of the Database Security Administrators group.

To specify a user to be a database installation administrator:

1. In the navigator pane, select Administrative Contexts.
2. In the right pane, select the Database Installation Administrators tab.

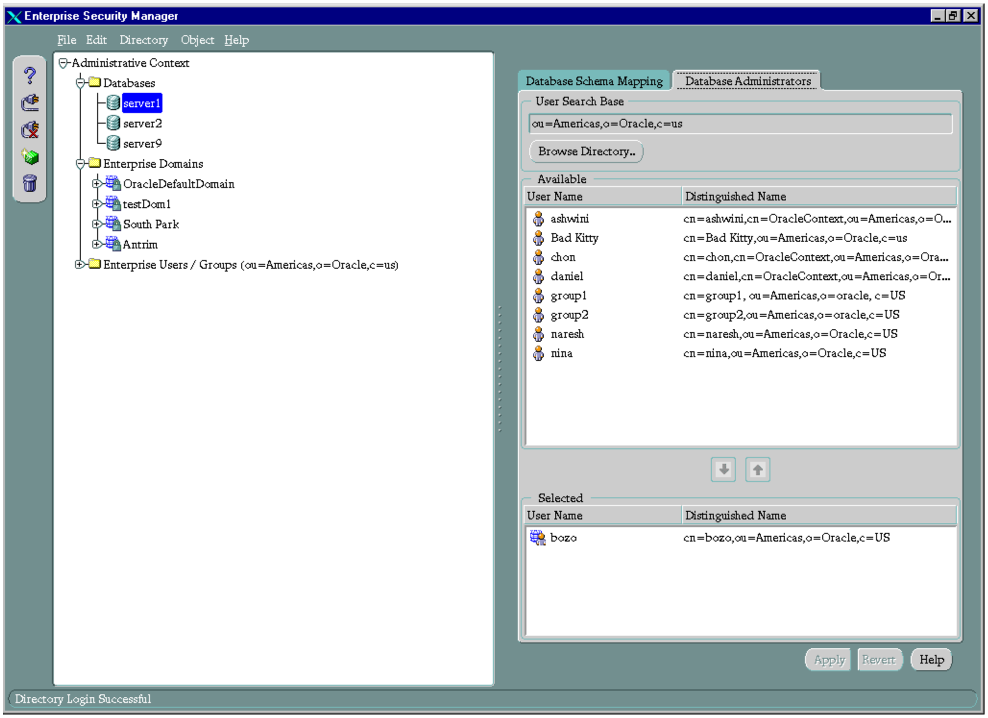
   Enterprise user names appear in the Available field.
3. Select the enterprise user you want to specify as an administrator.
4. Click the down arrow to move the user to the Selected window.
5. Click Apply.

Managing Database Administrators

To manage database administrators, you must be either a member of the Database Security Administrators group or a database administrator for this particular database.

1. In the navigator pane, expand Administrative Context > Database.
2. Select the database for which you want to assign administrators. The corresponding group of tab pages appears in the right pane.

3. Select the Database Administrators tab. The Available window displays user names of enterprise users available in the current user search base.



User Search Base

ou=Americas,o=Oracle,c=us

Available

User Name	Distinguished Name
ashwini	cn=ashwini,cn=OracleContext,ou=Americas,o=O...
Bad Kitty	cn=Bad Kitty,ou=Americas,o=Oracle,c=us
chon	cn=chon,cn=OracleContext,ou=Americas,o=Ora...
daniel	cn=daniel,cn=OracleContext,ou=Americas,o=Or...
group1	cn=group1, ou=Americas,o=oracle, c=US
group2	cn=group2,ou=Americas,o=oracle,c=US
naresh	cn=naresh,ou=Americas,o=Oracle,c=US
nina	cn=nina,ou=Americas,o=Oracle,c=US

Selected

User Name	Distinguished Name
bozo	cn=bozo,ou=Americas,o=Oracle,c=US

Directory Login Successful

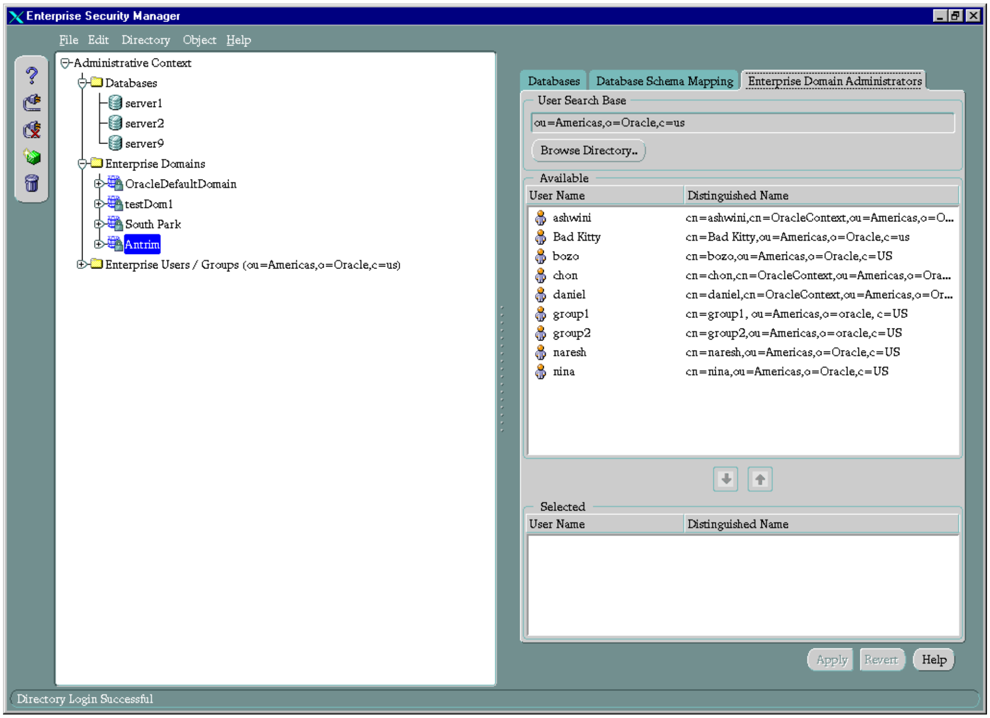
4. Select the enterprise user you want to specify as an administrator.
5. Click the down arrow to move the user to the Selected window.
6. Click Apply.

Managing Enterprise Domain Administrators

To manage enterprise domain administrators, you must be either a member of the Database Security Administrators group of a domain administrator for this particular domain.

1. In the navigator pane, expand Administrative Context > Enterprise Domains.
2. Select the enterprise domain for which you want to assign administrators. The corresponding group of tab pages appears in the right pane.

3. Select the Enterprise Domain Administrators tab. The Available window displays user names of enterprise users available in the current user search base.



User Search Base: ou=Americas,o=Oracle,c=us

Available

User Name	Distinguished Name
ashwini	cn=ashwini,cn=OracleContext,ou=Americas,o=O...
Bad Kitty	cn=Bad Kitty,ou=Americas,o=Oracle,c=us
bozo	cn=bozo,ou=Americas,o=Oracle,c=US
chon	cn=chon,cn=OracleContext,ou=Americas,o=Ora...
daniel	cn=daniel,cn=OracleContext,ou=Americas,o=Or...
group1	cn=group1, ou=Americas,o=oracle, c=US
group2	cn=group2,ou=Americas,o=oracle,c=US
naresh	cn=naresh,ou=Americas,o=Oracle,c=US
nina	cn=nina,ou=Americas,o=Oracle,c=US

Selected

User Name	Distinguished Name

Directory Login Successful

4. Select the enterprise user you want to specify as an administrator.

5. Click the down arrow to move the user to the Selected window.

6. Click Apply.

Part IV

Appendixes

The following appendixes are provided for your reference.

A

Data Encryption and Integrity Parameters

This appendix lists describes encryption and data integrity parameters supported by Oracle Advanced Security. It also includes an example of a `sqlnet.ora` file generated after you perform the network configuration described in Chapter 2 and Chapter 10.

This appendix covers 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)
```

Oracle Advanced Security Integrity

```
#ASO Checksum
sqlnet.crypto_seed = "-kdje83kkep39487dvmlqEPTbxxe70273"
sqlnet.crypto_checksum_server=requested
sqlnet.crypto_checksum_client=requested
sqlnet.crypto_checksum_types_server = (MD5)
sqlnet.crypto_checksum_types_client = (MD5)
```

SSL

```
#SSL
oss.source.my_wallet = (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
```

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
```

Identix

```
#Identix
sqlnet.authentication_services = (beq, identix)
sqlnet.identix_fingerprint_database = identix_scanner
sqlnet.identix_fingerprint_database_user = ofm_client
sqlnet.identix_fingerprint_database_password = ofm_client
sqlnet.identix_fingerprint_method = oracle
```

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/
```

SecurID

```
#SecurID
sqlnet.authentication_services = (beq, securid )
```

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.

If no encryption or data integrity algorithm is specified on the Server Encryption, Client Encryption, Server Checksum, or Client Checksum pages, the server side of the connection uses the first algorithm in its own list of installed algorithms that also appears in the client's list of installed algorithms.

Encryption and data integrity function independently of each other: encryption can be activated while data integrity is off, and data integrity can be activated while encryption is off.

Data Encryption and Integrity Parameters

There are nine parameters to enable data encryption and integrity. The parameters are described in the following sections.

- Server Encryption Level Setting
- Client Encryption Level Setting
- Server Encryption Selected List
- Client Encryption Selected List
- Server Integrity Level Setting
- Client Integrity Level Setting
- Client Integrity Selected List
- Client Integrity Selected List

- Client Profile Encryption

More Information: See Chapter 2, "Configuring Data Encryption and Integrity."

Server Encryption Level Setting

Table A–1 describes server encryption level settings.

Table A–1 Server Encryption Level Setting

Purpose:	This parameter specifies the desired behavior when a client or a server acting as a client is connecting 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`*
Possible values:	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default value:	ACCEPTED

Client Encryption Level Setting

Table A–2 describes client encryption level settings.

Table A–2 Client Encryption Level Setting

Purpose:	This parameter specifies the desired behavior when this client (or this server acting as a client) is connecting 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`*
Possible values:	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default value:	ACCEPTED

Server Encryption Selected List

Table A–3 describes the encryption selected list.

Table A–3 Server Encryption Selected List

Purpose:	This parameter specifies a list of encryption algorithms this server is allowed to use when acting as a server in the order of desired use. Enter the most desired algorithm first. 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 client algorithm types available 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* [,*valid_encryption_algorithm*])
Possible values:	RC4_40: This is RSA RC4 (40-bit key size) for Domestic and International RC4_56: This is RSA RC4 (56-bit key size) for Domestic and International RC4_128: This is RSA RC4 (128-bit key size) for Domestic only DES: This is Standard DES (56-bit key size) for Domestic and International DES40: This is DES40 (40-bit key size) for Domestic and International
Default value:	All installed algorithms are used in a negotiation if no algorithms are defined in the `sqlnet.ora` file.
Usage Notes:	**Domestic version**: If you are using the Domestic version, all five algorithms are installed: RC4_40, RC4_56, RC4_128, DES, and DES40. If no algorithms are specified, the installed algorithms are used in that order to negotiate a mutually acceptable algorithm with the other end of the connection. **Export version**: If you are using the Export version, the following algorithms are installed: RC4_40, RS4_56, DES40, and DES. If no algorithms are specified, the installed algorithms are used in that order to negotiate a mutually acceptable algorithm. You can specify multiple encryption algorithms, that is, 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 Encryption Selected List

Table A–4 describes the encryption selected list.

Table A–4 Client Encryption Selected List

Purpose:	This parameter specifies a list of encryption algorithms this client (or this server acting as a client) is allowed to use when connecting to a server. This list is used to negotiate a mutually acceptable algorithm with the other end of the connection. The parameters can be listed in any order. 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]`*`)`
Possible values:	RC4_40: This is RSA RC4 (40-bit key size) for Domestic and International RC4_56: This is RSA RC4 (56-bit key size) for Domestic and International RC4_128: This is RSA RC4 (128-bit key size) for Domestic only DES: This is Standard DES (56-bit key size) for Domestic and International DES40: This is DES40 (40-bit key size) for Domestic and International
Default value:	All installed algorithms are used if no algorithms are defined in the `sqlnet.ora` file.

Usage Notes:	**Domestic version**: If you are using the Domestic version, all five algorithms are installed: RC4_40, RC4_56, RC4_128, DES, and DES40. If no algorithms are defined in the `sqlnet.ora` file, the installed algorithms are used in that order to negotiate a mutually acceptable algorithm with the other end of the connection.
	Export version: If you are using the Export version, the RC4_40, RC4_56, DES40, and DES algorithms are installed. If no algorithms are defined in the `sqlnet.ora` file, the installed algorithms are used in that order to negotiate a mutually acceptable algorithm.
	You can specify multiple encryption algorithms, that is, 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`*`)`

Server Integrity Level Setting

Table A–5 describes server integrity level settings.

Table A–5 Server Integrity Level Setting

Purpose:	This parameter specifies the desired data integrity behavior when a client (or another server acting as a client) is connecting to this server. The resulting behavior partially depends on the SQLNET.CRYPTO_ CHECKSUM_CLIENT setting at the other end.
Syntax:	`SQLNET.CRYPTO_CHECKSUM_SERVER = valid_value`
Possible values:	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default value:	ACCEPTED

Client Integrity Level Setting

Table A–6 describes client integrity level settings.

Table A–6 Client Integrity Level Setting

Purpose:	This parameter specifies the desired data integrity behavior when this client (or this server acting as a client) is connecting to a server. The resulting behavior partially depends on the SQLNET.CRYPTO_ CHECKSUM_SERVER setting at the other end of the connection.
Syntax:	`SQLNET.CRYPTO_CHECKSUM_CLIENT =` *`valid_value`*
Possible values:	ACCEPTED, REJECTED, REQUESTED, REQUIRED
Default value:	ACCEPTED

Server Integrity Selected List

Table A–7 describes the server integrity selected list.

Table A–7 Server Integrity Selected List

Purpose:	This parameter specifies a list of the data integrity algorithms this server is allowed to use, in order of desired use with the most desired algorithm first, when acting as a server to a client or another server. This list is used to negotiate a mutually acceptable algorithm with the remote end. Each algorithm is checked against the list of client algorithm types available until a match is found. The first algorithm match is the one that is used. 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 = (`*`crypto_checksum_algorithm`*`)`
Possible values:	Currently, the only supported crypto-checksum algorithm choice is RSA Data Security's MD5 algorithm.
Default value:	MD5 (currently the only valid value)

Client Integrity Selected List

Table A–8 describes the client integrity selected list.

Table A–8 Client Integrity Selected List

Purpose:	This parameter specifies a list of data integrity algorithms this client (or this server acting as a client) is allowed to use when connecting to a server. This list is used to negotiate a mutually acceptable algorithm with the remote end. The order in which the algorithms are listed is not important. 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 = (`*`crypto_checksum_algorithm`*`)`
Possible values:	Currently, the only supported crypto-checksum algorithm choice is RSA Data Security's MD5 algorithm.
Default value:	MD5 (currently the only valid value)

Client Profile Encryption

```
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 turned on.

B

Authentication Parameters

This appendix shows some sample configuration files with the necessary profile (`sqlnet.ora`) and database initialization file authentication parameters when using the CyberSafe, Kerberos, SecurID, RADIUS, or SSL authentication. It includes the following sections:

- Parameters for Clients and Servers using CyberSafe Authentication
- Parameters for Clients and Servers using Identix Authentication
- Parameters for Clients and Servers using Kerberos Authentication
- Parameters for Clients and Servers using SecurID 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

initialization parameter file	REMOTE_OS_AUTHENT=FALSE
	OS_AUTHENT_PREFIX=""

Parameters for Clients and Servers using Identix Authentication

The following sections describe the parameters for Identix authentication

- sqlnet.ora File Parameters
- Recommended Minimum Sets of Identix Parameters

sqlnet.ora File Parameters

SQLNET.IDENTIX_USE_MD5HASH

Table B–2 SQLNET.IDENTIX_USE_MD5HASH

Description	The server uses MD5 hashing to validate the authentication decision made on the client PC: values are YES and NO.
Default	YES

SQLNET.IDENTIX_KEY_INDEX

Table B–3 SQLNET.IDENTIX_KEY_INDEX

Description	The Identix key index the client uses when it generates its MD5 checksum: 0 <= value <= 256.
Default	0

SQLNET.IDENTIX_VERIFICATION_THRESHOLD

Table B–4 SQLNET.IDENTIX_VERIFICATION_THRESHOLD

Description	This parameter specifies the verification threshold the server expects its Identix clients to use during fingerprint verification: 0 <= value <= 256.
Default	0

SQLNET.IDENTIX_FINGERPRINT_METHOD

Table B–5 SQLNET.IDENTIX_FINGERPRINT_METHOD

Description	This parameter specifies the storage method used for storing fingerprint template files: format = [file/oracle]
Default	None

SQLNET.IDENTIX_DATABASE_DIRECTORY

Table B–6 SQLNET.IDENTIX_DATABASE_DIRECTORY

Description	This file method specifies the file location in which the fingerprint templates are stored: format = <path-to-file>.
Default	None

SQLNET.IDENTIX_FINGERPRINT_DATABASE

Table B–7 SQLNET.IDENTIX_FINGERPRINT_DATABASE

Description	This paramter specifies the database SQL*NET alias for the Oracle fingerprint storage method: format = <db-alias>.
Default	None

SQLNET.IDENTIX_FINGERPRINT_DATABASE_USER

Table B–8 SQLNET.IDENTIX_FINGERPRINT_DATABASE_USER

Description	This parameter specifies the database user when using the Oracle fingerprint storage method: format = <username>.
Default	None

SQLNET.IDENTIX_FINGERPRINT_DATABASE_PASSWORD

Table B–9 SQLNET.IDENTIX_FINGERPRINT_DATABASE_PASSWORD

Description	This parameter specifies the database password when using the Oracle fingerprint storage method: format = <password>.
Default	None

Recommended Minimum Sets of Identix Parameters

Following are two sets of parameters: Oracle database method and file system method. You are presented with the minimum set of Identix parameters you need to define for each method.

Oracle Database Method

```
sqlnet.authentication_services = (beq, identix)
sqlnet.identix_fingerprint_method = oracle
sqlnet.identix_database_directory = identix_scanner
sqlnet.identix_fingerprint_database_user
sqlnet.identix_fingerprint_database_password
```

File System Method

```
sqlnet.authentication_services = (beq, identix)
sqlnet.identix_fingerprint_method = file
sqlnet.identix_database_directory = /etc/ofm_storage
```

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–10 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_REALMS=/krb5/krb.realms
	SQLNET.KERBEROS5_KEYTAB=/krb5/v5srvtab

initialization parameter file	REMOTE_OS_AUTHENT=FALSEOS_AUTHENT_PREFIX=""

Parameters for Clients and Servers using SecurID Authentication

Following is list of parameters to insert into the configuration files for clients and servers using SecurID.

Table B–11 SecurID Authentication Parameters

File Name	Configuration Parameters
sqlnet.ora	SQLNET.AUTHENTICATION_SERVICES=(securid)

initialization parameter file	REMOTE_OS_AUTHENT=FALSE
	OS_AUTHENT_PREFIX=""

Parameters for Clients and Servers using RADIUS Authentication

The following sections describe the parameters for Identix authentication

- sqlnet.ora File Parameters
- Recommended Minimum Sets of RADIUS Parameters
- Initialization File Parameters

sqlnet.ora File Parameters

SQLNET.AUTHENTICATION_SERVICES

Table B–12 SQLNET.AUTHENTICATION_SERVICES

Description	Configure the client or the server to use the RADIUS adapter: value = radius.
Default	None

SQLNET.RADIUS_AUTHENTICATION

Table B–13 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–14 SQLNET.RADIUS_AUTHENTICATION_PORT

Description	To set the listening port of the primary RADIUS server.
Default	1645

SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

Table B–15 SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

Description	To set the time to wait for response.
Default	5

SQLNET.RADIUS_AUTHENTICATION_RETRIES

Table B–16 SQLNET.RADIUS_AUTHENTICATION_RETRIES

Description	To set the number of times to re-send.
Default	3

SQLNET.RADIUS_SEND_ACCOUNTING

Table B–17 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–18 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–19 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–20 SQLNET.RADIUS_ALTERNATE_PORT

Description	To set the listening port for the alternate RADIUS server.
Default	1645

SQLNET.RADIUS_ALTERNATE_TIMEOUT

Table B–21 SQLNET.RADIUS_ALTERNATE_TIMEOUT

Description	To set the time to wait for response.
Default	5

SQLNET.RADIUS_ALTERNATE_RETRIES

Table B–22 SQLNET.RADIUS_ALTERNATE_RETRIES

Description	To set the number of times to re-send messages.
Default	3

SQLNET.RADIUS_CHALLENGE_RESPONSE

Table B–23 SQLNET.RADIUS_CHALLENGE_RESPONSE

Description	To turn challenge/response support ON/OFF.
Default	OFF

SQLNET.RADIUS_CHALLENGE_KEYWORD

Table B–24 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–25 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

SQLNET.RADIUS_CLASSPATH

Table B–26 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.
Default	There is no default. You must add this parameter to the `sqlnet.ora` file.

Recommended Minimum Sets of RADIUS Parameters

Following are two set of sample `sqlnet.ora` file RADIUS authentication parameters: one for "Static User Name and Password" and the other for "Challenge Response Mode".

Static User Name and Password

The following sample `sqlnet.ora` file shows the minimum set of RADIUS authentication parameters you need to configure for static user name and password PAP mode authentication with no accounting.

```
sqlnet.authentication_services = (radius)
sqlnet.authentication = IP-address-of-RADIUS-server
sqlnet.radius_secret = %ORACLE_HOME/network/security/radius.key (default value)
```

Challenge Response Mode

The following sample `sqlnet.ora` file shows the minimum set of RADIUS authentication parameters you need to configure for challenge response mode authentication using token cards or biometric authentication methods.

```
sqlnet.authentication_services = (radius)
sqlnet.authentication = IP-address-of-RADIUS-server
sqlnet.radius_challenge_response = ON
sqlnet.radius_secret = $ORACLE_HOME/network/security/radius.key (default value)
sqlnet.authentication_interface = oracle/net/radius/DefaultRadiusInterface
(default value)
sqlnet.radius_classpath = %ORACLE_HOME/jlib/netradius.jar (default value)
```

Initialization File 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 Net8 address.

Authentication Parameters

Table B–27 SSL Authentication Parameters

Parameter Name (static):	SQLNET.AUTHENTICATION_SERVICES
Parameter Name (dynamic):	AUTHENTICATION
Parameter Type:	String LIST
Parameter Class:	Static
Allowable 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, securid)`
Syntax (dynamic):	`AUTHENTICATION =` *`string`*
Example (dynamic):	`AUTHENTICATION = (TCPS)`

Cipher Suites

Table B–28 Cipher Suite Parameters

Parameter Name (static):	SSL_CIPHER_SUITES
Parameter Name (dynamic):	SSL_CIPHER_SUITES
Parameter Type:	String LIST
Parameter Class:	Static
Allowable 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	New
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–29 SSL Version Parameters

Parameter Name (static):	SSL_VERSION
Parameter Name (dynamic):	SSL_VERSION
Parameter Type:	string
Parameter Class:	Static
Allowable 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 (static):	SSL_VERSION=*version*
Example (dynamic):	SSL_VERSION=3.0

SSL Client Authentication

Table B–30 SSL Client Authentication Parameters

Parameter Name (static):	SSL_CLIENT_AUTHENTICATION
Parameter Name (dynamic):	SSL_CLIENT_AUTHENTICATION
Parameter Type:	Boolean
Parameter Class:	Static
Allowable Values:	TRUE/FALSE
Default Value:	TRUE
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*}*

Table B–30 SSL Client Authentication Parameters

Example (static):	`SSL_CLIENT_AUTHENTICATION=FALSE`
Syntax (dynamic):	`SSL_CLIENT_AUTHENTICATION={TRUE | FALSE}`
Example (dynamic):	`SSL_CLIENT_AUTHENTICATION=FALSE`

Wallet Location

For any application that needs to access a wallet for loading the security credentials into the process space, you must specify the wallet location in the parameter file it reads. The syntax of the parameter for static configuration is as follows:

```
oss.source.my_wallet =
(SOURCE=
  (METHOD=File)
  (METHOD_DATA=
    (DIRECTORY=your wallet location)
      )
    )
```

The dynamic way of specifying this parameter is:

```
MY_WALLET_DIRECTORY = your_wallet_dir
```

The default wallet location is the `$ORACLE_HOME` directory.

C

Integrating Authentication Devices Using RADIUS

This appendix explains how third party vendors of authentication devices customize the RADIUS challenge-response user interface to fit their particular device.

This appendix includes the following topics:

- About the RADIUS Challenge-Response User Interface
- Customizing the RADIUS Challenge-Response User Interface

More Information: See Chapter 4.

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.

Oracle has developed a Java interface class for this interface. It is a set of methods written in C code using the Java Native Interface as specified in release 1.1 of the Java Development Kit^{TM} from JavaSoft. This code, provided below, is system specific. You can find it in the file OracleRadiusInterface in the following directory: `$ORACLE_HOME/network/security/classes`.

Customizing the RADIUS Challenge-Response User Interface

You customize this interface by creating your own class to handle the challenge-response conversation between the Oracle client and the RADIUS server. You then open the `sqlnet.ora` file, look up the SQLNET.RADIUS_ AUTHENTICATION_INTERFACE parameter, and replace the name of the class listed there, namely, 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 relayed as the password from the server. If the user enters a valid password, a challenge may or may not be returned by the server.
radiusChallenge	Presents a request sent from the RADIUS server for the user to enter more information.
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 is undergoing U.S. Federal Information Processing Standards (FIPS) 140-1 validation 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. At the time of printing, the validation effort was not complete; check the NIST Cryptographic Modules Validation list at the following web site to see if validation has completed:

`http://csrc.nist.gov/cryptval/140-1/1401val.htm`

This appendix includes the following topics:

- Configuration Parameters
- Post Installation Checks
- Status Information

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
- 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 allowed 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 CRYPTO_SEED 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.CRYPTO_SEED_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 Net8 Assistant 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.

E

LDAP Directory Schema for Oracle Database Security

This appendix describes the object classes and attributes defined in the LDAP directory schema for Oracle database security.

This appendix includes the following topics:

- Structural Object Classes
- Attributes
- Access Controls

Structural Object Classes

Table E–1 lists the structural object classes and associated attributes related to the LDAP directory schema for Oracle database security.

Table E–1 Structural Object Classes and Attributes

Object Class	Database Attributes
orclDBServer	orclDBGlobalName
orclDBEnterpriseDomain	orclDBServerMember orclDBTrustedDomain
orclDBEnterpriseRole	orclDBServerRole orclDBRoleOccupant
orclDBEntryLevelMapping	orclDBDistinguishedName orclDBNativeUser
orclDBSubtreeLevelMapping	orclDBDistinguishedName orclDBNativeUser

Attributes

Table E–2 describes the attributes in the LDAP directory schema for Oracle database security.

Table E–2 LDAP Directory Schema Attributes

Attribute	Description
orclDBGlobalName	Identifies the global name of the server
orclDBServerMember	Defines a list of databases that are members of the domain
orclDBTrustedDomain	Indicates whether current user database links function between databases in the domain
orclDBServerRole	Defines a list of included global roles in the databases in the domain
orclDBRoleOccupant	Defines a list of users or groups to whom this enterprise role has been assigned
orclDBDistinguishedName	Specifies the full distinguished name of the enterprise user
orclDBNativeUser	Specifies the database shared schema name

Access Controls

Table E–3 describes the minimum required access controls for the LDAP directory security objects.

> **Note:** Members of the OracleDBSecurityAdmins group require create, update, and read access to all objects in the directory.

Table E–3 Minimum Required Access to Directory Objects.

Object	Created By	Updated By	Read By
database server	database creator during installation	DBA for the database	anyone
database-level mapping	DBA for the database, using Oracle Enterprise Security Manager	DBA for the database, using Oracle Enterprise Security Manager	■ database ■ DBA for the database, using Oracle Enterprise Security Manager
Oracle Default Domain	Oracle Context creator, using Net8 Configuration Assistant	domain administrator for the Oracle Default Domain, using Enterprise Security Manager database creator, using Oracle Database Configuration Assistant; can only modify the domain, not the subordinate roles	■ database creator, using Oracle Database Configuration Assistant ■ databases in the domain ■ domain administrator, using Oracle Enterprise Security Manager
enterprise domain	database security administrator, using Oracle Enterprise Security Manager	domain administrator	databases in the domain
domain-level mapping	domain administrator for the domain, using Oracle Enterprise Security Manager	domain administrator for the domain, using Oracle Enterprise Security Manager	■ domain administrator for the domain, using Oracle Enterprise Security Manager ■ databases in the domain

Table E–3 Minimum Required Access to Directory Objects.

Object	Created By	Updated By	Read By
enterprise role	domain administrator, using Oracle Enterprise Security Manager	domain administrator, using Oracle Enterprise Security Manager	■ databases in the domain ■ domain administrator using Oracle Enterprise Security Manager
OracleDBSecurityAdmins group	Oracle Context creator, using Net8 Configuration Assistant	database security administrator	database security administrator
OracleDBCreators group	Oracle Context creator, using Net8 Configuration Assistant	database security administrator	database creators

Note: Database security administrators are members of the OracleDBSecurityAdmins group. Database creators are members of the OracleDBCreators group.

More Information:

- For information on the context and usage of the Oracle security schema for LDAP, see Chapter 17, "Managing Enterprise User Security".
- For information on the OracleNetAdmins group, see the *Net8 Administrator's Guide*.

F

Oracle Implementation of Java SSL

This appendix provides an overview of the components and usage of the Oracle implementation of Java SSL; a standard extension of the JavaSoft Java platform.

The following topics are in this appendix:

- Oracle Java SSL Features
- SSL Cipher Suite Supported in Oracle Java SSL
- Certificate and Key Management with Oracle Wallet Manager
- Oracle Java SSL Examples
- Class Hierarchy for Extensions to the Java SSL Package
- Interface Hierarchy

Note: Readers are expected to know the basics of socket programming in Java and basic concepts of the SSL protocol.

More Information: See the Java documentation at the following web site for further information about Java SSL packages:

`http://java.sun.com/security/ssl/packages`

Oracle Java SSL Features

In addition to the SSL APIs and protocol implementation, the Oracle implementation of Java SSL supports the following:

- various cryptographic algorithms
- secure key management using Oracle Wallet Manager version 2.1
- X.509 certificate chain infrastructure for authentication policies used by the applications built of top of the Oracle implementation of Java SSL
- SSLSockets and SSLServerSockets, used like other sockets with SSL-specific features

> **Note:** The constructors on SSL sockets are not publicly accessible, and therefore APIs must be used to acquire SSL sockets.

- Socket factories, with which authentication contexts holding private keys, certificate chains, and similar data are associated
- SSL-specific session capabilities, including authentication
- SSL-specific exceptions that can be thrown

> **Note:** SSL-specific exceptions are all subtypes of IOException because it is the exception that can be thrown during I/O operations that produce the need to report such exceptions.

Choices related to the implementation of cryptographic security code are critically important, and therefore the interface defined by JavaSoft uses JNI native code instead of pure Java.

For example, in some environments hardware to accelerate cryptographic operations is important, and in other environments only specific implementations of cryptographic algorithms are permitted.

SSL Cipher Suite Supported in Oracle Java SSL

A cipher suite combines four kinds of security features and is named in the SSL protocol specification. Before data flows over an SSL connection, both ends attempt to negotiate a cipher suite. This allows them to establish the appropriate protection of their communications within the constraints of the particular combinations of mechanism that are available.

The features associated with a cipher suite are as follows:

- The implemented key exchange algorithms are RSA and Diffie-Hellman. The RSA authenticated key exchange algorithm is currently the most interoperable.
- There are two versions of Oracle Java SSL, exportable and domestic.

  The exportable version supports 512 bit keys for key exchange and 40 bit symmetric keys for encryption, and is not considered secure enough for use in commercial applications.

  The domestic version supports 768 and 1024 bit asymmetric keys for key exchange and 128 bit symmetric keys for encryption. The domestic version is considered secure enough for use in commercial applications.
- The following encryption algorithms are used:
  - RC4 stream cipher, which is the fastest option
  - DES and variants, DES40, 3DES-EDE
  - in cipher block chaining (CBC) mode
  - NULL encryption

  Note: With NULL encryption, SSL is only used to authenticate and provide integrity protection.
- The digest algorithm used for the Message Authentication Code are MD5 and SHA1.

Certificate and Key Management with Oracle Wallet Manager

Oracle Wallet Manager can be used to generate private key and public key pairs and certificate requests. An appropriate signer's certificate or certificates with the complete certificate chain should be added to produce a complete Oracle Wallet.

If there is a complete wallet with a certificate in Ready status, it can be exported in BASE64 format into a file using the menu option Operation ->ExportWallet. The file can be used to add SSL credentials in a Java SSL based program.

More Information: For information on Oracle Wallet Manager, see Appendix 18, "Using Oracle Wallet Manager".

If a user is not using Oracle Wallet Manager, the user can add individual components to a file and use them. In this case, the certificate should be added first, followed by the private key. The CA certificate and other trusted certificates should be added after the certificate and private key.

Oracle Java SSL Examples

The examples in this section demonstrate the following:

- How to write a SSL server and SSL client
- How a program can establish a secure connection to a server program using the SSL
- How the client can send data to and receive data from the server

The example shows an implementation of a client named SecureHelloClient, which connects to a server named SecureHelloServer. The server receives data from the client and sends a hello string.

The example consists of two independently running Java programs: the client program and the server program. The client program is implemented by a single class, SecureHelloClient. The server program is also implemented as a single class, SecureHelloServer, which contains the main method for the server program and performs the work of listening to the port, establishing connections, and data reading from and writing to the socket.

Prerequisites

The JDK version 1.1 or 1.2 should be installed with the following jar files in the CLASSPATH environment variable:

- For JDK1.1: javax-ssl-1_1.jar, jssl-1_1.jar
- For JDK1.2: javax-ssl-1_2.jar, jssl-1_2.jar

The following library should be added to the shared library path:

- For Solaris: libnjssl8.so

  Add the library path to LD_LIBRARY_PATH environment variable.
- For Windows NT: njssl8.dll

  Add the library path to PATH environment variable.

SecureHelloServer Program

This section describes the code that implements the SecureHelloServer program. The server program creates a new SSLServerSocketFactory and sets the required SSL protocol version as follows:

```
OracleSSLServerSocketFactory sslSrvSocketFactory
     = (OracleSSLServerSocketFactory)SSLServerSocketFactory.getDefault();
sslSrvSocketFactory.setSSLProtocolVersion(OracleSSLProtocolVersion.SSL_Version_
3_0);
```

Two scenarios are possible, depending on whether Oracle Wallet Manager is used.

If Oracle Wallet Manager is used to export the wallet, the setWallet API can be used to populate the OracleSSLCredential object as follows:

```
OracleSSLCredential sslCredObj = new OracleSSLCredential();
sslCredObj.setWallet("wlt.txt", "wltpasswd");
sslSrvSocketFactory.setSSLCredential(sslCredObj);
```

Note: The absolute path must be specified for wlt.txt if it is not in the current directory.

If the wallet is not generated by Oracle Wallet Manager, the user must set the following:

- private key
- certificate chain
- trusted certificates

> **Note:** The certificate chain must be set in the order listed, from root certificate, signer certificates, and user certificate.

The code for this scenario is as follows:

```
OracleSSLCredential sslCredObj = new OracleSSLCredential();
// Set trusted certificates
sslCredObj.addTrustedCert(easQACA);

// Construct certificate chain. Place CA at the top
// and user certificate at the bottom. The order of
// set certificates in the chain is important. You must set
// root certificate first, then signer certificates, and finally user
// certificate.
sslCredObj.addCertChain(rootCA); (set root CA certificate)
sslCredObj.addCertChain(signer CA);(set signer certificate)
sslCredObj.addCertChain(userCert); (set user certificate)

/*
 * Set private key
*/
sslCredObj.setPrivateKey(userKey, password);
```

If the Diffie-Hellman algorithm is being used, `setSSLCredentials` should be called with a null value as follows:

```
sslSrvSocketFactory.setSSLCredentials(null);
```

`SSLServerSocket` uses a specific port for listening. When writing a server, select a port that is not already dedicated to another service.

In this example, port 8443 is used as follows:

```
SSLServerSocket sslSrvSocket =
     (SSLServerSocket)sslSrvSocketFactory.createServerSocket(8443);
```

`SSLServerSocket` requires supported ciphers to be set as follows:

```
String [] ciphers = sslSrvSocket.getSupportedCipherSuites() ;
sslSrvSocket.setEnabledCipherSuites(ciphers);
```

Because this is a server, it is set to SSL server mode as follows:

```
sslSrvSocket.setUseClientMode(false);
```

Note: You can also use the client node to connect to another server.

Client authentication is not used in this example, and therefore `setNeedClientAuth` must be called with the parameter set to false as follows:

```
sslSrvSocket.setNeedClientAuth(false);
```

If client authentication is required, set `setNeedClientAuth` to TRUE.

To accept the client connection, `accept()` must be called, which returns a socket object. Using this socket, regular reads and writes can be performed similar to a regular socket object by calling `getOutputStream()` and `getInputStream()` as follows:

```
OutputStream  out = pSocket.getOutputStream();
InputStream   in  = pSocket.getInputStream();
```

After data is exchanged, close all streams and sockets before exiting the application as follows:

```
out.close();
in.close();
pSocket.close();
sslSrvSocket.close();
```

Note: The SSL package is used with the certificate package. However, there is a different certificate package for different JDK releases. Import the correct certificate package as follows:

- For JDK 1.1, import javax.security.cert.X509Certificate
- For JDK 1.2, import java.security.cert.X509Certificate

The complete SecureHelloServer example for JDKI 1.1 is as follows.

```
// SecureHelloServer.java

    import java.net.*;
    import java.io.*;
    import java.util.*;
    import java.lang.*;

    import javax.net.*;
    import javax.net.ssl.*;

    import javax.security.cert.X509Certificate;
    import oracle.security.ssl.OracleSSLServerSocketFactoryImpl;
    import oracle.security.ssl.OracleSSLServerSocketFactory;
    import oracle.security.ssl.OracleSSLProtocolVersion;
    import oracle.security.ssl.OracleSSLCredential;


    public class SecureHelloServer
    {
        public static void main(String[] args)
        {
            // We will use Oracle implementation here
            java.util.Properties prop = System.getProperties();
            prop.put("SSLServerSocketFactoryImplClass",
                  "oracle.security.ssl.OracleSSLServerSocketFactoryImpl");
            try
            {
                // Get the default socket factory
                OracleSSLServerSocketFactory sslSrvSocketFactory
                     = (OracleSSLServerSocketFactory)SSLServerSocketFactory.getDefault();

                // Set the SSL protocol version
                sslSrvSocketFactory.setSSLProtocolVersion(OracleSSLProtocolVersion.SSL_Version_3_0);

                // Create the ssl credential object
                OracleSSLCredential sslCredObj = new OracleSSLCredential();

               // If you are using Oracle’s wallet, certdb.txt, you can use setWallet as follows:
                sslCredObj.setWallet(certdb.txt,password)

                // If you are not using Oracle Wallet Manager, see the SecureHelloClient
                // program example.

                    // Add ssl credential to the ssl context
                sslSrvSocketFactory.setSSLCredentials(sslCredObj);

                // Create the server socket
                SSLServerSocket sslSrvSocket =
                 (SSLServerSocket)sslSrvSocketFactory.createServerSocket(8443);

                // Print the available ciphers
                String [] ciphers = sslSrvSocket.getSupportedCipherSuites() ;

                // Select the ciphers you want and put it.
                // Here we will put all available ciphers.
                // You can also set particular cipher suite.
                // Construct a cipher list and in a string array and
                // pass it to setEnabledCipherSuites.
                sslSrvSocket.setEnabledCipherSuites(ciphers);

                // We are creating ssl server socket, so set the mode to false.
                sslSrvSocket.setUseClientMode(false);

                // If you want  do client side authentication then set it to true.
                // We won't do client side authintication here.
                sslSrvSocket.setNeedClientAuth(false);


                System.out.println("Wating for client...");
                // Now accept a client connection
                Socket pSocket = sslSrvSocket.accept();

                if (sslSrvSocket.getNeedClientAuth() == true)
               {
                    System.out.println("Printing client information:");
                    X509Certificate[] peerCerts
                            =
    ((javax.net.ssl.SSLSocket)pSocket).getSession().getPeerCertificateChain();

                     if (peerCerts != null)
                     {
                          for(int i =0; i ? peerCerts.length; i++)
                          {
                              System.out.println("Peer Certificate  ["+i+"] Information:");
                              System.out.println("- Subject: " +
    peerCerts[i].getSubjectDN().getName());
                              System.out.println("- Issuer: " + peerCerts[i].getIssuerDN().getName());
                              System.out.println("- Version: " + peerCerts[i].getVersion());
                              System.out.println("- Start Time: " +
    peerCerts[i].getNotBefore().toString());
                              System.out.println("- End Time: " +
    peerCerts[i].getNotAfter().toString());
                              System.out.println("- Signature Algorithm: " +
    peerCerts[i].getSigAlgName());
                              System.out.println("- Serial Number: " + peerCerts[i].getSerialNumber());

                         }
                     }
                     else
                        System.out.println("Failed to get peer certificates");
                 }

                // Now do data exchange with client
                OutputStream  out = pSocket.getOutputStream();
                InputStream   in  = pSocket.getInputStream();

                String inputLine, outputLine;
                byte []  msg = new byte[1024];

                int readLen = in.read(msg, 0, msg.length);
                if(readLen>0)
                {
                    inputLine = new String(msg, 0, readLen);
                    if(inputLine.startsWith("HELLO"))
                    {
                        outputLine = "Hello !! Current Server Time: " + new Date().toString();
                        outputLine.getBytes();
                        out.write(outputLine.getBytes());
                    }
                    System.out.println("Client Message: " + inputLine );
                }
                else
                    System.out.println("Can’t read data from client");

                // Close all sockets and streams
                out.close();
                in.close();
                pSocket.close();
                sslSrvSocket.close();
            }
            catch(SSLException e)
            {
                System.out.println("SSL exception caught:");
                e.printStackTrace();
            }
            catch(IOException e)
            {
                System.out.println("IO exception caught:");
                e.printStackTrace();
            }
            catch(Exception e)
            {
                System.out.println("Exception caught:");
                e.printStackTrace();
            }
        }
    }
```

SecureHelloClient Program

The client program creates a new client SSLSocketFactory and sets the required SSL protocol version as follows:

```
OracleSSLSocketFactory sSocFactory
     = (OracleSSLSocketFactory)SSLSocketFactory.getDefault();
sSocFactory.setSSLProtocolVersion(OracleSSLProtocolVersion.SSL_Version_3_0);
```

Because the RSA algorithm is used, the OracleSSLCredential object is required. Adding trusted certificates is optional. If no trusted certificates are set, the peer certificate will not be verified against any trusted certificates. If the server needs client authentication, the complete certificate chain and client private key must be added to the SSL credential object as follows:

```
OracleSSLCredential sslCredObj = new OracleSSLCredential();
sslCredObj.addTrustedCert(caCert);
sSocFactory.setSSLCredentials(sslCredObj);
```

Create a SSL socket for connecting to the server by creating a socket with the required host name and port, in this example 8443, as follows:

```
SSLSocket jsslSoc =
     (SSLSocket)sSocFactory.createSocket(hostName, 8443);
```

Set the required ciphers from the supported ciphers as follows:

```
String [] ciphers = jsslSoc.getSupportedCipherSuites() ;
jsslSoc.setEnabledCipherSuites(ciphers);
```

Set the socket to SSL client mode and call `startHandshake()` to perform the SSL handshake as follows:

```
jsslSoc.setUseClientMode(true);
jsslSoc.startHandshake();
```

Note: `setUseClientMode` is set to TRUE by default.

Obtain the input stream and output stream from the socket and perform standard data input and output as follows:

```
OutputStream  out = jsslSoc.getOutputStream();
InputStream   in  = jsslSoc.getInputStream();
```

After data exchange, close all streams and sockets as follows:

```
out.close();
in.close();
jsslSoc.close();
```

Note: The SSL package is used with the certificate package. However, there is a different certificate package for different JDK versions. Import the correct certificate package as follows:

- For JDK 1.1, import javax.security.cert.X509Certificate
- For JDK 1.2, import java.security.cert.X509Certificate

The complete `SecureHelloClient` example is as follows.

```
// SecureHelloClient.java
     import java.net.*;
     import java.io.*;
     import java.util.*;

     import javax.net.ssl.*;

     import javax.security.cert.X509Certificate;
     import oracle.security.ssl.OracleSSLCredential;
     import oracle.security.ssl.OracleSSLSocketFactory;
     import oracle.security.ssl.OracleSSLProtocolVersion;
     import oracle.security.ssl.OracleSSLSession;

     public class SecureHelloClient
     {
         public static void main(String argv[])
         {
             String hostName = "localhost";
             if(argv.length != 0)
               String hostName = argv[0];

             // Set the SSLSocketFactoryImpl class as follows:
             java.util.Properties prop = System.getProperties();
             prop.put("SSLSocketFactoryImplClass",
                 "oracle.security.ssl.OracleSSLSocketFactoryImpl");

             try
             {
                 // Get the default socket factory
                 OracleSSLSocketFactory sSocFactory
```

```
    = (OracleSSLSocketFactory)SSLSocketFactory.getDefault();

sSocFactory.setSSLProtocolVersion(OracleSSLProtocolVersion.SSL_Version_3_0);

OracleSSLCredential sslCredObj = new OracleSSLCredential();

// Set the certificate chain and private key if the
// server requires client authentication
sslCredObj.addCertChain(caCert)
sslCredObj.addCertchain(userCert)
sslCredObj.setPrivateKey(userPvtKey, userPassword)

// Populate credential object
sslCredObj.addTrustedCert(trustedCert);
sSocFactory.setSSLCredentials(sslCredObj);

// Create the socket using factory
SSLSocket jsslSoc =
    (SSLSocket)sSocFactory.createSocket(hostName, 8443);

String [] ciphers = jsslSoc.getSupportedCipherSuites() ;

// Select the ciphers you want and put them.
// Here we will put all availabel ciphers
jsslSoc.setEnabledCipherSuites(ciphers);

// We are creating socket in client mode
jsslSoc.setUseClientMode(true);

// Do SSL handshake
jsslSoc.startHandshake();

// Print negotiated cipher
System.out.println("Negotiated Cipher Suite: "
    +jsslSoc.getSession().getCipherSuite());

System.out.println("");
X509Certificate[] peerCerts
        = ((javax.net.ssl.SSLSocket)jsslSoc).getSession().getPeerCertificateChain();

 if (peerCerts != null)
 {
      System.out.println("Printing server information:");
      for(int i =0; i ? peerCerts.length; i++)
      {
          System.out.println("Peer Certificate  ["+i+"] Information:");
          System.out.println("- Subject: " + peerCerts[i].getSubjectDN().getName());
          System.out.println("- Issuer: " + peerCerts[i].getIssuerDN().getName());
          System.out.println("- Version: " + peerCerts[i].getVersion());
```

```
                      System.out.println("- Start Time: " +
peerCerts[i].getNotBefore().toString());
                      System.out.println("- End Time: " + peerCerts[i].getNotAfter().toString());
                      System.out.println("- Signature Algorithm: " + peerCerts[i].getSigAlgName());

                      System.out.println("- Serial Number: " + peerCerts[i].getSerialNumber());
                 }
             }
             else
                System.out.println("Failed to get peer certificates");

            // Now do data exchange with client
            OutputStream  out = jsslSoc.getOutputStream();
            InputStream   in  = jsslSoc.getInputStream();

            String inputLine, outputLine;
            byte []  msg = new byte[1024];

            outputLine = "HELLO";
            out.write(outputLine.getBytes());
            int readLen = in.read(msg, 0, msg.length);
            if(readLen > 0)
            {
                inputLine = new String(msg, 0, readLen);
                System.out.println("");
                System.out.println("Server Message:");
                System.out.println(inputLine );
            }
            else
               System.out.println("Can’t read data from client");

            // Close all sockets and streams
            out.close();
            in.close();
            jsslSoc.close();
        }
        catch(SSLException e)
        {
            System.out.println("SSL exception caught:");
            e.printStackTrace();
        }
        catch(IOException e)
        {
            System.out.println("IO exception caught:");
            e.printStackTrace();
        }
        catch(Exception e)
        {
            System.out.println("Exception caught:");
            e.printStackTrace();
            }
        }
    }
```

Firewall Tunnelling Program Using the SSL Socket

The following example shows how to use the Java SSL Socket with firewall tunnelling.

```
import java.net.*;
import java.io.*;
import java.util.*;
import java.lang.*;

import java.security.cert.*;
import javax.net.ssl.*;

import oracle.security.ssl.OracleSSLCredential;
import oracle.security.ssl.OracleSSLSocketFactory;
import oracle.security.ssl.OracleSSLProtocolVersion;

public class SSLSocketTest
{
    public static void main(String argv[])
    {

        if(OracleSSLCipher.isSSLLibDomestic())
            System.out.println("Domestic SSL library");
        else
            System.out.println("Export SSL library");

        String hostName = "";
        int i          = 0;

        try
        {
            hostName = argv[0];
        } catch (Exception e)
        {
            hostName = "localhost";
        }

        try
        {
            i = (new Integer(argv[1])).intValue();
        }
        catch (Exception e)
        {
            i = 443;
```

```
}

String proxy = System.getProperty("PROXY");
String certdb = System.getProperty("CERTDBFILE");

java.util.Properties prop = System.getProperties();
prop.put("SSLSocketFactoryImplClass", "oracle.security.ssl.OracleSSLSocketFactoryImpl");

try
{
   /*
    * User can set their own x.509 impl. class and the default
    * is set to the oracle impl. in the factory class
    * java.security.Security.setProperty("cert.provider.x509v1",
    * "oracle.security.cert.X509CertificateImpl");
    */

    // Get the default socket factory
    OracleSSLSocketFactory sSocFactory
        = (OracleSSLSocketFactory)OracleSSLSocketFactory.getDefault();

    // sSocFactory.setSSLProtocolVersion(OracleSSLProtocolVersion.SSL_Version_3_0_With_2_0_Hello);
    sSocFactory.setSSLProtocolVersion(OracleSSLProtocolVersion.SSL_Version_3_0);

    OracleSSLCredential sslCredObj = new OracleSSLCredential();

    if (certdb == null)
      System.out.println("certdb is null");
    else
      sslCredObj.setWallet (certdb, "welcome12");

    /*
     * Populate credential object
     */

     sSocFactory.setSSLCredentials(sslCredObj);

    SSLSocket jsslSoc = null;

    // Create a regular java Socket connect to proxy server
    // www-proxy1
    // port 80

    Socket soc = new Socket("www-proxy1", 80);
    if (makeProxyConnection(soc, hostName, i))
    {
      System.out.println("Proxy enable sucessfully");
    }
    // Pass the soc generated using
    // Java SSLSocket Constructor
```

```
            jsslSoc = (SSLSocket)sSocFactory.createSocket(soc);

            // Now you can use the jsslSoc for ssl connection
            // to a ssl server through a proxy server
            java.security.cert.X509Certificate[] peerCerts
              = jsslSoc.getSession().getPeerCertificateChain();

            exchangeData(jsslSoc);

            jsslSoc.close();


        }
        catch(Exception e)
        {
                e.printStackTrace();
        }
        System.exit(0);
    }

    // Connect string needs to be set up for firewall tunnelling connection
    private static boolean makeProxyConnection(Socket pjsoc, String host, int port)
    {
        try
        {
          InputStream  rawInStream  = pjsoc.getInputStream();
          OutputStream rawOutStream = pjsoc.getOutputStream();
          String portStr = String.valueOf(port);
          String connString
            =   "CONNECT "+host+":"+portStr+" HTTP/1.0 \n"
            + "User-Agent: Oracle Proxy Enabled SSL Socket \n\n";
          rawOutStream.write(connString.getBytes(), 0, connString.length());
          byte[] pxyMsg = new byte[2048];
          int rdData = rawInStream.read(pxyMsg, 0, 2048);
          System.out.println("Proxy Message:\n"+ new String(pxyMsg, 0, rdData));
          return true;
        }
        catch(Exception e)
        {
          return false;
        }
}
    public static void exchangeData(SSLSocket sslSoc) throws
        IOException
    {

      String outs = "GET / HTTP/1.0 \r\n\r\n";

      BufferedInputStream isr = new
      BufferedInputStream(sslSoc.getInputStream(), 8192);
      BufferedOutputStream os = new
      BufferedOutputStream(sslSoc.getOut putStream(), outs.length());

      os.write(outs.getBytes(), 0, outs.length());
      os.flush();
      System.out.println("Server Response:");
      System.out.println("----------------");

      String srvResp = new String();
      byte[] srvmsg = new byte[4096*2];
      int n = 0;
      do
      {
            n = isr.read(srvmsg, 0, srvmsg.length);
        if(n > 0)
      }
      os.close();
      isr.close();
    }


}
```

Class Hierarchy for Extensions to the Java SSL Package

The following is the class hierarchy for the extensions to the Java SSL package for JDK 1.2.

```
class java.lang.Object
    class java.security.cert.Certificate
        class java.security.cert.X509Certificate (implements
        java.security.cert.X509Extension)
            class oracle.security.cert.X509CertificateImpl
    class java.security.cert.CertificateFactory
        class oracle.security.cert.OracleCertificateFactory
    class oracle.security.ssl.OracleSSLCredential
    class oracle.security.ssl.OracleSSLSession (implements
        javax.net.ssl.SSLSession)
    class javax.net.ServerSocketFactory
        class javax.net.ssl.SSLServerSocketFactory
            class oracle.security.ssl.OracleSSLServerSocketFactory
                class oracle.security.ssl.OracleSSLServerSocketFactoryImpl
    class javax.net.SocketFactory
        class javax.net.ssl.SSLSocketFactory
            class oracle.security.ssl.OracleSSLSocketFactory
                class oracle.security.ssl.OracleSSLSocketFactoryImpl
```

More Information: See the current Java documentation for information on complete class packages.

Interface Hierarchy

The interface hierarchy is as follows:

```
interface oracle.security.ssl.OracleSSLProtocolVersion
```

oracle.security.ssl

Description

Class Summary

Interfaces

OracleSSLProtocolVersion

Classes

OracleSSLCredential

OracleSSLServerSocket

OracleSSLServerSocketFactory

OracleSSLServerSocketFactoryImpl

OracleSSLSession

OracleSSLSocketFactory

OracleSSLSocketFactoryImpl

SSLSocketSession

oracle.security.ssl
OracleSSLCredential

Syntax

```
public class OracleSSLCredential extends java.lang.Object
```

```
java.lang.Object
  |
  +--oracle.security.ssl.OracleSSLCredential
```

Description

Member Summary

Constructors

OracleSSLCredential()

Methods

addCertChain(byte[])

addCertChain(String)

addTrustedCert(byte[])

addTrustedCert(String)

removeCertChainCert(int)

removeTrustedCert(int)

setPrivateKey(byte[], String)

setPrivateKey(String, String)

setWallet(String, String)

toString()

Inherited Member Summary

Methods inherited from class java.lang.Object

clone, equals, finalize, getClass, hashCode, notify, notifyAll, wait, wait, wait

Constructors

OracleSSLCredential()

```
public  OracleSSLCredential()
```

Methods

addCertChain(byte[])

```
public void addCertChain(byte[] certChainCert)
```

addCertChain(String)

```
public void addCertChain(java.lang.String b64certChainCert)
```

addTrustedCert(byte[])

```
public void addTrustedCert(byte[] trustedCert)
```

addTrustedCert(String)

removeCertChainCert(int)

```
public void removeCertChainCert(int index)
```

removeTrustedCert(int)

```
public void removeTrustedCert(int index)
```

setPrivateKey(byte[], String)

```
public void setPrivateKey(byte[] pvtKey, java.lang.String password)
```

setPrivateKey(String, String)

```
public void setPrivateKey(java.lang.String b64PvtKey, java.lang.String password)
```

setWallet(String, String)

```
public void setWallet(java.lang.String wltPath, java.lang.String password)
```

toString()

```
public java.lang.String toString()
```

Overrides:

java.lang.Object.toString() in class java.lang.Object

oracle.security.ssl
OracleSSLProtocolVersion

Syntax

```
public interface OracleSSLProtocolVersion
```

All Known Implementing Classes:

`OracleSSLServerSocket`

Description

Member Summary

Fields	
SSL_Version_2_0	SSL protocol version 2.0
SSL_Version_3_0	SSL protocol version 3.0
SSL_Version_3_0_Only	SSL protocol version 3.0 only
SSL_Version_3_0_With_2_0_Hello	SSL protocol version 3.0 with 2.0 hello
SSL_Version_Undetermined	SSL protocol version undetermined

Fields

SSL_Version_2_0

```
public static final int SSL_Version_2_0
```

SSL protocol version 2.0

Since:

1.0

SSL_Version_3_0

`public static final int SSL_Version_3_0`

SSL protocol version 3.0

Since:

1.0

SSL_Version_3_0_Only

`public static final int SSL_Version_3_0_Only`

SSL protocol version 3.0 only

Since:

1.0

SSL_Version_3_0_With_2_0_Hello

`public static final int SSL_Version_3_0_With_2_0_Hello`

SSL protocol version 3.0 with 2.0 hello

Since:

1.0

SSL_Version_Undetermined

`public static final int SSL_Version_Undetermined`

SSL protocol version undetermined

Since:

1.0

oracle.security.ssl
OracleSSLServerSocket

Syntax

```
public abstract class OracleSSLServerSocket implements OracleSSLProtocolVersion
```

oracle.security.ssl.OracleSSLServerSocket

All Implemented Interfaces:

OracleSSLProtocolVersion

Description

Member Summary

Constructors	
OracleSSLServerSocket(int)	Default constructor Creats a server socket which uses all network interfaces on the host, and is bound to the specified port.
OracleSSLServerSocket(int, int)	Creats a a server socket which uses all network interfaces on the host, is bound to a the specified port, and uses the specified connection backlog.
OracleSSLServerSocket(int, int, InetAddress)	Creats a server socket which uses only the specified network interface on the local host, is bound to a the specified port, and uses the specified connection backlog.
Methods	
setSSLProtocolVersion(int)	Sets the SSL protocol version

Inherited Member Summary

Fields inherited from interface OracleSSLProtocolVersion

SSL_Version_2_0, SSL_Version_3_0, SSL_Version_3_0_Only, SSL_Version_3_0_With_2_0_Hello, SSL_Version_Undetermined

Constructors

OracleSSLServerSocket(int)

`protected OracleSSLServerSocket(int i)`

Default constructor Creats a server socket which uses all network interfaces on the host, and is bound to the specified port.

Parameters:

`port` - the port number, or `0` to use any free port.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLServerSocketImpl`

OracleSSLServerSocket(int, int)

`protected OracleSSLServerSocket(int i, int j)`

Creats a a server socket which uses all network interfaces on the host, is bound to a the specified port, and uses the specified connection backlog.

Parameters:

`port` - the specified port, or `0` to use any free port.

`backlog` - the maximum length of the queue.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLServerSocketImpl`

OracleSSLServerSocket(int, int, InetAddress)

`protected  OracleSSLServerSocket(int i, int j, java.net.InetAddress inetAddr)`
Creats a server socket which uses only the specified network interface on the local host, is bound to a the specified port, and uses the specified connection backlog.

Parameters:

`port` - the local TCP port

`backlog` - the listen backlog

`bindAddr` - the local InetAddress the server will bind to

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLServerSocketImpl`

Methods

setSSLProtocolVersion(int)

`public abstract void setSSLProtocolVersion(int version)`
Sets the SSL protocol version

Parameters:

`version` - SSL protocol version

Throws:

Since:

1.0

oracle.security.ssl
OracleSSLServerSocketFactory

Syntax

```
public abstract class OracleSSLServerSocketFactory
```

`oracle.security.ssl.OracleSSLServerSocketFactory`

Direct Known Subclasses:

`OracleSSLServerSocketFactoryImpl`

Description

Member Summary

Constructors	
OracleSSLServerSocketFactory()	
Methods	
setSSLCredentials(OracleSSLCredential)	Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection
setSSLProtocolVersion(int)	Sets the SSL protocol version

Constructors

OracleSSLServerSocketFactory()

```
public  OracleSSLServerSocketFactory()
```

Methods

setSSLCredentials(OracleSSLCredential)

`public abstract void setSSLCredentials(OracleSSLCredential sslCredential)`

Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection

Returns:

none

Since:

1.0

setSSLProtocolVersion(int)

`public abstract void setSSLProtocolVersion(int version)`

Sets the SSL protocol version

Parameters:

`version` - SSL protocol version

Throws:

Since:

1.0

oracle.security.ssl OracleSSLServerSocketFactoryImpl

Syntax

```
public class OracleSSLServerSocketFactoryImpl extends
OracleSSLServerSocketFactory

OracleSSLServerSocketFactory
  |
  +--oracle.security.ssl.OracleSSLServerSocketFactoryImpl
```

Description

Member Summary

Constructors	
OracleSSLServerSocketFactoryImpl()	Default constructor
Methods	
createServerSocket(int)	Returns a server socket which uses all network interfaces on the host, and is bound to the specified port.
createServerSocket(int, int)	Returns a server socket which uses all network interfaces on the host, is bound to a the specified port, and uses the specified connection backlog.
createServerSocket(int, int, InetAddress)	Returns a server socket which uses only the specified network interface on the local host, is bound to a the specified port, and uses the specified connection backlog.
getDefaultCipherSuites()	Returns the list of cipher suites which are enabled by default.
getSupportedCipherSuites()	Returns the names of the cipher suites which could be enabled for use on an SSL connection created by this factory.
setSSLCredentials(OracleSSLCredential)	Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection
setSSLProtocolVersion(int)	Sets the SSL protocol version

Constructors

OracleSSLServerSocketFactoryImpl()

`public  OracleSSLServerSocketFactoryImpl()`

Default constructor

Since:

1.0

Methods

createServerSocket(int)

`public java.net.ServerSocket createServerSocket(int port)`

Returns a server socket which uses all network interfaces on the host, and is bound to the specified port.

Parameters:

`port` - the port number, or `0` to use any free port.

Returns:

a new instance of `SSLServerSocketImpl`.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLServerSocketImpl`

createServerSocket(int, int)

`public java.net.ServerSocket createServerSocket(int i, int j)`

Returns a server socket which uses all network interfaces on the host, is bound to a the specified port, and uses the specified connection backlog.

Parameters:

`port` - the specified port, or `0` to use any free port.

`backlog` - the maximum length of the queue.

Returns:

a new instance of `SSLServerSocketImpl`.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLServerSocketImpl`

createServerSocket(int, int, InetAddress)

```
public java.net.ServerSocket createServerSocket(int i, int j,
java.net.InetAddress inetAddress)
```

Returns a server socket which uses only the specified network interface on the local host, is bound to a the specified port, and uses the specified connection backlog.

Parameters:

`port` - the local TCP port

`backlog` - the listen backlog

`bindAddr` - the local InetAddress the server will bind to

Returns:

a new instance of `SSLServerSocketImpl`.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLServerSocketImpl`

getDefaultCipherSuites()

```
public java.lang.String[] getDefaultCipherSuites()
```

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication.

Returns:

an array of default cipher suites strings

Since:

1.0

getSupportedCipherSuites()

```
public java.lang.String[] getSupportedCipherSuites()
```

Returns the names of the cipher suites which could be enabled for use on an SSL connection created by this factory. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns:

an array of supported cipher suites strings

Since:

1.0

setSSLCredentials(OracleSSLCredential)

```
public void setSSLCredentials(OracleSSLCredential sslCredential)
```

Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection

Overrides:

setSSLCredentials(OracleSSLCredential) in class OracleSSLServerSocketFactory

Returns:

none

Since:

1.0

setSSLProtocolVersion(int)

`public void setSSLProtocolVersion(int version)`

Sets the SSL protocol version

Overrides:

setSSLProtocolVersion(int) in class OracleSSLServerSocketFactory

Parameters:

`version` - SSL protocol version

Throws:

Since:

1.0

oracle.security.ssl
OracleSSLSession

Syntax

```
public class OracleSSLSession
```

oracle.security.ssl.OracleSSLSession

Description

Member Summary

Constructors

OracleSSLSession()

Methods

getCipherSuite()	Returns the name of the SSL cipher suite which is used for all connections in the session.
getCreationTime()	Returns the time at which this Session representation was created, in milliseconds since midnight, January 1, 1970 UTC.
getId()	Returns the identifier assigned to this Session.
getLastAccessedTime()	Returns the last time this Session representation was accessed by the session level infrastructure, in * milliseconds since midnight, January 1, 1970 UTC.
getNegotiatedProtocolVersion()	
getPeerCertificateChain()	Returns the cert chain presented by the peer.
getPeerHost()	Returns the peer host name
getPeerRawCertificateChain()	
getSessionContext()	Returns the context in which this session is bound.
getValue(String)	Returns the object bound to the given name in the session's application layer data.
getValueNames()	Returns an array of the names of all the application layer data objects bound into the Session.
invalidate()	Invalidates the session.
putValue(String, Object)	Binds the specified object into the session's application layer data with the given name.

Member Summary	
removeValue(String)	Removes the object bound to the given name in the session's application layer data.
setSSLSessionContext(byte[])	Sets the ssl session context pointer for native layer

Constructors

OracleSSLSession()

```
public  OracleSSLSession()
```

Methods

getCipherSuite()

```
public java.lang.String getCipherSuite()
```

Returns the name of the SSL cipher suite which is used for all connections in the session. This defines the level of protection provided to the data sent on the connection, including the kind of encryption used and most spects of how authentication is done..

Returns:

The name of the session's cipher suite in `String` format

Since:

1.0

getCreationTime()

```
public long getCreationTime()
```

Returns the time at which this Session representation was created, in milliseconds since midnight, January 1, 1970 UTC.

Returns:

creation time in long format

Since:

1.0

getId()

```
public byte[] getId()
```

Returns the identifier assigned to this Session.

Returns:

byte array

Since:

1.0

getLastAccessedTime()

```
public long getLastAccessedTime()
```

Returns the last time this Session representation was accessed by the session level infrastructure, in * milliseconds since midnight, January 1, 1970 UTC. Access indicates a new connection being established using session data. Application level operations, such as getting or setting a value associated with the session, are not reflected in this access time.

This information is particularly useful in session management policies. For example, a session manager thread could leave all sessions in a given context which haven't been used in a long time; or, the sessions might be sorted according to age to optimize some task.

Returns:

last accessed time in long format

Since:

1.0

getNegotiatedProtocolVersion()

```
public java.lang.String getNegotiatedProtocolVersion()
```

getPeerCertificateChain()

```
public java.security.cert.X509Certificate[] getPeerCertificateChain()
```

Returns the cert chain presented by the peer.

Returns:

array of peer X.509 certificates, with the peers own cert first in the chain, and with the "root" CA last.

Throws:

Since:

1.0

getPeerHost()

`public java.lang.String getPeerHost()`
Returns the peer host name

Returns:

Peer hostname in `String` format

Since:

1.0

getPeerRawCertificateChain()

`public byte[][] getPeerRawCertificateChain()`

getSessionContext()

`public oracle.security.ssl.SSLSessionContext getSessionContext()`
Returns the context in which this session is bound. This context may be unavailable in some environments, in which case this method returns null.

Returns:

`SSLSessionContext`

Since:

1.0

getValue(String)

`public java.lang.Object getValue(java.lang.String name)`
Returns the object bound to the given name in the session's application layer data. Returns null if there is no such binding.

Parameters:

`name` - The name of the binding to find.

Returns:

The value bound to that name, or null if the binding does not exist.

Since:

1.0

getValueNames()

`public java.lang.String[] getValueNames()`
Returns an array of the names of all the application layer data objects bound into the Session. return the array of value names

Since:

1.0

invalidate()

`public void invalidate()`
Invalidates the session. Future connections will not be able to resume or join this session.

Since:

1.0

putValue(String, Object)

`public void putValue(java.lang.String name, java.lang.Object obj)`
Binds the specified object into the session's application layer data with the given name. Any existing binding with the same name is replaced. If the new (or existing) value implements the SSLSessionBindingListener interface, it is notified appropriately.

Parameters:

`name` - - the name to which the data object will be bound. This may not be null.

`value` - - the data object to be bound. This may not be null.

Since:

1.0

removeValue(String)

`public void removeValue(java.lang.String name)`

Removes the object bound to the given name in the session's application layer data. Does nothing if there is no object bound to the given name. If the value implements the SessionBindingListener interface, it is notified appropriately.

Parameters:

name - - the name of the object to remove

Since:

1.0

setSSLSessionContext(byte[])

`public void setSSLSessionContext(byte[] ssl_session)`

Sets the ssl session context pointer for native layer

Parameters:

ssl_session - in byte array format

Since:

1.0

oracle.security.ssl OracleSSLSocketFactory

Syntax

```
public abstract class OracleSSLSocketFactory
```

oracle.security.ssl.OracleSSLSocketFactory

Direct Known Subclasses:

OracleSSLSocketFactoryImpl

Member Summary

Constructors	
OracleSSLSocketFactory()	
Methods	
createSocket(Socket)	Creates an SSL Socket based on an existing plain socket
setSSLCredentials(OracleSSLCredential)	Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection
setSSLProtocolVersion(int)	Sets the SSL protocol version

Constructors

OracleSSLSocketFactory()

```
public  OracleSSLSocketFactory()
```

Methods

createSocket(Socket)

`public abstract java.net.Socket createSocket(java.net.Socket soc)`

Creates an SSL Socket based on an existing plain socket

Returns:

Socket

Since:

1.0

setSSLCredentials(OracleSSLCredential)

`public abstract void setSSLCredentials(OracleSSLCredential sslCredential)`

Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection

Returns:

none

Since:

1.0

setSSLProtocolVersion(int)

`public abstract void setSSLProtocolVersion(int version)`

Sets the SSL protocol version

Parameters:

`version` - SSL protocol version

Throws:

Since:

1.0

oracle.security.ssl OracleSSLSocketFactoryImpl

Syntax

```
public class OracleSSLSocketFactoryImpl extends OracleSSLSocketFactory

OracleSSLSocketFactory
  |
  +--oracle.security.ssl.OracleSSLSocketFactoryImpl
```

Description

Member Summary

Constructors	
OracleSSLSocketFactoryImpl()	Default constructor
Methods	
createSocket(InetAddress, int)	Returns a connected client socket to the specified port number on the specified host.
createSocket(InetAddress, int, InetAddress, int)	Creates a socket and connects it to the specified remote address on the specified remote port.
createSocket(Socket)	Returns a ssl client socket from an existing socket
createSocket(String, int)	Returns a connected client socket to the specified port number on the specified host.
createSocket(String, int, InetAddress, int)	Returns a socket and connects it to the specified remote host on the specified remote port.
getDefaultCipherSuites()	Returns the list of cipher suites which are enabled by default.
getSupportedCipherSuites()	Returns the names of the cipher suites which could be enabled for use on an SSL connection created by this factory.
setSSLCredentials(OracleSSLCredential)	Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection
setSSLProtocolVersion(int)	Sets the SSL protocol version

Constructors

OracleSSLSocketFactoryImpl()

`public  OracleSSLSocketFactoryImpl()`

Default constructor

Since:

1.0

Methods

createSocket(InetAddress, int)

`public java.net.Socket createSocket(java.net.InetAddress inetAddress, int port)`

Returns a connected client socket to the specified port number on the specified host.

Parameters:

`host` - the server name to connect in `InetAddress` format

`port` - the port number, or `0` to use any free port.

Returns:

a new instance of `SSLSocketImpl`.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLSocketImpl`

createSocket(InetAddress, int, InetAddress, int)

```
public java.net.Socket createSocket(java.net.InetAddress inetAddress1, int
port1, java.net.InetAddress inetAddress2, int port2)
```

Creates a socket and connects it to the specified remote address on the specified remote port. The Socket will also bind() to the local address and port supplied.

Parameters:

address - the remote address

port - the remote port

localAddr - the local address the socket is bound to

localPort - the local port the socket is bound to

Throws:

IOException - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLSocketImpl`

createSocket(Socket)

```
public java.net.Socket createSocket(java.net.Socket soc)
```

Returns a ssl client socket from an existing socket

Overrides:

createSocket(Socket) in class OracleSSLSocketFactory

Parameters:

an - socket object

Returns:

a new instance of SSLSocketImpl.

Throws:

IOException - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLSocketImpl`

createSocket(String, int)

`public java.net.Socket createSocket(java.lang.String host, int port)`

Returns a connected client socket to the specified port number on the specified host.

Parameters:

`host` - the server name to connect

`port` - the port number, or `0` to use any free port.

Returns:

a new instance of `SSLSocketImpl`.

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLSocketImpl`

createSocket(String, int, InetAddress, int)

```
public java.net.Socket createSocket(java.lang.String host, int port1,
java.net.InetAddress inetAddress, int port2)
```

Returns a socket and connects it to the specified remote host on the specified remote port. The Socket will also bind to the local address and port supplied.

Parameters:

`host` - the name of the remote host

`port` - the remote port

`localAddr` - the local address the socket is bound to

`localPort` - the local port the socket is bound to

Throws:

`IOException` - IO error when creating the socket.

Since:

1.0

See Also:

`oracle.security.ssl.SSLSocketImpl`

getDefaultCipherSuites()

`public java.lang.String[] getDefaultCipherSuites()`

Returns the list of cipher suites which are enabled by default. Unless a different list is enabled, handshaking on an SSL connection will use one of these cipher suites. The minimum quality of service for these defaults requires confidentiality protection and server authentication.

Returns:

an array of default cipher suites strings

Since:

1.0

getSupportedCipherSuites()

`public java.lang.String[] getSupportedCipherSuites()`

Returns the names of the cipher suites which could be enabled for use on an SSL connection created by this factory. Normally, only a subset of these will actually be enabled by default, since this list may include cipher suites which do not meet quality of service requirements for those defaults. Such cipher suites are useful in specialized applications.

Returns:

an array of supported cipher suites strings

Since:

1.0

setSSLCredentials(OracleSSLCredential)

`public void setSSLCredentials(OracleSSLCredential sslCredential)`

Creates authentication contexts (holding private keys, certificate chains, and similar data) for ssl connection

Overrides:

setSSLCredentials(OracleSSLCredential) in class OracleSSLSocketFactory

Returns:

none

Since:

1.0

setSSLProtocolVersion(int)

```
public void setSSLProtocolVersion(int version)
```

Sets the SSL protocol version

Overrides:

setSSLProtocolVersion(int) in class OracleSSLSocketFactory

Parameters:

`version` - SSL protocol version

Throws:

Since:

1.0

oracle.security.ssl
SSLSocketSession

Syntax

```
public class SSLSocketSession
```

oracle.security.ssl.SSLSocketSession

Description

Member Summary

Constructors	
SSLSocketSession()	
Methods	
getCipherSuite()	
getCreationTime()	
getId()	
getLastAccessedTime()	
getPeerCertificateChain()	
getPeerHost()	
getSessionContext()	getSessionContext Returns the context in which this session is bound.
getValue(String)	
getValueNames()	
invalidate()	
putValue(String, Object)	
removeValue(String)	

Constructors

SSLSocketSession()

```
protected  SSLSocketSession()
```

Methods

getCipherSuite()

```
public java.lang.String getCipherSuite()
```

getCreationTime()

```
public long getCreationTime()
```

getId()

```
public byte[] getId()
```

getLastAccessedTime()

```
public long getLastAccessedTime()
```

getPeerCertificateChain()

```
public oracle.security.ssl.X509Certificate[] getPeerCertificateChain()
```

getPeerHost()

```
public java.lang.String getPeerHost()
```

getSessionContext()

```
public oracle.security.ssl.SSLSessionContext getSessionContext()
```

getSessionContext Returns the context in which this session is bound. This context may be unavailable in some environments, in which case this method returns null.

getValue(String)

```
public java.lang.Object getValue(java.lang.String name)
```

getValueNames()

```
public java.lang.String[] getValueNames()
```

invalidate()

```
public void invalidate()
```

putValue(String, Object)

```
public void putValue(java.lang.String name, java.lang.Object obj)
```

removeValue(String)

```
public void removeValue(java.lang.String name)
```

oracle.security.cert
X509CertificateImpl

Syntax

```
public class SSLSocketTest extends java.lang.Object

java.lang.Object
  |
  +--java.security.cert.Certificate
        |
        +--java.security.cert.X509Certificate
              |
              +--oracle.security.cert.X509CertificateImpl
```

public class X509CertificateImpl

extends java.security.cert.X509Certificate

Description

Member Summary

Fields	
private	
X509CertificateHelper	
_x509CertHelper	Fields inherited from class java.security.cert.Certificate type
`Constructors`	
X509CertificateImpl()	Construct a uninitialized X509 Cert on which decode must later be called (or which may be deserialized).
X509CertificateImpl(byte[] buf)	Unmarshals a certificate from its encoded form, parsing the BER encoded bytes.
X509CertificateImpl(byte[] buf, int offset, int len)	Instantiates an X509Certificate with input certificate data
`Methods`	
checkValidity()	Checks for the validity of the certificate with current time
checkValidity(java.util.Date date)	Checks for the validity of the certificate with given time

Member Summary

decode(java.io.InputStream in)	Decodes the input stream data and instantiates an X509Certificate object, and initializes it with the data read from the nput stream inStream.
equals(java.lang.Object obj)	Checks for the equility
getBasicConstraints()	
getCriticalExtensionOIDs()	
getEncoded()	Returns the encoded certificate
getExtensionValue(java.lang.String oid)	
getIssuerDN()	Returns the certificate issuer DN
getIssuerUniqueID()	
getKeyUsage()	
getNonCriticalExtensionOIDs()	
getNotAfter()	Returns the date when this certificate will expired
getNotBefore()	Returns the date when this certificate will be vaild
getPublicKey()	Returns the encoded certificate
getSerialNumber()	Gets the serialNumber value from the certificate.
getSigAlgName()	Returns the signature algorithm
getSigAlgOID()	Returns the signature algorithm OID
getSigAlgParams()	Returns the signature algorithm paramaters
getSignature()	
getSubjectDN()	Returns the certificate subject DN
getSubjectUniqueID()	
getTBSCertificate()	
getVersion()	Returns the certificate version
hashCode()	Returns the public key of this certificate
hasUnsupportedCriticalExtension()	
toString()	Returns information about this certificate
verify(java.security.PublicKey key)	Checks for the validity of the input public key for this certificate

Member Summary	
verify(java.security.PublicKey key, java.lang.String sigProvider)	Checks for the validity of the input public key for this certificate
X509Certificate(java.io.InputStream in)	Instantiates an X509Certificate object, and initializes it with the data read from the input stream inStream.

Inherited Member Summary

Methods inherited from class java.security.cert.Certificate

getType

Methods inherited from class java.lang.Object

clone, finalize, getClass, notify, notifyAll, registerNatives, wait, wait, wait

Fields

_x509CertHelper

```
private X509CertificateHelper _x509CertHelper
```

Constructors

X509CertificateImpl

```
public X509CertificateImpl()
```

Construct a uninitialized X509 Cert on which decode must later be called (or which may be deserialized).

X509CertificateImpl

```
public X509CertificateImpl(byte[] buf)
throws java.security.cert.CertificateException
```

Unmarshals a certificate from its encoded form, parsing the BER encoded bytes. This form of constructor is used by agents which need to examine and use certificate contents. That is, this is one of the more commonly used constructors.

X509CertificateImpl

```
public X509CertificateImpl(byte[] buf,
int offset,
int len)
throws java.security.cert.CertificateException
```

Instantiates an X509Certificate with input certificate data

Parameters:

buff - - the certificate data buffer

offset - - offset of the data buffer

len - - the data buffer length

Methods

X509Certificate

```
public void X509Certificate(java.io.InputStream in)
throws java.io.IOException
```

Instantiates an X509Certificate object, and initializes it with the data read from the input stream inStream. The implementation is provided by the class specified as the value of the cert.provider.x509v1 property in the security properties file.

> **Note:** Only one DER-encoded certificate is expected to be in the input stream.

Parameters:

DER - encoded InputStream data

Since:

1.0

decode

```
public void decode(java.io.InputStream in)
throws java.io.IOException
```

Decodes the input stream data and instantiates an X509Certificate object, and initializes it with the data read from the input stream inStream.

Parameters:

DER - encoded InputStream data

Since:

1.0

equals

```
public boolean equals(java.lang.Object obj)
```

Checks for the equility

Parameters:

Certificate - object

Overrides:

equals in class java.security.cert.Certificate

Since:

1.0

checkValidity

```
public void checkValidity()
throws java.security.cert.CertificateExpiredException,
java.security.cert.CertificateNotYetValidException
```

Checks for the validity of the certificate with current time

Throws:

CertificateExpiredException, - CertificateNotYetValidException

Overrides:

checkValidity in class java.security.cert.X509Certificate

Since:

1.0

checkValidity

```
public void checkValidity(java.util.Date date)
```

```
throws java.security.cert.CertificateExpiredException,
java.security.cert.CertificateNotYetValidException
```

Checks for the validity of the certificate with given time

Throws:

CertificateExpiredException, - CertificateNotYetValidException

Overrides:

checkValidity in class java.security.cert.X509Certificate

Since:

1.0

verify

```
public void verify(java.security.PublicKey key)
throws java.security.cert.CertificateException
```

Checks for the validity of the input public key for this certificate

Parameters:

key - -PublicKey

Throws:

CertificateException -

Overrides:

verify in class java.security.cert.Certificate

Since:

1.0

verify

```
public void verify(java.security.PublicKey key,
java.lang.String sigProvider)
throws java.security.cert.CertificateException
```

Checks for the validity of the input public key for this certificate

Parameters:

key - - PublicKey

sigProvider - - Provider

Throws:

CertificateException -

Overrides:

verify in class java.security.cert.Certificate

Since:

1.0

getSubjectDN

```
public java.security.Principal getSubjectDN()
```

Returns the certificate subject DN

Returns:

Subject name

Overrides:

getSubjectDN in class java.security.cert.X509Certificate

Since:

1.0

See Also:

java.security.Principal

getIssuerDN

```
public java.security.Principal getIssuerDN()
```

Returns the certificate issuer DN

Returns:

issuer name

Overrides:

getIssuerDN in class java.security.cert.X509Certificate

Since:

1.0

See Also:

java.security.Principal

getVersion

```
public int getVersion()
```

Returns the certificate version

Returns:

version value

Overrides:

getVersion in class java.security.cert.X509Certificate

Since:

1.0

getSerialNumber

```
public java.math.BigInteger getSerialNumber()
```

Gets the serialNumber value from the certificate. The serial number is an integer assigned by the certification authority to each certificate. It must be unique for each certificate issued by a given CA (i.e., the issuer name and serial number identify a unique certificate).

Returns:

the serial number.

Overrides:

getSerialNumber in class java.security.cert.X509Certificate

Since:

1.0

getSigAlgName

```
public java.lang.String getSigAlgName()
```

Returns the signature algorithm

Returns:

signature algorithm

Overrides:

getSigAlgName in class java.security.cert.X509Certificate

Since:

1.0

getSigAlgOID

```
public java.lang.String getSigAlgOID()
```

Returns the signature algorithm OID

Returns:

signature algorithm OID

Overrides:

getSigAlgOID in class java.security.cert.X509Certificate

Since:

1.0

getSigAlgParams

```
public byte[] getSigAlgParams()
```

Returns the signature algorithm paramaters

Returns:

signature algorithm paramaters

Overrides:

getSigAlgParams in class java.security.cert.X509Certificate

Since:

1.0

getNotBefore

public java.util.Date getNotBefore()

Returns the date when this certificate will be vaild

Returns:

date when this certificate will be valid

Overrides:

getNotBefore in class java.security.cert.X509Certificate

Since:

1.0

getNotAfter

```
public java.util.Date getNotAfter()
```

Returns the date when this certificate will expired

Returns:

date when this certificate will expired

Overrides:

getNotAfter in class java.security.cert.X509Certificate

Since:

1.0

getEncoded

```
public byte[] getEncoded()
throws java.security.cert.CertificateEncodingException
```

Returns the encoded certificate

Returns:

byte array data of this certificate

Overrides:

getEncoded in class java.security.cert.Certificate

Since:

1.0

getPublicKey

```
public java.security.PublicKey getPublicKey()
```

Returns the encoded certificate

Returns:

public key of this certificate

Overrides:

getPublicKey in class java.security.cert.Certificate

Since:

1.0

See Also:

PublicKey

hashCode

```
public int hashCode()
```

Returns the public key of this certificate

Returns:

returns the hash coded

Overrides:

hashCode in class java.security.cert.Certificate

Since:

1.0

toString

```
public java.lang.String toString()
```

Returns information about this certificate

Returns:

information of this certificate in string format

Overrides:

toString in class java.security.cert.Certificate

Since:

1.0

getSubjectUniqueID

```
public boolean[] getSubjectUniqueID()
```

Overrides:

getSubjectUniqueID in class java.security.cert.X509Certificate

getSignature

```
public byte[] getSignature()
```

Overrides:

getSignature in class java.security.cert.X509Certificate

getBasicConstraints

```
public int getBasicConstraints()
```

Overrides:

getBasicConstraints in class java.security.cert.X509Certificate

getIssuerUniqueID

public boolean[] getIssuerUniqueID()

Overrides:

getIssuerUniqueID in class java.security.cert.X509Certificate

getKeyUsage

```
public boolean[] getKeyUsage()
```

Overrides:

getKeyUsage in class java.security.cert.X509Certificate

getTBSCertificate

```
public byte[] getTBSCertificate()
```

throws java.security.cert.CertificateEncodingException

Overrides:

getTBSCertificate in class java.security.cert.X509Certificate

getCriticalExtensionOIDs

```
public java.util.Set getCriticalExtensionOIDs()
```

Overrides:

getCriticalExtensionOIDs in class java.security.cert.X509Certificate

getExtensionValue

```
public byte[] getExtensionValue(java.lang.String oid)
```

Overrides:

getExtensionValue in class java.security.cert.X509Certificate

getNonCriticalExtensionOIDs

```
public java.util.Set getNonCriticalExtensionOIDs()
```

Overrides:

getNonCriticalExtensionOIDs in class java.security.cert.X509Certificate

hasUnsupportedCriticalExtension

```
public boolean hasUnsupportedCriticalExtension()
```

Overrides:

hasUnsupportedCriticalExtension in class java.security.cert.X509Certificate

Glossary

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

authentication

The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to allowing access to resources in a system.

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.

certificate

A certificate is created when an entity's public key is signed by a trusted identity, that is, a certificate authority. This 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. The important property of a cryptographic checksum is that, without knowing the secret key, a malicious interceptor has only an infinitesimally small chance of being able to construct an altered message with a valid corresponding checksum.

Cipher Block Chaining

A feedback mechanism that protects against block replay attacks by making the encryption of a block dependent on all blocks that precede it.

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.

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 act of writing and deciphering secret code resulting in secure messages.

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.

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 naming context

A subtree which is of significance within a directory server. It is usually the top of some organizational subtree. Some directories only allow one such context which is fixed; others allow none to many to be configured by the directory administrator.

encryption

The process of disguising a message rendering it unreadable to anybody but the intended recipient.

enterprise domain

A group of databases and **enterprise roles**.

enterprise role

Access privileges assigned **enterprise users**.

enterprise user

A user defined and managed in a directory.

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.

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.

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.

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.

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.

MD5

An algorithm that assures data integrity by generating a unique, 128-bit cryptographic message digest value from the contents of a file. If as little as a single bit value in the file is modified, the MD5 checksum for the file will change. Forgery of a file in a way that will cause MD5 to generate the same result as that for the original file is considered extremely difficult.

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**.

Net8

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. Net8 supports distributed processing and distributed database capability. Net8 is an "open system" because it is independent of the communication protocol, and users can interface Net8 to many network environments.

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.

Oracle Context

An entry in the directory called `cn=OracleContext`, under which all Oracle software relevant information is kept, including entries for Net8 directory naming and **enterprise user** security.

principal

A uniquely-identified client or server. A Kerberos object, consisting of *kservice/kinstance@REALM*. See also *kservice*, *kinstance*, and *realm*.

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 the recipient's private key.

public/private key pair

A mathematically related set of two numbers where one is called the private key and the other is called the public key. Public keys are typically made widely available, while a private key is available only to the owner. Data encrypted with a public key can be decrypted with its associated private key and vice versa. However, data encrypted with a public key cannot be decrypted with the same public 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 but that have the same name are unique.

Secure Hash Algorithm

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).

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 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.

session key

A key shared by at least two parties (usually a client and a server).

server

A provider of a service.

service principal

See **principal**.

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.

SHA

See **Secure Hash Algorithm**.

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.

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. 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.

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.

trusted certificate

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.

trust point

See **trusted certificate**.

wallet

An abstraction 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 Resource Locator

A directory path that provides all the necessary information to locate a particular wallet.

WRL

See **Wallet Resource Locator**.

X.509

The public keys can be signed in various data formats. The X.509 format from ISO is one such popular format.

Index