Oracle® Advanced Security

Administrator's Guide

Release 8.1.7

September 2000

Part No. A85430-01

ORACLE®

Oracle Advanced Security Administrator's Guide, Release 8.1.7

Part No. A85430-01

Primary Author: Mike Cowan

Contributing Authors: Cynthia Kibbe, Rita Moran, Richard Smith, Deborah Steiner, Ted Burroughs

Contributors: Kristy Browder, Mary Ann Davidson, Quan Dinh, Gary Gilchrist, Duncan Harris, Michael Hwa, Adam Jacobs, Lakshmi Kethana, Andrew Koyfman, Van Le, Nina Lewis, Valarie Moore, Andy Philips, Ramana Turlapati, Marcie Young

Contents

4 Configuring RADIUS Authentication

5 Configuring CyberSafe Authentication

6 Configuring Kerberos Authentication

7 Configuring SecurID Authentication

8 Configuring Identix Biometric Authentication

9 Configuring Secure Socket Layer Authentication

10 Configuring Entrust-Enabled SSL Authentication

11 Configuring Multiple Authentication Methods

Part IV Oracle DCE Integration

12 Overview of Oracle DCE Integration

13 Configuring DCE for Oracle DCE Integration

14 Configuring Oracle8*i* for Oracle DCE Integration

18 Using Oracle Wallet Manager

Part VI Appendices

A Data Encryption and Integrity Parameters

B Authentication Parameters

C Integrating Authentication Devices Using RADIUS

D Oracle Advanced Security FIPS 140-1 Settings

List of Figures

List of Tables

Send Us Your Comments

Oracle Advanced Security Oracle Advanced Security Administrator's Guide, Release 8.1.7

Part No. A85430-01

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

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

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

- Electronic mail: infodev@us.oracle.com
- FAX: (650) 506-7228 Attn: Oracle 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, telephone number, and (optionally) electronic mail address.

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

Preface

Welcome to the Oracle Advanced Security Administrator's Guide for Release 8.1.7 of Oracle Advanced Security (formerly Oracle Advanced Networking Option).

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

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

See Also: Related Documents on page -xxv

Intended Audience

This guide is intended for users or systems professionals involved with the implementation, configuration, and administration of Oracle Advanced Security including:

- Implementation consultants
- System administrators
- Security administrators

Structure

This guide is organized into the following parts:

- Part I: Introduction
- Part II: Encryption, Integrity, and JDBC

- Part III: Configuring Authentication Methods
- Part IV: Oracle DCE Integration
- Part V: Oracle8i Enterprise User Security
- Part VI: Appendices

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

Part I: Introduction

Chapter 1, Introduction to Oracle Advanced Security

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

Part II: Encryption, Integrity, and JDBC

Chapter 2, Configuring Data Encryption and Integrity

This chapter describes how to configure data encryption and integrity within an existing Net8 Release 8.1.7 network.

Chapter 3, Thin JDBC 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.

Part III: Configuring Authentication Methods

Chapter 4, Configuring RADIUS Authentication

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

Chapter 5, Configuring CyberSafe Authentication

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

Chapter 6, Configuring Kerberos Authentication

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

Chapter 7, Configuring 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, known limitations, and troubleshooting information.

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 Secure Socket Layer Authentication

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

Chapter 10, Configuring Entrust-Enabled SSL Authentication

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

Chapter 11, Configuring Multiple Authentication Methods

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

Part IV: Oracle DCE Integration

Chapter 12, Overview of Oracle DCE Integration

This chapter provides a brief discussion of Open Software Foundation (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 Oracle8i 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 V: Oracle8i Enterprise User Security

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, Using 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 VI: Appendices

Appendix A, Data Encryption and Integrity Parameters

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

Appendix B, Authentication Parameters

This appendix describes Oracle Advanced Security authentication configuration file parameters.

Appendix C, Integrating Authentication Devices Using RADIUS

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

Appendix D, Oracle Advanced Security FIPS 140-1 Settings

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

Appendix E, 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.

Related Documents

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:

- ❏ See the following Oracle documents for information that applies across platforms:
  - *ACE/Server Administration Manual, Release 3.3,* from Security Dynamics
  - *ACE/Server Client for UNIX, Version 2.0,* from Security Dynamics

- *ACE/Server Installation Manual, Release 3.3,* from Security Dynamics
- *Net8 Administrator's Guide*
- *Oracle8i Distributed Database Systems*
- Oracle8i Enterprise JavaBeans Developer's Guide and Reference
- Oracle8i JDBC Developer's Guide and Reference
- Oracle Internet Directory Administrator's Guide

❏ For information about roles and privileges, see:

- *Oracle8i Administrator's Guide*

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

- *RADIUS Administrator's Guide*
- *Steel-Belted RADIUS v2.11 (Funk Software)*
- *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*
- C*yberSafe TrustBroker Client v1.2*
- *CyberSafe TrustBroker Server v1.2*

❏ For information about MIT Kerberos see:

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

❏ For information about Entrust / PKI see:

- *Entrust/PKI 5.0.2 for Oracle*
- *Administering Entrust/PKI 5.0 on UNIX*

- 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 TouchSafe II User's Guide
- Server-side documentation:
  - *Identix TouchSafe II System Administrator's Guide*

Abbreviations and Acronyms

Table 0–1 defines abbreviations and acronyms used in this document:

Table 0–1 Abbreviations and Acronyms

3DES	A version of the DES encryption algorithm that provides triple-encryption; see Triple-DES.
ACL	Access Control List
CA	Certificate Authority
CBC	Outer Cypher-Block-Chaining mode
CDS	Cell Directory Service
CORBA	Common Object Request Broker Architecture
DBCA	Oracle Database Configuration Assistant
DCE	Distributed Computing Environment
DES	Data Encryption Standard (U.S.)
DES40	Data Encryption Standard with 40-bit encryption keys
DES56	Data Encryption Standard with 56-bit encryption keys
DIT	Directory Information Tree
DN	Distinguished Name
ESM	Oracle Enterprise Security Manager
FIPS	Federal Information Processing Standard
GSSAPI	Generic Security Services Application Programming Interface
IIOP	Internet Inter-ORB Protocol
ISM	Bull Integrated System Management
ISP	Internet Service Provider
JDBC	Java Database Connectivity
JDK	Java Development Kit
JRE	Java Runtime Environment
LAN	Local Area Network
LDAP	Lightweight Directory Access Protocol
MD4	Message Digest 4; a xxx-bit encryption algorithm.

Table 0–1 Abbreviations and Acronyms

MD5	Message Digest 5; a 128-bit encryption algorithm; successor to MD4.
Net8CA	Oracle Net8 Configuration Assistant
OCI	Oracle Call Interface
OID	Oracle Internet Directory
OSF	Open Software Foundation
PIN	Personal Identification Number
PKE	Public Key Encoding
PKI	Public Key Infrastructure
RADIUS	Remote Authentication Dial-In User Service
RC4	A public-key algorithm of RSA
RSA	RSA Data Security, Inc.; refers to the RSA encryption module
SASL	Simple Authentication and Security Layer
SHA	Secure Hash Algorithm
SSL	Secure Sockets Layer
SSO	Single Sign-on
Triple-DES	A version of the DES encryption algorithm that provides triple-encryption; see 3DES
WAN	Wide Area Network

Conventions

The following syntax conventions are used in this guide:

...	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
[ ]	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.
Boldface text	Boldface text indicates a term defined in the glossary.
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.
Punctuation	Punctuation other than brackets and vertical bars must be entered as shown.
UPPERCASE	Uppercase characters within the text represent parameters.

Part I

Introduction

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

- Chapter 1, Introduction to Oracle Advanced Security

1

Introduction to Oracle Advanced Security

This chapter introduces Oracle Advanced Security and describes its features. These features are available to database and related products that interface with Oracle Net8, including Oracle8*i*, Oracle Designer, and Oracle Developer.

This chapter contains the following sections:

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

About Oracle Advanced Security

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

This section contains the following topics:

- Security in an Intranet or Internet Environment
- Security Threats

Security in an Intranet or Internet Environment

Oracle databases power the largest and most popular web sites on the Internet. In record numbers, organizations throughout the world are deploying distributed databases and client/server applications based on Oracle8*i* and Net8. This proliferation of distributed computing is matched by an increase in the amount of information that organizations place on computers. Employee and financial records, customer orders, product information, and other sensitive data have moved from filing cabinets to file structures. The volume of sensitive information on the web has thus increased the value of data that can be compromised.

Security Threats

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

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

Eavesdropping and Data Theft

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

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

Data Tampering

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

Falsifying User Identities

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

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

Password-Related Threats

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

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

- They may select easy-to-guess passwords—such as a name, fictional character, or a word found in a dictionary. All of these passwords are vulnerable to **dictionary attack**s.
- They may also choose to standardize passwords so that they are the same on all machines or web sites. This results in a potentially large exposure in the event of a compromised password. They can also use passwords with slight variations that can be easily derived from known passwords.
- Users with complex passwords may write them down where an attacker can easily find them, or they may just forget them—requiring costly administration and support efforts.

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

Oracle Advanced Security Features

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

For example, you can configure either Net8 native encryption or Secure Socket Layer (SSL) for data privacy. Oracle Advanced Security also provides the choice of several strong authentication methods, including Kerberos, smart cards, and digital certificates.

Oracle Advanced Security features are described in the following sections:

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

Data Privacy

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

- RSA Encryption
- DES Encryption
- Triple-DES Encryption

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

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

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

RSA Encryption

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

DES Encryption

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

Triple-DES Encryption

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

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

Federal Information Processing Standard

Oracle Advanced Security Release 8.1.7 has been validated under U.S. Federal Information Processing Standard 140-1 (FIPS) at the Level 2 security level. This provides independent confirmation that Oracle Advanced Security conforms to

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

See Also:

- Chapter 2, Configuring Data Encryption and Integrity
- Appendix A, Data Encryption and Integrity Parameters

Data Integrity

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

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

- ❑ Data modification
- ❑ Deleted packets
- ❑ Replay attacks

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

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

Authentication

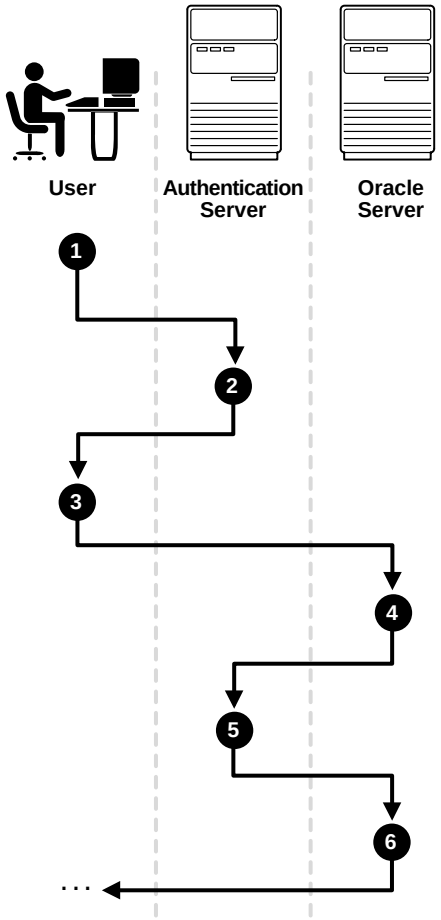
Authenticating user identity is imperative in distributed environments, without which there can be little confidence in network security. Passwords are the most common **authentication** method, and Oracle Advanced Security provides enhanced user authentication through several third-party authentication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

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

How Centralized Network Authentication Works

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

Figure 1–1 How a Network Authentication Service Authenticates a User



1. A user (client) requests authentication services, providing some identification—such as a token or password—proving that the user is who he or she claims to be.
2. The authentication server validates the user's identify, and 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.

Supported Authentication Methods

Oracle Advanced Security supports the following authentication methods:

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

Secure Sockets Layer

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

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

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

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

Entrust/PKI

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

Remote Authentication Dial-In User Service

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

environment to enable use of any authentication method that supports the RADIUS protocol. RADIUS can be used with a variety of authentication mechanisms, including token cards, smart cards, and biometrics.

Kerberos and CyberSafe

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

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

Smart Cards

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

Smart cards provide the following benefits:

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

Token Cards

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

Token cards provide the following benefits:

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 must have the token card as well as the personal identification number (PIN) required to operate it. This is called two-factor authentication.
Ease of use	Users need only remember a PIN, instead of multiple passwords.
Enhanced accountability	Token cards provide a stronger authentication mechanism; users are thus more accountable for their actions.

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

Biometric Authentication

Identix Biometric Authentication (Identix or RADIUS-compliant) is used on both Oracle clients and 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 Integrated System Management

Bull Integrated System Management (ISM), from Bull Worldwide Information Systems, provides a variety of management tools for system administrators. This authentication method is available on the AIX operating system only.

See Also: AIX documentation

Single Sign-On

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

Oracle Advanced Security single sign-on authenticates the user once upon initial connection, with strong authentication occurring transparently in subsequent connections to other databases or services. Using single sign-on, users can access multiple accounts and applications with a single password. Oracle Advanced Security supports many forms of single sign-on, including Kerberos and CyberSafe.

Oracle Advanced Security also provides SSL-based single sign-on for Oracle users by integrating with **LDAP** v3-compliant directory services. The combination of integrated directory services and Oracle's PKI implementation enable SSL-based single sign-on to Oracle*8*i databases. Single sign-on lets users be authenticated once, with subsequent connections relying on the user's digital certificate.

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

Authorization

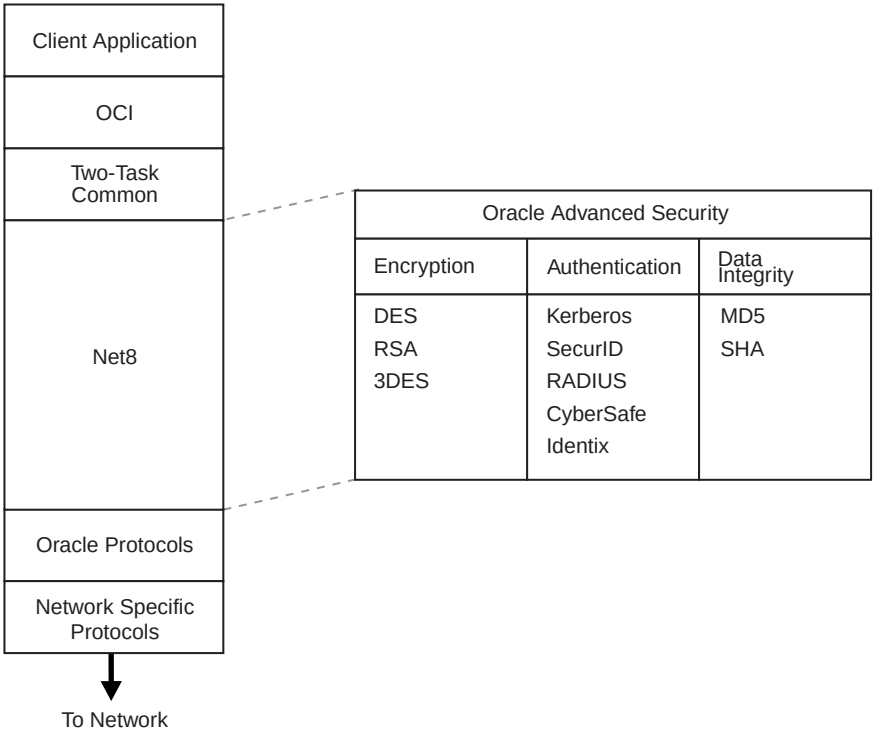
User authorization, a function of Oracle8*i* roles and privileges, is significantly enhanced by using the authentication methods supported by Oracle Advanced Security. For example, on certain operating systems, such as Solaris, Oracle Advanced Security supports authorization with DCE.

Authorizations are also provided by Oracle Advanced Security Enterprise User Security (See: Chapter 17, Managing Enterprise User Security). Oracle Advanced Security can integrate with LDAP version 3-compliant directories to centrally manage users and authorizations. Your Oracle Advanced Security license entitles you to deploy Oracle Internet Directory for user management as well as authorization storage and retrieval. You must license Oracle Internet Directory separately if you use it for additional purposes.

Oracle Advanced Security Architecture

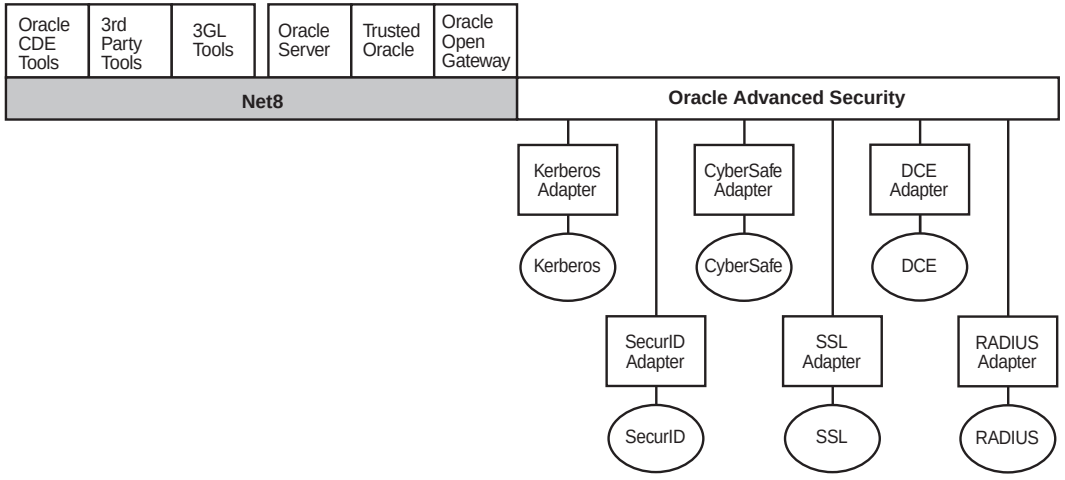
Oracle Advanced Security is an add-on product that complements a standard Oracle server or client installation. Figure 1–2 shows the Oracle Advanced Security architecture within an Oracle networking environment.

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



Oracle Advanced Security supports authentication through adapters that are 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



See Also: *Net8 Administrator's Guide*, for more information about stack communications in an Oracle networking environment

Secure Data Transfer Across Network Protocol Boundaries

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

System Requirements

Oracle Advanced Security is an add-on product bundled with the standard Oracle Net8 Server or Net8 Client. It must be purchased and installed on both the client and the server.

Oracle Advanced Security release 8.1.7 requires Net8 release 8.1.7 and supports Oracle8*i* Enterprise Edition. Table 1–1 lists additional system requirements.

Note: Oracle Advanced Security is not available with Oracle 8*i* Standard Edition.

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 an operating system 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.
Entrust/PKI	■ Entrust IPSEC Negotiator Toolkit Release 5.0.2 ■ Entrust/Toolkit Server Login Release 5.0.2

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 Microsoft Windows. *The portions of these products that use Oracle Display Manager (ODM) do not take advantage of Oracle Advanced Security, since ODM does not use Net8.*

Part II

Encryption, Integrity, and JDBC

This part describes how to configure data encryption and integrity for your existing Net8 network, and the Java implementation of Oracle Advanced Security. It contains the following chapters:

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

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

2

Configuring Data Encryption and Integrity

This chapter describes how to configure native Net8 data **encryption** and **integrity** for Oracle Advanced Security, in the following sections:

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

Oracle Advanced Security Encryption

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

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

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

Overview

The purpose of a secure cryptosystem is to convert **plaintext** data into unintelligible **ciphertext** based on a key, in such a way that it is very hard (computationally infeasible) to convert ciphertext back into its corresponding plaintext without knowledge of the correct key. In a symmetric cryptosystem, the same key is used both for encryption and decryption of the same data. Oracle Advanced Security provides the DES, 3DES, and RC4 symmetric cryptosystems for protecting the confidentiality of Net8 traffic.

DES Algorithm for Standards-Based Encryption

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

Triple-DES Support

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

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

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

DES40 Algorithm

The DES40 algorithm, available in every release of Oracle Advanced Security, Oracle Advanced Networking Option, and Secure Network Services, is a variant of DES in which the secret key is preprocessed to provide 40 effective key bits. It was designed to provide DES-based encryption to customers outside the U.S. and Canada at a time when the U.S. export laws were more restrictive. Now, in Oracle Advanced Security Release 8.1.7, DES40, DES, and 3DES are all available for export. DES40 is still supported to provide backward-compatibility for international customers.

RSA RC4 Algorithm for High Speed Encryption

The RC4 algorithm, developed by RSA Data Security Inc., has become the international standard for high-speed data encryption. Despite ongoing attempts by cryptographic researchers to crack it, the only known method of unauthorized decryption is brute-force. RC4 is a variable key-length 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.

Oracle Advanced Security Release 8.1.7 provides an RC4 implementation with 40-bit, 56-bit, 128-bit, and 256-bit key lengths. This provides backward-compatibility and strong encryption, with no material performance compromise.

See Also:

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

Oracle Advanced Security Data Integrity

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

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

Data Integrity Algorithms Supported

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

See Also:

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

Diffie-Hellman Based Key Management

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

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

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

Authentication Key Fold-in

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

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

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

Configuring Data Encryption and Integrity

This section describes how to configure Oracle Advanced Security native Net8 encryption and integrity, and presumes the prior installation of Oracle Net8.

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

- Activating Encryption and Integrity
- Negotiating Encryption and Integrity
- Setting the Encryption Seed
- Configuring Encryption and Integrity Parameters Using Net8 Assistant

> **See Also:** Chapter 9, Configuring Secure Socket Layer Authentication, to configure the SSL feature for encryption, integrity, and authentication

Activating Encryption and Integrity

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

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

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

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

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

See Also: Appendix A, Data Encryption and Integrity Parameters

Negotiating Encryption and Integrity

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

- REJECTED
- ACCEPTED
- REQUESTED
- REQUIRED

The default value for each of the parameters is ACCEPTED.

REJECTED

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

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

REQUIRED

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

Table 2–1 Encryption and Data Integrity Negotiation

Server \ Client	REJECTED	ACCEPTED	REQUESTED	REQUIRED
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 ACCEPTED is set to ON for both the client and the server.

Setting the Encryption Seed

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

Configuring Encryption and Integrity Parameters Using Net8 Assistant

You can set up or change encryption and integrity parameter settings using Net8 Assistant. This section describes the following topics:

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

See Also:

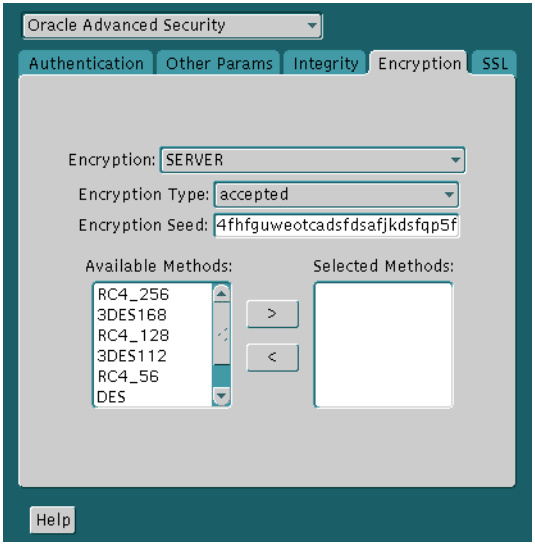
- Appendix A, Data Encryption and Integrity Parameters, for valid encryption algorithms
- Net8 Assistant online help, for more detailed configuration information

Configuring Encryption on the Client and the Server

To configure encryption on the client and on the server:

1. Start 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 window, expand Local > Profile.
3. From the list in the right window pane, select Oracle Advanced Security.

The Oracle Advanced Security tabbed window appears:



4. Choose the Encryption tab.
5. Depending upon which system you are configuring, in the Encryption list, select CLIENT or SERVER.
6. From the Encryption Type list, select one of the following:
   - REQUESTED
   - REQUIRED
   - ACCEPTED
   - REJECTED
7. In the Encryption Seed field, enter between 10 and 70 random characters; the encryption seed for the client should not be the same as that for the server.
8. Select an encryption algorithm in the Available Methods list. Move it to the Selected Methods list by choosing the right arrow [>]. Repeat for each additional method you want to use.
9. Choose File > Save Network Configuration; the `sqlnet.ora` file is updated.
10. Repeat this procedure to configure encryption on the other system. The `sqlnet.ora` file on the two systems should contain the following entries:

- On the server:

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

- On the client:

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

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

Table 2–2 Valid Encryption Algorithms

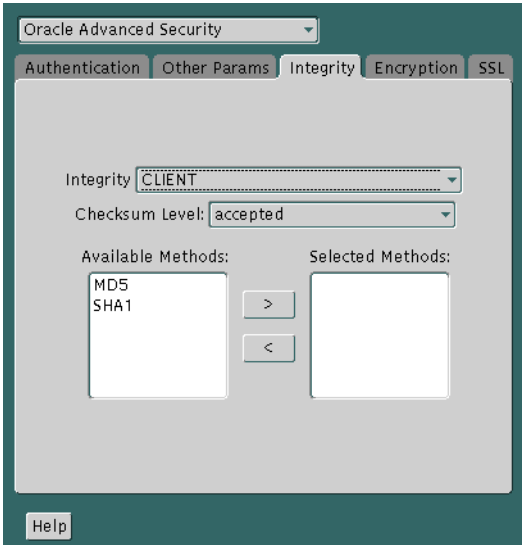
Algorithm Name	Legal Value
RC4 256-bit key	RC4_256
RC4 128-bit key	RC4_128
RC4 56-bit key	RC4_56
RC4 40-bit key	RC4_40
3-key 3DES	3DES168
2-key 3DES	3DES112
DES 56-bit key	DES
DES 40-bit key	DES40

Configuring Integrity on the Client and the Server

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

1. Start 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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Integrity tab.
5. Depending upon which system you are configuring, choose the Server or Client check box.
6. From the Checksum Level list, select one of the following checksum level values:
   - REQUESTED
   - REQUIRED
   - ACCEPTED
   - REJECTED
7. Choose File > Save Network Configuration; the `sqlnet.ora` file is updated.
8. Repeat this procedure to configure integrity on the other system. The `sqlnet.ora` file on the two systems should contain the following entries:
   - On the server:

     ```
     SQLNET.CRYPTO_CHECKSUM_SERVER = [accepted | rejected | requested |
     ```

required]
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (valid_*crypto_checksum_algorithm*
[,valid_crypto_checksum_algorithm])

- On the client:

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

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

Table 2–3 Valid Integrity Algorithms

Algorithm Name	Legal Values
MD5	MD5
SHA-1	SHA1

Note: Under Oracle Advanced Security Release 8.1.7, two integrity algorithms are available: MD5 and SHA-1.

MD5 can be enabled or disabled by using the Oracle Advanced Security tabbed window displayed under Configuring Integrity on the Client and the Server on page 2-12.

To enable SHA-1, you must edit the `sqlnet.ora` file by adding SHA1 as a `valid_crypto_checksum_algorithm`.

3

Thin JDBC Support

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

This chapter contains the following topics:

- About the Java Implementation
- Configuration Parameters

> **See Also:** *Oracle8i JDBC Developer's Guide and Reference*, for information about JDBC, including examples

About the Java Implementation

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

This section contains the following topics:

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

Java Database Connectivity Support

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

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

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

Securing Thin JDBC

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

- Data encryption
- Data integrity checking
- Secure connections from Thin JDBC clients to the Oracle RDBMS
- Ability for developers to build applets that transmit data over a secure communication channel
- Secure connections from middle tier servers with Java Server Pages (JSP) to the Oracle RDBMS
- Secure connections from 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. It does not support Oracle Advanced Security SSL implementation, nor does it support third party authentication features such as RADIUS, Kerberos, and SecurID. However, the Oracle JDBC OCI (thick) driver support is the same as thick client support, where all Oracle Advanced Security features are implemented.

Oracle Advanced Security continues to encrypt and provide integrity checking of 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:

- RC4_256
- RC4_128
- RC4_56
- RC4_40
- DES56
- DES40

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

In addition, this implementation provides data integrity checking for Thin JDBC with the following algorithms:

- MD5
- SHA

Implementation Overview

On the server side, the negotiation of algorithms and the generation of keys function exactly the same as Oracle Advanced Security native encryption. This 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.

Obfuscation

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

Java byte code obfuscation is a process 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 possible to decompile and read non-obfuscated Java code, obfuscated Java code is sufficiently difficult to decompile to satisfy U.S. government export controls.

Configuration Parameters

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

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

Client Encryption Level

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

Client Encryption Selected List

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

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

Client Integrity Level

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

Client Integrity Selected List

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

Part III

Configuring Authentication Methods

This part describes how to configure authentication methods into your existing Net8 network. It contains the following chapters, each of which describes a particular authentication method supported by Oracle Advanced Security Release 8.1.7:

- 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 Secure Socket Layer Authentication
- Chapter 10, Configuring Entrust-Enabled SSL Authentication
- Chapter 11, Configuring Multiple Authentication Methods

4

Configuring RADIUS Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle8*i*, or for the Oracle8*i* server, for use with RADIUS (Remote Authentication Dial-In User Service).

This chapter contains the following sections:

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

RADIUS Overview

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

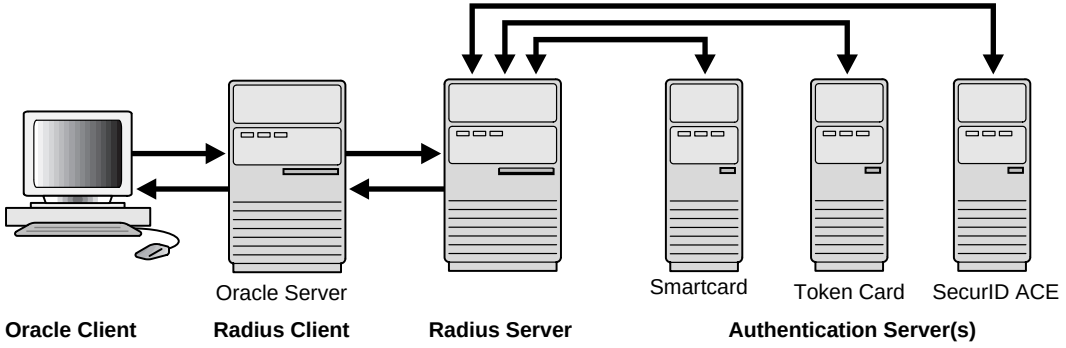
You can enable the network to use any authentication method that supports the RADIUS standard, including token cards and smart cards, by installing and configuring the RADIUS protocol. Moreover, when you use RADIUS, you can change the authentication method without modifying either the Oracle client or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

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

- ❏ Looks up the user's security information.
- ❏ Passes authentication and authorization information between the appropriate authentication server or servers and the Oracle database server.
- ❏ Grants the user access to the Oracle database server.
- ❏ Logs session information, including when, how often, and for how long the user was connected to the Oracle database server.

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

Figure 4–1 RADIUS in an Oracle Environment



The Oracle database server acts as the RADIUS client, passing information between the Oracle client and the RADIUS server. Similarly, the RADIUS server passes information between the Oracle database server and the appropriate authentication

server or servers. To ensure transmission integrity of the authentication information, RADIUS converts it to a hash value. The authentication components are listed in Table 4–1:

Table 4–1 RADIUS Authentication Components

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

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

> **See Also:** *Net8 Administrator's Guide,* for information about the `sqlnet.ora` file

RADIUS Authentication Modes

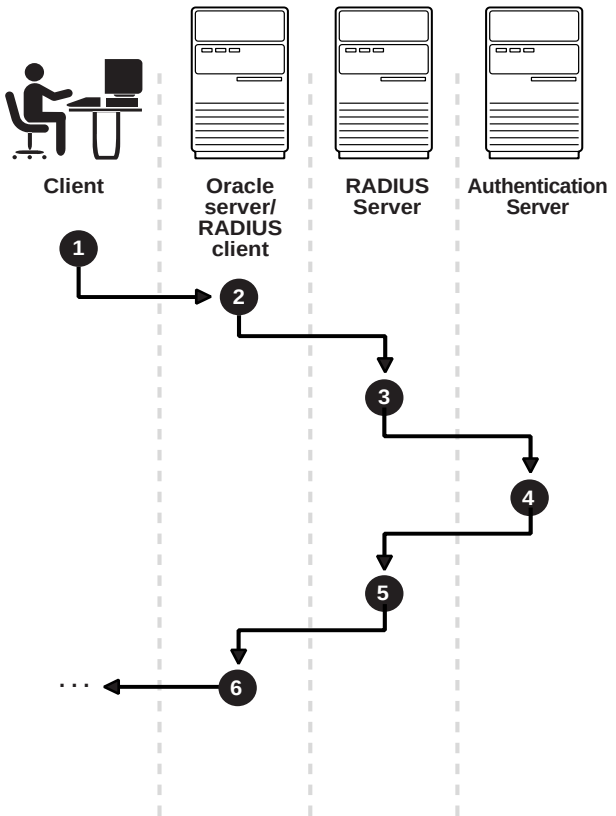
User authentication can take place in either of two ways:

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

Synchronous Authentication Mode

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

Figure 4–2 Synchronous Authentication Sequence



1. A user logs in by entering a connect string, passcode, or other value. The client machine passes this data to the Oracle database server.
2. The Oracle database server, acting as the RADIUS client, passes the data from the Oracle client to the RADIUS server.
3. The RADIUS server passes the data to the appropriate authentication server, such as Smart card or SecurID ACE for validation.
4. The authentication server sends 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 database server / RADIUS client.
6. The Oracle database 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 that displays a dynamic number that changes every sixty seconds. To gain access to the Oracle database server/RADIUS client, the user enters a valid passcode that includes both a personal identification number (PIN) and the dynamic number currently displayed on the user's SecurID card. The Oracle database server passes this authentication information from the Oracle client to the RADIUS server, which in this case is the authentication server for validation. Once the authentication server (Security Dynamics ACE/Server) validates the user, it sends an "accept" packet to the Oracle database server, which, in turn, passes it to the Oracle client. The user is now authenticated and able to access the appropriate tables and applications.

See Also:

- Chapter 1, Introduction to Oracle Advanced Security, and Chapter 7, Configuring SecurID Authentication, for more information about SecurID token cards
- Documentation provided by Security Dynamics

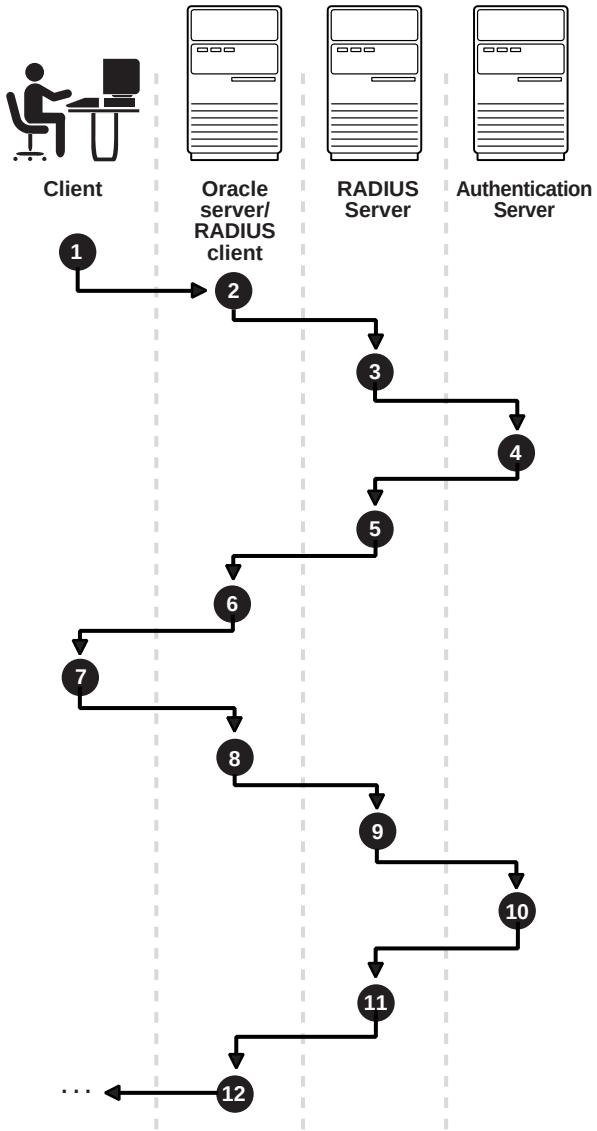
Challenge-Response (Asynchronous) Authentication Mode

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

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

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

Figure 4–3 Asynchronous Authentication Sequence



1. A user seeks a connection to the Oracle database server. The client machine passes the data to the Oracle database server.
2. The Oracle database server, acting as the RADIUS client, passes the data from the Oracle client to the RADIUS server.
3. The RADIUS server passes the data to the appropriate authentication server, such as a smart card, SecurID ACE, or token card.
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 database server / RADIUS client.
6. The Oracle database 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 database server / RADIUS client.
8. The Oracle database server / RADIUS client sends the user's response to the RADIUS server.
9. The RADIUS server passes the user's response to the appropriate authentication server for validation.
10. The authentication server sends back to the RADIUS server either an Access Accept or an Access Reject message.
11. The RADIUS server passes the response to the Oracle database server / RADIUS client.
12. The Oracle database server / RADIUS client passes the response to the Oracle client.

Example: Asynchronous Authentication with Smart Cards

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

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

Example: Asynchronous Authentication with ActivCard Tokens

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

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

Enabling RADIUS Authentication and Accounting

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

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

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

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

> **See:** platform-specific installation documentation for Oracle8*i*, for information about installing Oracle Advanced Security and the RADIUS adapter

Task 2: Configure RADIUS Authentication

This task includes the following steps:

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

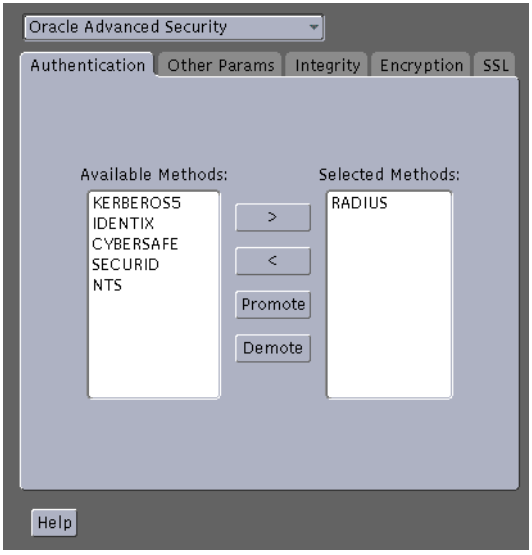
Unless otherwise indicated, perform these configuration tasks by using the Net8 Assistant or by using any text editor to modify the `sqlnet.ora` file.

Step 1: Configure RADIUS on the Oracle Client

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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. From the Available Methods list, select RADIUS.
6. Choose the right-arrow [>] to move RADIUS to the Selected Methods list. Move any other methods you want to use in the same way.
7. Arrange the selected methods in order of required usage by selecting a method in the Selected Methods list, and clicking Promote or Demote to position it in the list. For example, put RADIUS at the top of the list for it to be the first service used.
8. Choose File > Save Network Configuration.

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

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

Step 2: Configure RADIUS on the Oracle Database Server

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

Create the RADIUS Secret Key File on the Oracle Database Server

1. Obtain the RADIUS secret key from the RADIUS server. For each RADIUS client, the administrator of the RADIUS server creates a shared secret key, which can be as simple as "test123".
2. On the Oracle database server, create a directory `$ORACLE_HOME/network/security` on UNIX or *`ORACLE_HOME`*`\network\security` on Windows NT.
3. Create the file `radius.key` to hold the shared secret from the RADIUS server. Place the file in the directory you just created, namely, `$ORACLE_HOME/network/security` on UNIX or *`ORACLE_HOME`*`\network\security` on Windows NT.
4. Copy the shared secret key and paste it (and nothing else) into the `radius.key` file created on the Oracle database server.

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

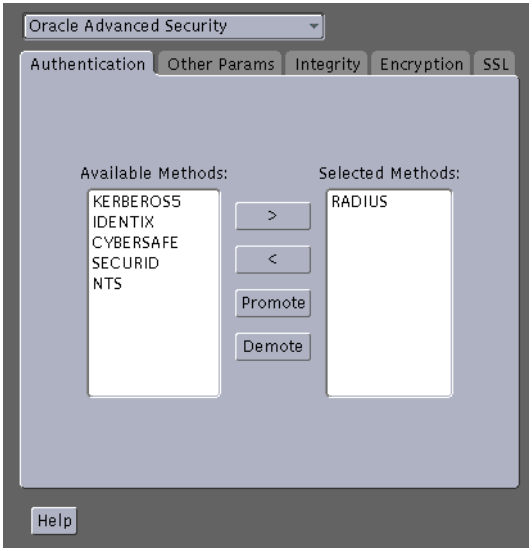
---

> **Note:** For security reasons, Oracle Corporation recommends that you change the `radius.key` file to root access only (on UNIX servers).

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

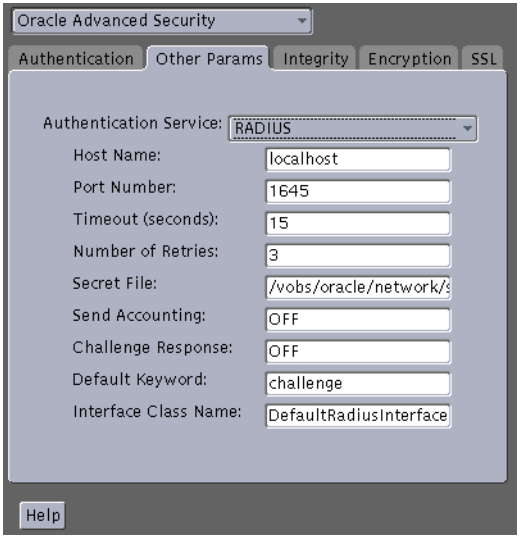
1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the Navigator window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security.

The Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. From the Available Methods list, select RADIUS.
6. Move RADIUS to the Selected Methods list by choosing the right-arrow [>].
7. To arrange the selected methods in order of desired use, select a method in the Selected Methods list, and choose Promote or Demote to position it in the list. For example, if you want RADIUS to be the first service used, put it at the top of the list.

8. Choose the Other Params tab; the Other Params window appears:



9. From the Authentication Service list, select RADIUS.
10. In the Host Name field, accept the `localhost` as the default primary RADIUS server, or enter another host name.
11. Ensure that the default value of the Secret File field is valid.
12. Choose File > Save Network Configuration.

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

    ```
    SQLNET.AUTHENTICATION_SERVICES=service

    SQLNET.RADIUS_AUTHENTICATION=location
    ```

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

Set Oracle Database Server Initialization Parameters

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

```
REMOTE_OS_AUTHENT=FALSE

OS_AUTHENT_PREFIX=""
```

Caution: Setting REMOTE_OS_AUTHENT to TRUE can 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).

See Also: *Oracle8i Reference* and the *Oracle8i Administrator's Guide*, for information about setting initialization parameters on the Oracle8*i* database server

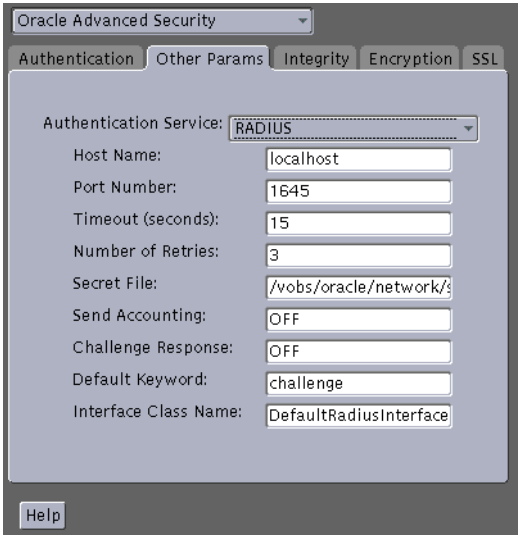
Step 3: Configure Additional RADIUS Features

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

Change Default Settings

1. 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 window, expand Local > Profile.

3. From the list in the right window pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Other Params tab.
5. From the Authentication Service list, select RADIUS.

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

Field	Description
Port Number	Specifies the listening port of the primary RADIUS server. The default value is 1645.
Timeout (seconds)	Specifies the time the Oracle database server waits for a response from the primary RADIUS server. The default is 15 seconds.
Number of Retries	Specifies the number of times the Oracle database server resends messages to the primary RADIUS server. The default is three retries. For instructions on configuring RADIUS accounting, see: Task 4: Configure RADIUS Accounting on page 4-19 .
Secret File	Specifies the location of the secret key on the Oracle database server. The field specifies the location of the secret key file, not the secret key itself. For information about specifying the secret key, see: Create the RADIUS Secret Key File on the Oracle Database Server on page 4-10 .

7. Choose File > Save Network Configuration.

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

```
SQLNET.RADIUS_AUTHENTICATION_PORT=(PORT)

SQLNET.RADIUS_AUTHENTICATION_TIMEOUT=
(NUMBER OF SECONDS TO WAIT FOR response)

SQLNET.RADIUS_AUTHENTICATION_RETRIES=
(NUMBER OF TIMES TO RE-SEND TO RADIUS server)

SQLNET.RADIUS_SECRET=(path/radius.key)
```

Configure Challenge-Response

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

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

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

To configure challenge-response:

1. If you are using JDK 1.1.7 or JRE 1.1.7, set the JAVA_HOME environment variable to the JRE or JDK location on the system where the Oracle client is run:
   - On UNIX, enter this command at the prompt:

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

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

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

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

   ---

2. If you are using a third-party graphic user interface, you must define its location by entering parameters in each of the following configuration files:
   - In `sqlnet.ora`:

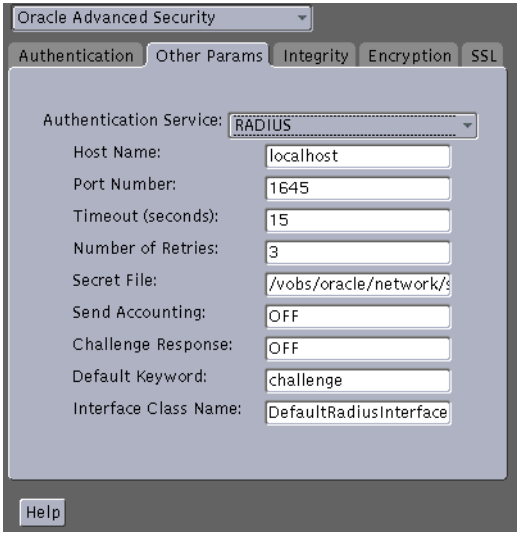
Enter `SQLNET.RADIUS_CLASSPATH=(location)`, where `location` is the complete pathname of the jar file. It defaults to `$oracle_home/network/jlib/netradius.jar`.

- In sqlnet.radius: Enter `sqlnet.radius_authentication_interface`.

Use a text editor to add the SQLNET.RADIUS_CLASSPATH parameter to the `sqlnet.ora` file (to set the path for the Java classes for the graphical interface), as follows:

```
SQLNET.RADIUS_CLASSPATH=(location), where location is the
complete pathname of the netradius.jar file.
```

3. Start Net8 Assistant:
   - 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 window, expand Local > Profile.
5. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



6. Choose the Other Params tab.

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

   Note: The keyword feature is provided by Oracle and supported by some, but not all, RADIUS servers. You can use this feature only if your RADIUS server supports it.

   By setting a keyword, you let the user avoid using a password to verify identity. If the user does not enter a password, the keyword you set here is passed to the RADIUS server which responds with a challenge requesting, for example, a driver's license number or birth date. If the user does enter a password, the RADIUS server may or may not respond with a challenge, depending upon the configuration of the RADIUS server.

10. In the Interface Class Name field, accept the default value of *DefaultRadiusInterface* or enter the name of the class you have created to handle the challenge-response conversation between the Oracle client and the RADIUS server.
11. Choose File > Save Network Configuration.

    The `sqlnet.ora` file updates with these entries:

    `SQLNET.RADIUS_CHALLENGE_RESPONSE=([ON | OFF])`

    `SQLNET.RADIUS_CHALLENGE_KEYWORD=(KEYWORD)`

    `SQLNET.RADIUS_AUTHENTICATION_INTERFACE=(`*name of interface including the package name delimited by "/" for "."*`)`

Set Parameters for an Alternate RADIUS Server

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

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

`SQLNET.RADIUS_ALTERNATE_PORT=(`*`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: Create a User and Grant Access

To grant user access:

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

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

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

   See Also: *Oracle8i Administrator's Guide* and *Oracle8i Distributed Database Systems*

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

   See Also: Administration documentation for the RADIUS server

Task 4: Configure RADIUS Accounting

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

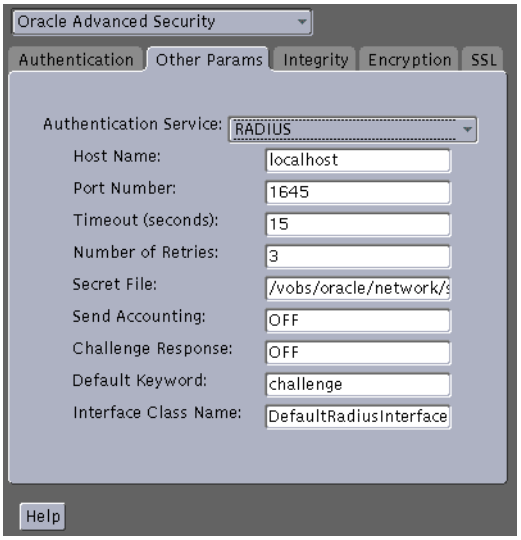
Set RADIUS Accounting on the Oracle Database Server

To enable or disable RADIUS accounting:

1. Start 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 window, expand Local > Profile.
3. From the list in the right window pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Other Params tab.
5. From the Authentication Service list, select RADIUS.
6. In the Send Accounting field, enter ON to enable accounting or OFF to disable accounting.
7. Choose File > Save Network Configuration.

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

   ```
   SQLNET.RADIUS_SEND_ACCOUNTING= ON
   ```

Configure the RADIUS Accounting Server

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

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

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

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

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

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

   ```
   @ (#) clients 1.1 2/21/96 Copyright 1991 Livingston
   Enterprises Inc

   This file contains a list of clients which are allowed to
   make authentication requests and their encryption key. The
   first field is a valid hostname. The second field
   (separated by blanks or tabs) is the encryption key.

   Client Name                    Key
   ```

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

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

3. Save and close the clients file.

See Also: Administration documentation for the RADIUS server

Task 6: Configure the Authentication Server for Use with RADIUS

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

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

See the RADIUS server documentation.

Task 8: Configure Mapping Roles

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

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

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

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

To configure roles on the RADIUS server, use the following syntax:

```
ORA_DatabaseName.DatabaseDomainName_RoleName
```

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

Example:

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

See Also: *Oracle 8i Administrator's Guide*

Logging in to the Database

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

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

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

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

```
CONNECT /@database_alias
```

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

5

Configuring CyberSafe Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle 8*i*, or for the Oracle8*i* server, so that CyberSafe TrustBroker, a Kerberos-based authentication server, can be used to authenticate Oracle users.

This chapter contains the following sections:

- Configuring CyberSafe Authentication
- Troubleshooting

Configuring CyberSafe Authentication

To configure CyberSafe authentication:

- ❏ Task 1: Install the CyberSafe Server
- ❏ Task 2: Install the CyberSafe TrustBroker Client
- ❏ Task 3: Install the CyberSafe Application Security Toolkit
- ❏ Task 4: Configure a Service Principal for an Oracle Database Server
- ❏ Task 5: Extract the Service Table from CyberSafe
- ❏ Task 6: Install an Oracle Database Server
- ❏ Task 7: Install Oracle Advanced Security With CyberSafe
- ❏ Task 8: Configure Net8 and Oracle8i
- ❏ Task 9: Configure CyberSafe Authentication
- ❏ Task 10: Create a CyberSafe User on the Authentication Server
- ❏ Task 11: Create an Externally Authenticated Oracle User on the Oracle Database Server
- ❏ Task 12: Get the Initial Ticket for the CyberSafe / Oracle User
- ❏ Task 13: Connect to an Oracle Database Server Authenticated by CyberSafe

Task 1: Install the CyberSafe Server

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

> **See Also:** CyberSafe documentation listed under Related Documents on page -xxv

Task 2: Install the CyberSafe TrustBroker Client

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

> **See Also:** CyberSafe documentation listed under Related Documents on page -xxv

Task 3: Install the CyberSafe Application Security Toolkit

Perform this task on both the client and server systems.

See Also: CyberSafe documentation listed under Related Documents on page -xxv

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

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

The name of the principal has the following format:

```
kservice/kinstance@REALM
```

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

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

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

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

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

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

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

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

Task 5: Extract the Service Table from CyberSafe

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

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

1. Enter the following in `kdb5_edit`:

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

   kdb5_edit:  exit

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

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

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

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

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

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

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

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

Task 6: Install an Oracle Database Server

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

See Also: 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.

See Also: Oracle8*i* installation documentation for your platform

Task 8: Configure Net8 and Oracle8*i*

Configure Net8 and Oracle8*i* on both the server and client systems.

See Also: Oracle8i installation documentation for your platform

Task 9: Configure CyberSafe Authentication

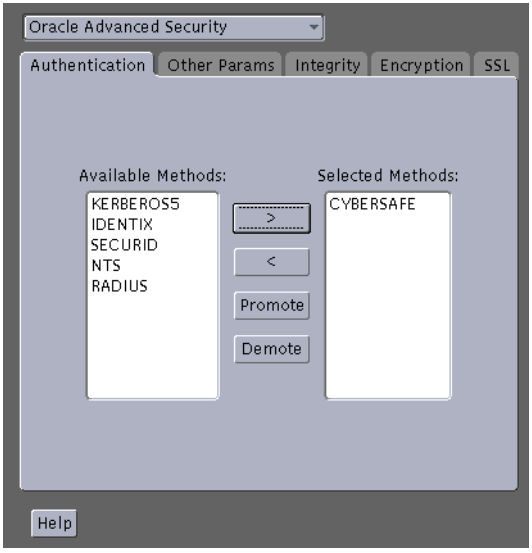
Perform the following tasks to set parameters in the Oracle database server and client `sqlnet.ora` files to configure CyberSafe:

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

Configure CyberSafe on both the Client and the Oracle Database Server

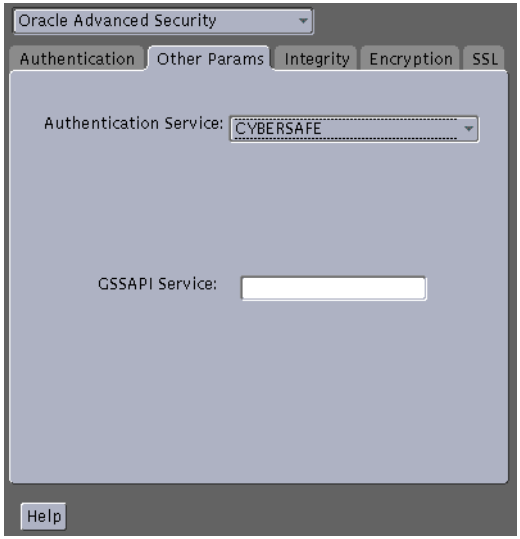
To configure CyberSafe authentication service parameters on both the client and the database server:

1. Start 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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. In the Available Methods list, select CYBERSAFE.
6. Move CYBERSAFE to the Selected Methods list by choosing the right-arrow [>].
7. Arrange the selected methods in order of desired use. To do this, select a method in the Selected Methods list, then choose Promote or Demote to position it in the list. For example, if you want CYBERSAFE to be the first service used, put it at the top of the list.

8. Choose the Other Params tab.



9. From the Authentication Service list, select CYBERSAFE.
10. Enter the name of the GSSAPI Service, as in the following example:

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

    Insert the principal name, using the format described in Task 4: Configure a Service Principal for an Oracle Database Server.
11. Choose File > Save Network Configuration.

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

    ```
    SQLNET.AUTHENTICATION_SERVICES=(CYBERSAFE)

    SQLNET.AUTHENTICATION_GSSAPI_
    SERVICE=KSERVICE/KINSTANCE@REALM
    ```

Set REMOTE_OS_AUTHENT in the Initialization Parameter File

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

```
REMOTE_OS_AUTHENT=FALSE
```

> **Note:** Setting REMOTE_OS_AUTHENT to TRUE can allow a security breach because it allows someone using a non-secure protocol, such as TCP, 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 database server after modifying the configuration files to enable the changes.

> **See Also:** Operating system specific documentation and *Oracle8i Administrator's Guide* for more information about how to restart the Oracle database server

Task 10: Create a CyberSafe User on the Authentication Server

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

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

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

Enter the following:

```
#
kdb5_edit
```

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

See Also: Cybersafe documentation listed in Related Documents on page -xxv for information about creating the realm

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

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

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

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

See Also: *Oracle8i Administrator's Guide*

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

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

1. Enter the following:

   ```
   % kinit cyberuser
   ```

2. Enter the password *(password does not display)*.
3. To list currently owned tickets, run `klist` on the clients. Enter the following at the system command prompt:

   ```
   % klist
   ```

   The system displays the following information:

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

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

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

```
%  sqlplus /@net_service_name
```

where `net_service_name` is a Net8 service name.

For example:

```
% sqlplus /@npddoc_db
```

> **See Also:** Chapter 1, Introduction to Oracle Advanced Security, and *Oracle8i Distributed Database Systems*

Troubleshooting

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

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

- ❏ Ensure that the default realm is correct by looking at `krb.conf`.
- ❏ Ensure that the TrustBroker Master Server is running on the host specified for the realm.
- ❏ Ensure that the Master Server has an entry for the user principal and that the passwords match.
- ❏ Ensure that the `krb.conf` and `krb.realms` files are readable by Oracle.

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

- ❏ After trying to connect, check for a service ticket.
- ❏ Check that the `sqlnet.ora` file on the database server side has a service name that corresponds to a service known to the CyberSafe Master Server.
- ❏ Check that the clocks on all the involved systems are within a few minutes of each other.

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

- ❏ Check the clocks on the client and database server.
- ❏ Check that the `v5srvtab` file exists in the correct location and is readable by Oracle.
- ❏ Check that the `v5srvtab` file has been generated for the service named in the profile (`sqlnet.ora`) on the database server side.

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

- ❏ Check that the initial ticket is forwardable. You must have obtained the initial ticket by running `kinit -f`.
- ❏ Check the expiration date on the credentials.
- ❏ If the credentials have expired, close the connection and run `kinit` to get a new initial ticket.

6

Configuring Kerberos Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle8*i*, or for the Oracle8*i* server, for use with Kerberos authentication, and how to configure Kerberos to authenticate Oracle database users. This chapter contins the following sections:

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

Enabling Kerberos Authentication

To enable Kerberos authentication:

- ❏ Task 1: Install Kerberos
- ❏ Task 2: Configure a Service Principal for an Oracle Database Server
- ❏ Task 3: Extract a Service Table from Kerberos
- ❏ Task 4: Install an Oracle Database Server and an Oracle Client
- ❏ Task 5: Install 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 system that functions as the authentication server

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

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

To enable the Oracle database server to validate the identity of clients that authenticate themselves using Kerberos, you must create a **service principal** for Oracle8*i*.

The name of the principal should have the following format:

```
kservice/kinstance@REALM
```

kservice	A case-sensitive string that represents the Oracle service; this can be the same as the database service name.
kinstance	This is typically the fully-qualified name of the system on which Oracle8*i* is running.
REALM	The domain name of the database server. REALM must always be uppercase, and is typically the DNS domain name.

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

For example, if `kservice` is *oracle*, the fully-qualified name of the system on which Oracle8*i* is running is `dbserver.someco.com`, and the realm is `SOMECO.COM`; the principal name is:

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

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

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

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

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

Task 3: Extract a Service Table from Kerberos

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

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

1. Enter the following:

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

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

kadmin.local:  exit

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

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

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

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

   The following example is UNIX-specific.

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

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

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

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

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

Install the Oracle database server and client software.

See Also: Oracle8i operating system specific documentation

Task 5: Install Net8 and Oracle Advanced Security

Install Net8 and Oracle Advanced Security on the Oracle database server and Oracle client systems.

See Also: Oracle8*i* operating system specific installation documentation

Task 6: Configure Net8 and Oracle

Configure Net8 on the Oracle database server and client.

See Also:

- Oracle8*i* operating system specific installation documentation
- *Net8 Administrator's Guide*.

Task 7: Configure Kerberos Authentication

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

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

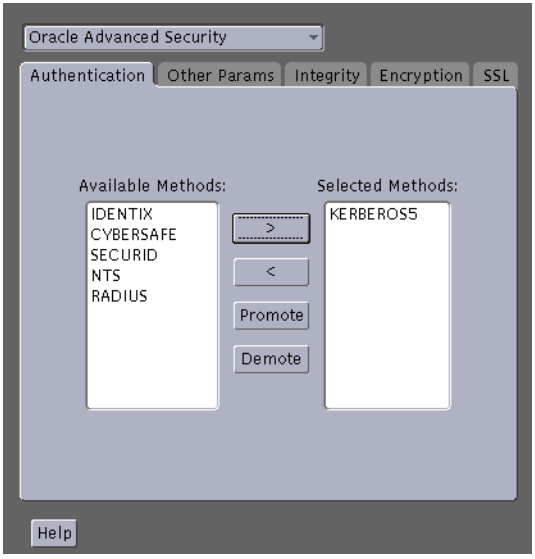
Step 1: Configure Kerberos on the Client and on the Database Server

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

1. Start 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 window, expand Local > Profile.

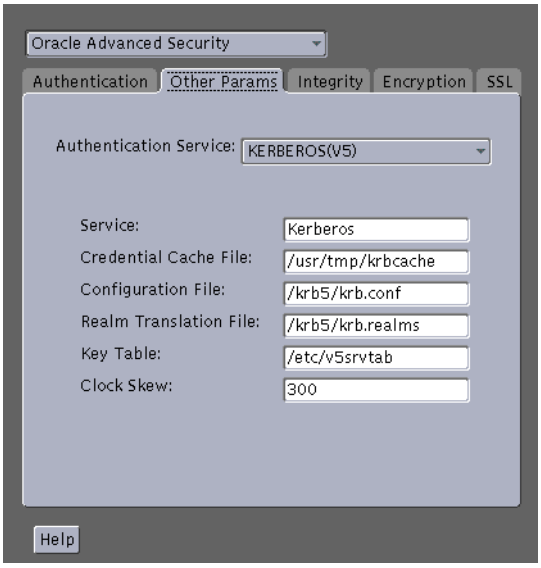
3. From the list in the right pane, select Oracle Advanced Security.

   The Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. From the Available Methods list, select KERBEROS5.
6. Move KERBEROS5 to the Selected Methods list by clicking the right-arrow [>].
7. Arrange the selected methods in order of use. To do this, select a method in the Selected Methods list, then click Promote or Demote to position it in the list. For example, if you want KERBEROS5 to be the first service used, move it to the top of the list.

8. Choose the Other Params tab.



9. From the Authentication Service list, select KERBEROS(V5).
10. The Service field defines the name of the service Oracle8*i* uses to obtain a Kerberos **service ticket**; enter `Kerberos`. When you provide the value for this field, the other fields are enabled.
11. Optionally enter values for the following fields:
    - Credential Cache File
    - Configuration File
    - Realm Translation File
    - Key Table
    - Clock Skew

    See Also: Net8 Assistant online help, and Step 3: Set sqlnet.ora Parameters (optional) on page 6-9, for more information about the fields and the parameters they configure

12. Choose File > Save Network Configuration.

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

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

Step 2: Set the Initialization Parameters

To set parameters in the initialization parameter file:

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

   ```
   REMOTE_OS_AUTHENT=FALSE
   ```

   Attention: Setting REMOTE_OS_AUTHENT to TRUE can 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 you set the value of OS_AUTHENT_PREFIX to *null* as follows:

   ```
   OS_AUTHENT_PREFIX=""
   ```

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

Step 3: Set sqlnet.ora Parameters (optional)

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

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

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

Task 8: Create a Kerberos User

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

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

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

The following example is UNIX specific:

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

Task 9: Create an Externally-authenticated Oracle User

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

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

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

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

```
% okinit user_name
```

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

```
sqlplus /@oracle
```

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

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

Utilities for the Kerberos Authentication Adapter

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

- Use `okinit` to obtain an initial ticket.
- Use `oklist` to display credentials
- Use `okdstry` to remove credentials from the credentials cache.

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

Use okinit to Obtain the Initial Ticket

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

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

Table 6–1 Options for the okinit Utility

Option	Description
`-f`	Ask for a forwardable ticket-granting ticket. This option is necessary to follow database links.
`-l`	Specify the lifetime of the ticket-granting ticket and all subsequent tickets. By default, the ticket-granting ticket is good for eight (8) hours, but shorter or longer-lived credentials may be desired. Note that the KDC can ignore this option or put site-configured limits on what can be specified. The lifetime value is a string that consists of a number qualified by w (weeks), d (days), h (hours), m (months), or s (seconds), as in the following example: `okinit -l 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

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

Table 6–2 Options for the oklist Utility

Option	Description
`-f`	Show flags with credentials. Relevant flags are I, credential is a ticket-granting ticket, F, credential is forwardable, and f, credential is forwarded.
`-c`	Specify an alternative credential cache. In UNIX, the default is `/tmp/krb5cc_`*`uid`*. The alternate credential cache can also be specified by using the SQLNET.KERBEROS5_CC_NAME parameter in the `sqlnet.ora` file.
`-k`	List the entries in the service table (default `/etc/v5srvtab`) on UNIX. The alternate service table can also be specified by using the SQLNET.KERBEROS5_KEYTAB parameter in the `sqlnet.ora` file.

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

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

Use OKDSTRY to Remove Credentials from the Cache File

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

```
$ okdstry -f
```

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

Connecting to an Oracle Database Server Authenticated by Kerberos

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

```
$ sqlplus /@net_service_name
```

where net_service_name is a Net8 service name. For example:

```
$ sqlplus /@oracle_dbname
```

> **See Also:** Chapter 1, Introduction to Oracle Advanced Security, for information about external authentication and *Oracle8i Distributed Database Systems*

Troubleshooting

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

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

- ❏ Ensure that the default realm is correct by examining the `krb.conf` file.
- ❏ Ensure that the KDC is running on the host specified for the realm.
- ❏ Ensure that the KDC has an entry for the user principal and that the passwords match.
- ❏ Ensure that the `krb.conf` and `krb.realms` files are readable by Oracle.

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

- ❏ After trying to connect, check for a service ticket.
- ❏ Check that the `sqlnet.ora` file on the database server side has a service name that corresponds to a service known by Kerberos.
- ❏ Check that the clocks on all systems involved are within a few minutes of each other (or change the SQLNET.KERBEROS5_CLOCKSKEW parameter in the `sqlnet.ora` file).

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

- ❏ Check the clocks on the client and database server.
- ❏ Check that the `v5srvtab` file exists in the correct location and is readable by Oracle (remember to see the `sqlnet.ora` parameters).
- ❏ Check that the `v5srvtab` file has been generated for the service named in the `sqlnet.ora` file on the database server side.

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

- ❏ Check that the initial ticket is forwardable. (You must have obtained the initial ticket by running the `okinit` utility.)
- ❏ Check the expiration date on the credentials.
- ❏ If the credentials have expired, close the connection and run `okinit` to get a new initial ticket.

7

Configuring SecurID Authentication

This chapter describes how to configure Oracle Advanced Security for Oracle8*i*, or for the Oracle8*i* server, for use with SecurID authentication. It assumes that you are familiar with the RSA Data Security, Inc. ACE/Server, and that the ACE/Server is installed and running. This chapter contains the following sections:

- Prerequisites
- Known Limitations
- Enabling SecurID Authentication
- Creating Users for SecurID Authentication
- Using SecurID Authentication
- Troubleshooting

See Also: **Related Documents** on page -xxv, for a list of documents related to SecurID authentication

---

Note: Oracle Advanced Security Release 8.1.7 is the last release to support the SecurID adapter. Effective with the next release (8.2), the SecurID functionality will be available through RADIUS; RADIUS support is built into the RSA ACE/Server. The RADIUS adapter is described by Chapter 4, Configuring RADIUS Authentication.

Prerequisites

The following are prerequisites for configuring and using SecurID authentication:

- Net8
- Oracle 8.0.3 or higher
- ACE/Server 1.2.4 or higher
- The Oracle database 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 Oracle8*i*, Oracle8*i* 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 database server in plain 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 database server.

See Also: Chapter 2, Configuring Data Encryption and Integrity

Enabling SecurID Authentication

Enable SecurID authentication by performing these 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 system on which the Oracle server resides as a SecurID client with the ACE server. You can do this with the RSA Data Security tool `sdadmin`. To create a client:

1. Navigate to the client menu.
2. On ACE/Server 1.2.4: select Create Client
3. On ACE/Server 2.0: select Add Client

> **See Also:** RSA Data Security ACE/Server Instruction manual, version 1.2.4, or the ACE/Server version 2.0 Administration manual

Task 2: Install Oracle Advanced Security

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

> **See Also:** Oracle8*i* operating system specific installation instructions

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

1. Verify that the ACE/Server, the Oracle database server, and Oracle Advanced Security are installed.
2. Ensure that the Oracle database 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 RSA Data Security 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 NT and Windows 95/98 Platforms
- UNIX-based Platforms and ACE/Server Release 1.2.4
- UNIX-based Platforms and ACE/Server Release 2.0

Windows NT and Windows 95/98 Platforms

Ask the SecurID administrator to ensure that:

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

UNIX-based Platforms and ACE/Server Release 1.2.4

1. Install the SecurID configuration files on the Oracle8*i* database server system.

   You can obtain the files from any other SecurID client or from the system that runs the ACE/Server.

2. Create a directory (typically `/var/ace`) on the Oracle8*i* database server system and copy the configuration files to it; the `sdconf.rec` file must be present.

3. 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.

   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.

   Caution: Do not attempt to overcome access problems by running the Oracle executable `setuid root`. It is not necessary, and it is dangerous to do so.

   There are two methods (Method 1, Method 2) for configuring Oracle8*i* as a SecurID client without compromising security. Both methods work, and each allows you to use Oracle8*i* with SecurID authentication, but Method 1 is the preferred method.

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*, execute these commands 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 must 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
```

Oracle8*i* must 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 preceding symbolic link.

Whether Oracle can read the configuration data depends on how the ACE client software is installed on the Oracle database 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 next, ensure that you do not install Oracle8*i* 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 RSA Data Security 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 Oracle8*i* executable, typically *Oracle8*.

- Add the owner of the Oracle8*i* 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.

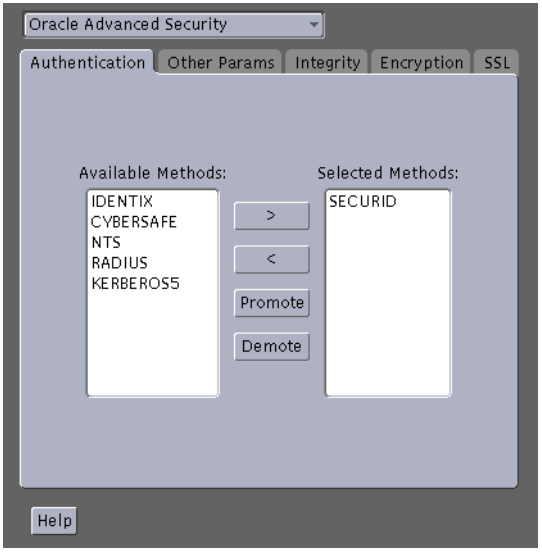
For the change to take effect (Method 1 or Method 2), do the following:

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

Task 5: Configure SecurID Authentication

To configure the 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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. From the Available Methods list, select SECURID.

6. Move SECURID to the Selected Methods list by choosing the right-arrow [>].
7. Arrange the selected methods in order of desired use. To do this, select a method in the Selected Methods list, then choose 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 is updated 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 RSA Data Security 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 RSA Data Security sdadmin Program

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

   See Also: RSA Data Security documentation listed under Related Documents on page -xxv

2. If the user should have authority to specify a new PIN to the card using the Oracle tools, choose the option that lets the user define a PIN. If you do not do this, the user will have to use the RSA Data Security tools to generate a PIN if the card is in new-PIN mode.
3. Activate the user on the Oracle8*i* database server; the server should already be registered as a SecurID client.

Task 2: Create an Oracle Server Account for the User

You can create an Oracle database 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 RSA Data Security 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, if you have assigned a card to the user `king`, and OS_AUTHENT_PREFIX has been set to a *null* value (""), create an Oracle user account using the following script:

```
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 Oracle8*i* using the appropriate SecurID card.

> **See Also:** "Logging On to the Oracle Server" on page 7-10, for information about how to log on to an Oracle8*i* database server after SecurID authentication has been installed and configured

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:

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

Preparing to Use SecurID Authentication

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 on page 7-7.

Logging On to the Oracle Server

SecurID authentication allows users to log on to the Oracle database 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)	Enter the PIN as part of the Oracle connect statement.
PINPAD (model SD520)	Enter the PIN directly onto the card.

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 RSA Data Security 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 must 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 RSA Data Security 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 Oracle8*i*, you receive the following error message:

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

Select a PIN

To assign a PIN to a card you must 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 may 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 Oracle8*i* 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 Oracle8*i*. The next time you want to connect to Oracle, use the procedure described in Logging On to the Oracle Server on page 7-10. 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 PIN may be rejected for the following reasons:

- 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 RSA Data Security 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 RSA Data Security 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 on actually has possession of the card. 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 to Oracle8*i*, you must specify the next two card codes. The syntax you use to log on depends on the kind of SecurID card you have.

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 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_dbname
```

This connects you to the Oracle8*i* database server and puts the card back into normal mode. The next time you want to log on to the Oracle server, use the procedure described in Logging On to the Oracle Server on page 7-10.

Logging on with a PINPAD Card

If you have a PINPAD card, perform the following steps to log on to the Oracle database 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 database server with the two passcodes, separated by a plus (+) 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 Oracle8*i* 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

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

- The services map should have an entry for the RSA Data Security 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 system running the Oracle database 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 database server system is registered as a SecurID client. You can do this by using the RSA Data Security tool `sdadmin`.

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

- RSA Data Security, Inc. 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 database server and the ACE server happens at the Oracle database 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.

8

Configuring Identix Biometric Authentication

This chapter describes how to configure Oracle8*i* for use with Identix biometric authentication, in the following sections:

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

> **Note:** Oracle Advanced Security Release 8.1.7 is the last release to support the Indentix Biometric adapter. Effective with the next release (8.2), this functionality will be available through the RADIUS adapter if Indentix supports the RADIUS protocol by that time. Oracle Advanced Security will continue to support industry-standard protocols, such as RADIUS. The RADIUS adapter is described by Chapter 4, Configuring RADIUS Authentication.

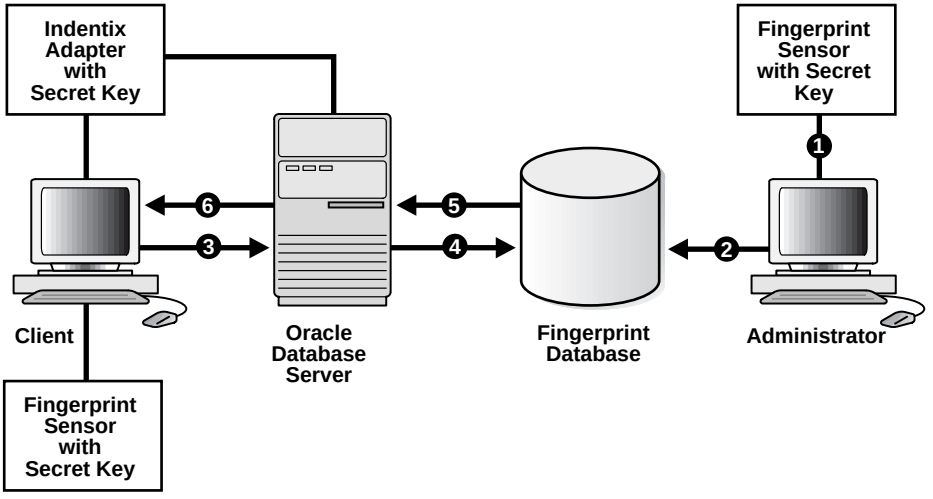
Overview

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 configuration of the Biometric Authentication Service.

Figure 8–1 Typical Biometric Authentication Service Configuration



The Fingerprint Repository has an administrator who is responsible for enrolling multiple user fingerprint templates, and defining the default policy 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 a user authentication request, 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:
  - Verification threshold
  - False finger threshold
  - High security threshold

    See Also: Identix documentation for detailed information about the threshold levels

- At the client, the biometric authentication service passes the user fingerprint template and the threshold values to the sensor. The user fingerprint is scanned and the verification result, fingerprint template, secret key, and threshold values are used to generate a hash total. The hash total is sent to the server-side adapter for comparison with the hash total generated by the server-side adapter.

Architecture of the Biometric Authentication Service

The Biometric Authentication Service consists of the following modules:

Biometric Manager *(the manager)*	The administrator uses this module to enter the security policy and fingerprints.
Biometric Authentication Server *(authentication server)*	A specially configured version of an Oracle database server, this module stores the security policies and fingerprint templates.

Identix Authentication Adapters	Identix authentication adapters are used on both the client and database servers to communicate biometric authentication data between the authentication server and client systems—to authenticate a database user.

Both the manager and the client-side adapter interface with the following Identix products:

- ❏ TouchSafe II Software Libraries
- ❏ TouchSafe II Hardware Interface
- ❏ TouchSafe II Desktop Sensor
- ❏ TouchSafe III software libraries
- ❏ TouchSafe III Desktop Sensor

> **See Also:** Related Documents on page -xxv for a list of Identix documentation that describes these Identix products

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. The adapter 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 the user to place a finger on the TouchSafe II Desktop Sensor. The adapters on the client and the database servers 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 system that is to become the manager PC must be running Oracle Enterprise Manager 2.0 or later.
- ❏ Each Windows NT or Windows 95/98 system 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*.
- ❏ Ensure that each Windows NT and Windows 95/98 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.

During the installation of the Biometrics Manager, if you chose not 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 hexadecimal. Ensure that the hash key matches the one set in the default security policy.
6. Re-boot the system; the device driver should start.

   To ensure that the device driver is running, check the Device Manager in Control Panel after re-boot; the device ETSIINT should be running.

Configuring the Biometric Manager PC

To configure the Biometric manager PC:

1. Install Oracle Enterprise Manager on both the Oracle database server and the Oracle client.
2. Install both the Identix hardware and driver firmware, and configure the Identix variables and devices.
3. Install and test Identix TouchSafe II (Encrypt) 2.0 or TouchSafe III.

   > **See Also:** Installing the TouchSafe II Encrypt Device Driver for Windows NT on page 8-5, 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.

Configuring the Client PC

To configure each client PC system:

1. Install Oracle Enterprise Manager.
2. Install the Identix hardware and the Identix driver firmware and configure the Identix variables and devices. See the Identix Readme file for additional information.
3. 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.

   See Also: Installing the TouchSafe II Encrypt Device Driver for Windows NT on page 8-5, and the platform-specific installation documentation

4. Install the Oracle Advanced Security Identix authentication adapter.

   See Also: Platform-specific documentation and the Identix Readme file

Configuring Each Database Server

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

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

Enabling Biometric Authentication

The Biometric Authentication Service is a 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 installed on a secure system with strict security and access controls. The Identix adapter should *not* be installed on this database.

To configure the Biometric Authentication Service:

- ❏ Task 1: Configure the Database Server
- ❏ Task 2: Configure Identix Authentication
- ❏ Task 3: Establish a Net Service Name
- ❏ Task 4: Verify the Database Server Address
- ❏ Task 5: Configure the Biometric Manager PC

Task 1: Configure the Database Server

To configure the database server that is to become the Authentication 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

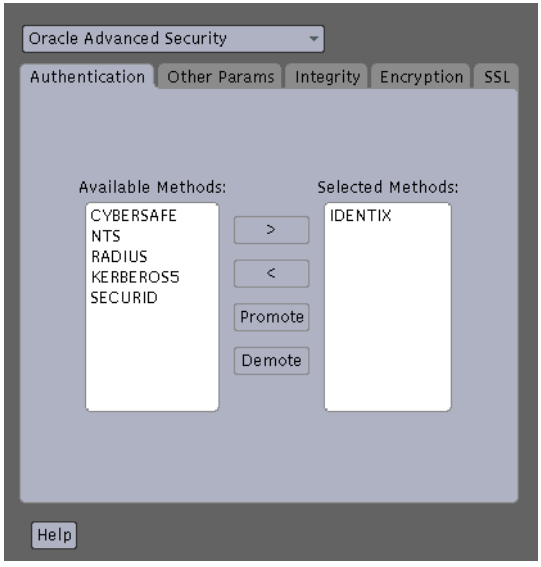
To configure Identix authentication:

1. Configure an Authentication Method and Fingerprint Server on the Client and Server Systems
2. Configure the User Name, Password, and Fingerprint Method
3. Configure the Initialization Parameter File
4. 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 Systems

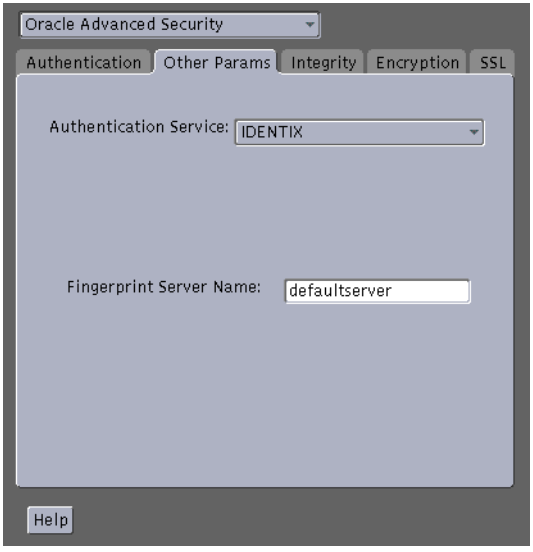
1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the Navigator window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security;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



4. Choose the Authentication tab.
5. From the Available Methods list, select IDENTIX.
6. Move IDENTIX to the Selected Methods list by choosing the right-arrow [>].
7. Arrange the selected methods in order of use. To do this, select a method in the Selected Methods list, and choose Promote or Demote to position it in the list.

   For example, to select IDENTIX as the first service used, put it at the top of the list.

8. Choose the Other Params tab:



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

    The `sqlnet.ora` file is updated 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 `ofm_client`, and *password* is `ofm_client`.

Note:

- The `samples` directory contains a file that shows how to set these parameters.
- The ofm_client user name and password are set up by running nauicat.sql. *Do 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 on the database server (`tnsnames.ora`) contains the service name of the fingerprint repository. If they are on the same database, enter 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

Establish a net service name for the fingerprint repository server.

> **See Also:** *Net8 Administrator's Guide* for information about net service names

Task 4: Verify the Database Server Address

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

> **See Also:** *Net8 Administrator's Guide* for information about 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.

> **See Also:** *Net8 Administrator's Guide* for information about net service name configuration

Administering the Biometric Authentication Service

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

See Also: Identix documentation

Create a Hashkey on Each of the Clients

Use the Identix Setkey utility to configure a hexadecimal hashkey on each of the client systems. 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.
2. On the database server, restart the database and create an Oracle database server account for the user.

   Use SQL*Plus if using the Oracle Enterprise Manager, or SQL*Plus connected as a user with the CREATE USER database privilege.

   To create an account, enter the following:

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

   OS_AUTHENT_PREFIX is an Oracle database server initialization parameter. The default value for OS_AUTHENT_PREFIX is OPS$. The user name in this step should match the user name created on the client.

   If you reset the OS_AUTHENT_PREFIX parameter, you must 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=""
```

For example:

If you create the user `king` on the client, and set OS_AUTHENT_PREFIX to a null value (""), use SQL*Plus to create an Oracle user account as follows:

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

At a 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 Oracle8*i*.

See Also:

- *Oracle8i Administrator's Guide* and *Oracle8i Distributed Database Systems*, for information about creating users identified externally
- Identix documentation, for information about the Biometric Authentication Service, and storing the secret key on the client

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 described in Administering the Biometric Authentication Service on page 8-13 have been executed.

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 setting in the following example 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 the Net8 net service name.

5. The Net8 Native Authentication dialog box appears; choose OK.

   Note: On some systems, this dialog box is displayed behind the current window.

6. When a message appears telling you to place your finger on the desktop fingerprint sensor, use the same finger that you entered into the authentication server repository; remove your finger at the prompt. Another prompt tells you whether you have been authenticated.

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

   - Restart the authentication process.
   - Request the security administrator to lower the threshold value to *80*.
   - Request the security administrator to re-enroll you.

   See Also: Biometric Manager online help for task oriented procedures

Troubleshooting

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

- ❏ The Identix Set Key utility hash key must match the Biometric Manager default policy hash key.
- ❏ The NT user name must match the externally defined user name in the database server and the user name added to 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, 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, although *this is not recommended.* If you do this, add the following lines 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 Secure Socket Layer Authentication

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

- SSL in an Oracle Environment
- SSL Beyond an Oracle Environment
- SSL Combined with Other Authentication Methods
- Issues When Using SSL
- Enabling SSL

SSL in an Oracle Environment

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

This section discusses the following topics:

- What You Can Do with SSL
- Architecture of SSL in an Oracle Environment
- Components of SSL in an Oracle Environment
- How SSL Works in an Oracle Environment: The SSL Handshake

What You Can Do with SSL

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

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

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

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

- Only the server authenticates itself to the client
- Both client and server authenticate themselves to each other
- Neither the client nor the server authenticates itself to the other, thus using the SSL encryption feature by itself

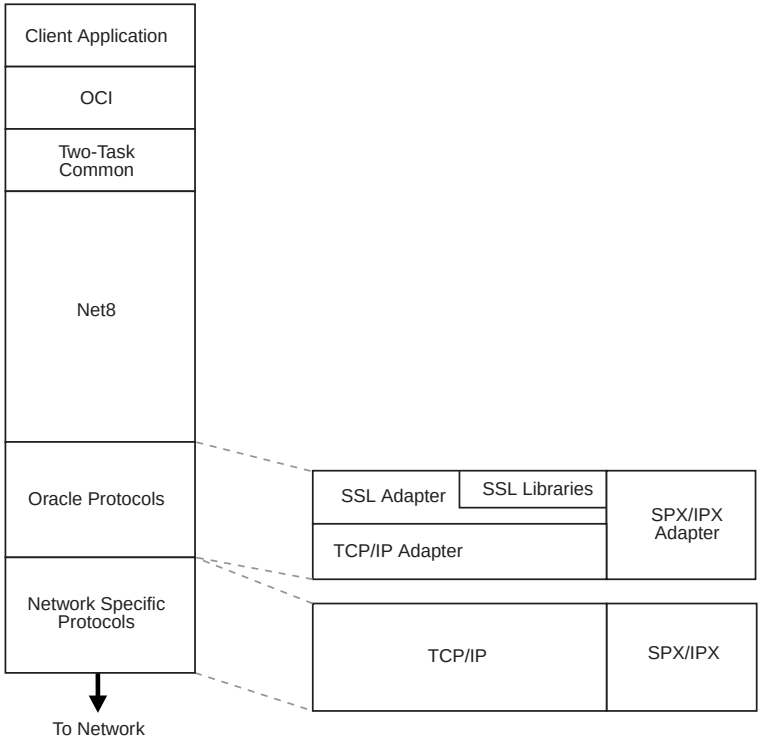
See Also:

- *The SSL Protocol*, Version 3.0, published by the Internet engineering Task Force, for a more detailed discussion of SSL
- Chapter 1, Configuring Secure Socket Layer Authentication, for more information about authentication methods

Architecture of SSL in an Oracle Environment

In an Oracle environment, SSL operates at the Oracle Protocols layer using TCP/IP, as illustrated by Figure 9–1:

Figure 9–1 SSL Architecture in an Oracle Environment



Components of SSL in an Oracle Environment

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

- Certificate Authority
- Certificate
- Wallet

Certificate Authority

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

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

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

Network entities can obtain their certificates from the same or different CAs. By default, Oracle Advanced Security automatically installs trusted certificates from VeriSign, RSA, Entrust, and GTE CyberTrust when you install a new wallet (See: Wallet on page 9-5).

Certificate

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

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

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

Wallet

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

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

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

Note: Installation of Oracle Advanced Security Release 8.1.7 also installs Oracle Wallet Manager release 2.0 and Oracle Enterprise Login Assistant release 1.0.

See Also:

- Chapter 18, Using Oracle Wallet Manager
- Creating a New Wallet on page 18-4.
- Managing Trusted Certificates on page 18-12.

How SSL Works in an Oracle Environment: The SSL Handshake

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

- The client and server establish which **cipher suites** to use.
- The server sends its certificate to the client, and the client verifies that the server's certificate was signed by a trusted CA.
- Similarly, if client authentication is required, the client sends its own certificate to the server, and the server verifies that the client's certificate was signed by a trusted CA.
- The client and server exchange key information using public key cryptography; based on this information, each generates a **session key**. All subsequent communications between the client and the server is encrypted and decrypted by using this set of session keys and the negotiated cipher suite.

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

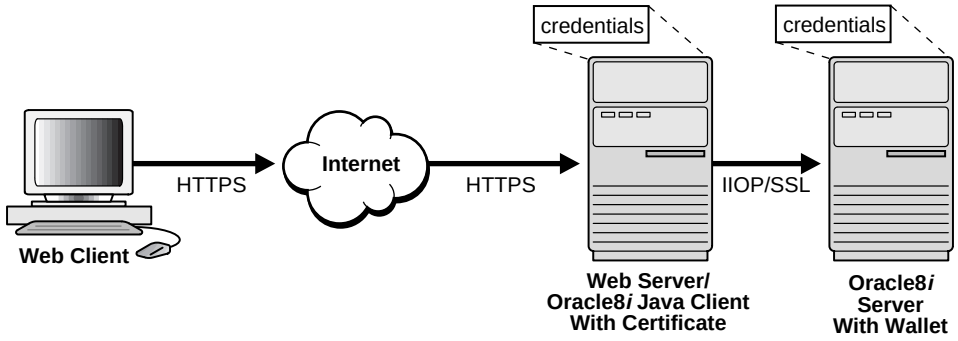
1. The user initiates a Net8 connection to the server by using SSL.
2. SSL performs the handshake between the client and the server.
3. If the handshake is successful, the server verifies that the user has the appropriate **authorization** to access the database.

SSL Beyond an Oracle Environment

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

Figure 9–2 shows how SSL is used to secure connections between Oracle and non-Oracle entities over the Internet. In this example, a Web server runs as an Oracle8*i* Java client. It receives messages over **HTTPS** (HTTP secured by SSL), and sends **CORBA** requests to the Oracle database server over **IIOP**/SSL (IIOP secured by SSL). In this example, the Web server *passes its own certificate to the Oracle server*, rather than the certificate of the Web client.

Figure 9–2 Connecting to an Oracle Server over the Internet



See Also: *Oracle8i Enterprise JavaBeans Developer's Guide and Reference.* for information about using and configuring IIOP / SSL

SSL Combined with Other Authentication Methods

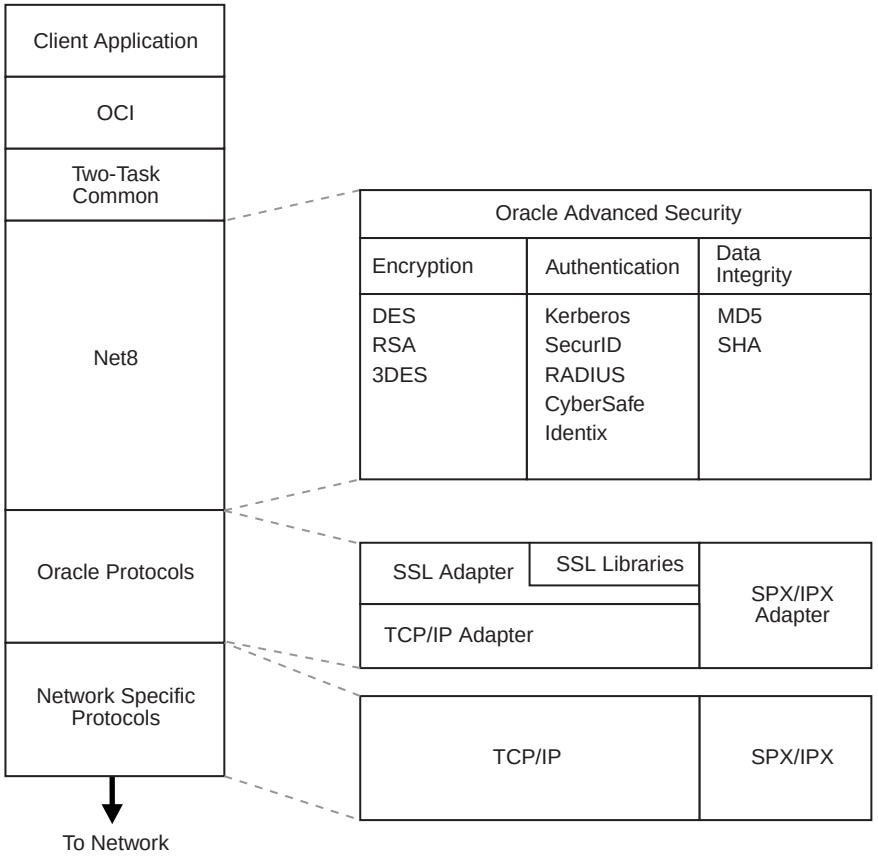
Because of its implementation architecture, you can configure Oracle Advanced Security to use SSL concurrently with other supported authentication methods, such as Kerberos, SecurID, RADIUS, and Identix—as discussed in the next sections:

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

Architecture: Oracle Advanced Security and SSL

Figure 9–3 displays the Oracle Advanced Security implementation architecture, which shows that (i) Oracle Advanced Security operates at the **session layer** on top of SSL, which (ii) uses TCP/IP at the **transport layer**. This separation of functionality lets you employ SSL concurrently with other supported protocols.

Figure 9–3 SSL in Relation to Oracle Advanced Security

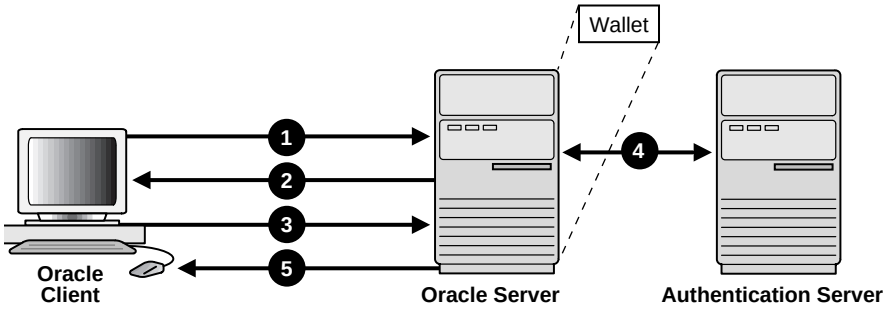


See Also: *Net8 Administrator's Guide,* for information about stack communications in an Oracle networking environment

Using SSL with Other Authentication Methods

Figure 9–4 illustrates a configuration in which SSL is used in combination with another authentication method supported by Oracle Advanced Security. In this example, SSL is used to establish the initial handshake (server authentication), and an alternative authentication method is used to authenticate the client.

Figure 9–4 SSL in Relation to Other Authentication Methods



1. The client seeks to connect to the Oracle database server.
2. SSL performs a handshake during which the server authenticates itself to the client and both the client and server establish which cipher suite to use. See How SSL Works in an Oracle Environment: The SSL Handshake on page 9-6.
3. Once the SSL handshake is successfully completed, the user seeks access to the database.
4. The Oracle database server exchanges the user's authentication information with the authentication server—using a non-SSL authentication method (e.g., Kerberos, SecurID, Indentix, RADIUS).
5. Upon validation by the authentication server, the Oracle database server grants access and authorization to the user.
6. The user accesses the Oracle database securely using SSL.

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 authorization retrieval from an LDAP-based directory service. Client-side SSL authentication is required in order to manage enterprise users and their privileges in a directory.
- Because SSL supports both authentication and encryption, it is *somewhat slower* than the standard Net8 TCP/IP transport (using native adapters).
- Oracle Advanced Security SSL requires Oracle8*i*; it does not work with earlier database releases.
- Each SSL authentication mode requires unique configuration settings. See: Enabling SSL (next section).

Note:

- *U.S. government regulations prohibit double encryption. Accordingly, if you configure Oracle Advanced Security to use SSL encryption and another encryption method concurrently, the connection fails (you also cannot configure SSL authentication concurrently with non-SSL authentication).*
- If you configure SSL encryption, you must disable non-SSL encryption. To disable such encryption, see: Disabling Oracle Advanced Security Authentication on page 11-3.

Enabling SSL

To enable SSL:

- Task 1: Install Oracle Advanced Security and Related Products
- Task 2: Configure SSL on the Client
- Task 3: Configure SSL on the Server
- Task 4: Log on to the Database

Task 1: Install Oracle Advanced Security and Related Products

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

See Also: The Oracle8*i* installation documentation for your platform.

Task 2: Configure SSL on the Client

To configure SSL on the client:

- ❏ Step 1: Confirm Wallet Creation
- ❏ Step 2: Specify Required Client Configuration (Wallet Location)
- ❏ Step 3: Set the SSL Cipher Suites on the Client (Optional)
- ❏ Step 4: Set the Required SSL Version (Optional)
- ❏ Step 5: Set SSL as an Authentication Service (Optional)
- ❏ Step 6: Create a Net Service Name that Uses TCP/IP with SSL in the Connect Descriptor

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

Step 1: Confirm Wallet Creation

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

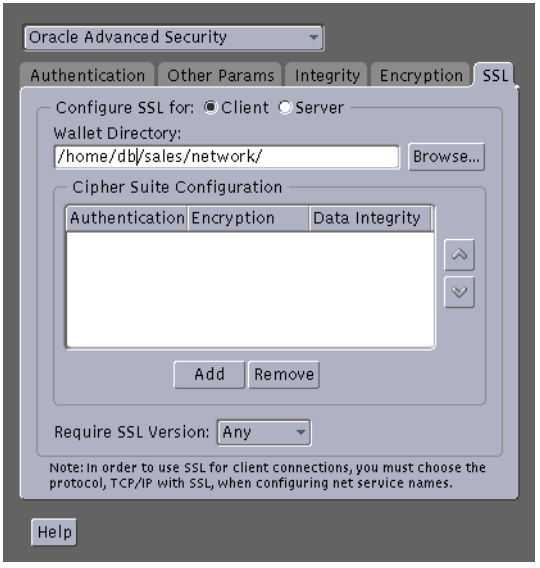
> **See Also:**
>
> - Chapter 18, Using Oracle Wallet Manager, for general information about wallets
> - Opening an Existing Wallet on page 18-5, for information about opening an existing wallet
> - Creating a New Wallet on page 18-4, for information about creating a new wallet

Step 2: 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 window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the SSL tab.
5. Select Configure SSL for Client.
6. In the Wallet Directory box, enter the directory in which the Oracle wallet is located, or click Browse to find it by searching the file system.

Important: There are two occasions during the client and the server configuration when you set the location of the Oracle wallet. *Be sure to enter the same location on both occasions.*

- On the occasion described in this section, you set the location of the wallet either by using the Net8 Assistant or by modifying the `sqlnet.ora` file.
- Later, you use the Oracle Wallet Manager. See: Step 1: Create a Database Wallet on page 17-42.

---

7. Choose File > Save Network Configuration.

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

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

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

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

When you install Oracle Advanced Security, several SSL cipher suites are set for you by default. You can override the default by setting the SSL_CIPHER_SUITES parameter. For example, if you use 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.

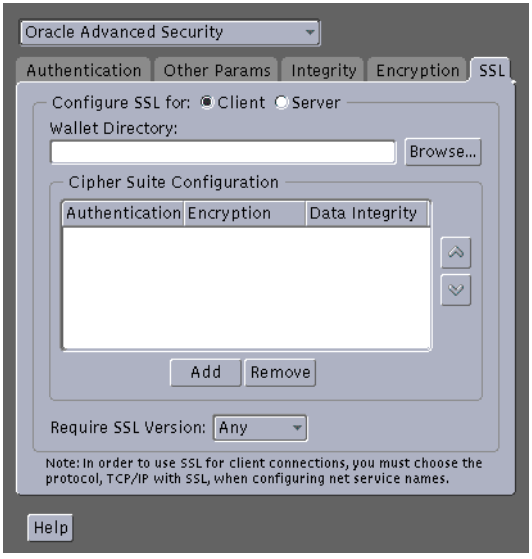
Table 9–1 lists the SSL cipher suites supported in the current release of Oracle Advanced Security. These cipher suites are set by default when you install Oracle Advanced Security. This table also lists the authentication, encryption, and data integrity types each cipher suite uses.

Table 9–1 Oracle Advanced Security Cipher Suites

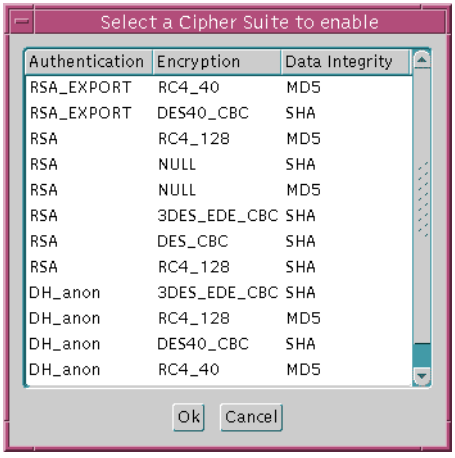
Cipher Suite	Authentication	Encryption	Data Integrity
SSL_RSA_WITH_3DES_EDE_CBC_SHA	RSA	3DES EDE CBC	SHA
SSL_RSA_WITH_RC4_128_SHA	RSA	RC4 128	SHA
SSL_RSA_WITH_RC4_128_MD5	RSA	RC4 128	MD5
SSL_RSA_WITH_DES_CBC_SHA	RSA	DES CBC	SHA
SSL_DH_anon_WITH_3DES_EDE_CBC_SHA	DH anon	3DES EDE CBC	SHA
SSL_DH_anon_WITH_RC4_128_MD5	DH anon	RC4 128	MD5
SSL_DH_anon_WITH_DES_CBC_SHA	DH anon	DES CBC	SHA
SSL_RSA_EXPORT_WITH_RC4_40_MD5	RSA	RC4 40	MD5
SSL_RSA_EXPORT_WITH_DES40_CBC_SHA	RSA	DES40 CBC	SHA
SSL_DH_anon_EXPORT_WITH_RC4_40_MD5	DH anon	RC4 40	MD5
SSL_DH_anon_EXPORT_WITH_DES40_CBC_SHA	DH anon	DES40 CBC	SHA

To specify cipher suites for the client:

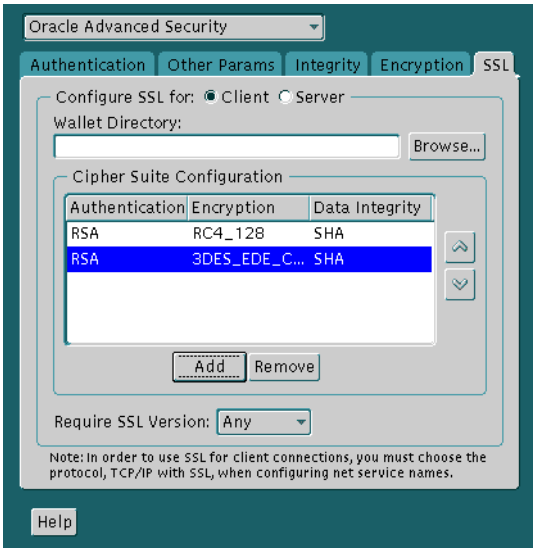
1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the Navigator window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security;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



4. Choose the SSL tab.
5. Select Configure SSL for Client.
6. Choose the Add button; a secondary dialog box that lists available cipher suites appears:



7. Select a suite and choose OK; the Cipher Suite Configuration list is updated.



8. Use the up and down arrows to prioritize the cipher suites.
9. Choose File > Save Network Configuration.

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

   `SSL_CIPHER_SUITES= (SSL_cipher_suite1 [,SSL_cipher_suite2])`

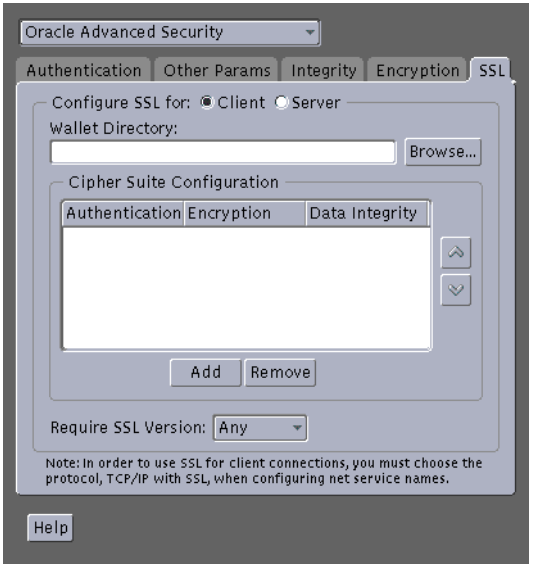
Step 4: Set the Required SSL Version (Optional)

You can set the SSL_VERSION parameter in the `sqlnet.ora` file. This parameter defines the version of SSL that must run on the systems with which the client communicates. You can require these systems to use SSL 3.0, or any valid, future version. The default setting for this parameter in `sqlnet.ora` is 0; in 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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the SSL tab.
5. Select Configure SSL for Client.
6. In the Require SSL Version scroll box the default is *Any*; accept this default or select the SSL version you want to configure.
7. Choose File > Save Network Configuration.

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

   ```
   SSL_VERSION=UNDETERMINED
   ```

Step 5: Set SSL as an Authentication Service (Optional)

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

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

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

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

- 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, do not set this parameter.

Step 6: Create a Net Service Name that Uses TCP/IP with SSL in the Connect Descriptor

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

See Also: *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 database server and on the Oracle client for all SSL parameters except the location of the Oracle wallet. To configure SSL on the server, perform these steps:

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

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

Step 1: Confirm Wallet Creation

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

See Also:

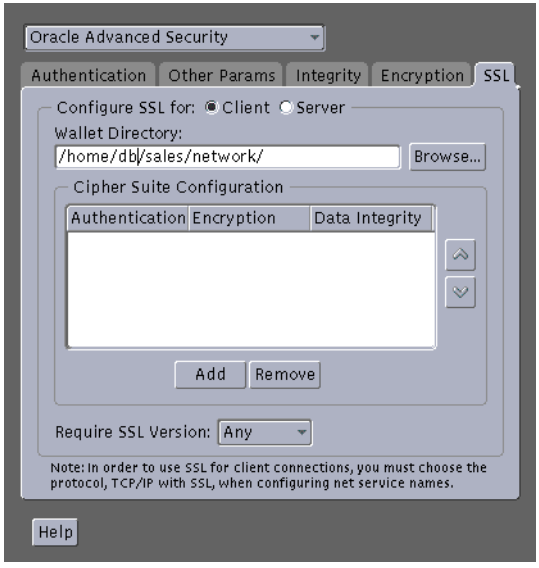
- Chapter 18, Using Oracle Wallet Manager, for general information about wallets
- Opening an Existing Wallet on page 18-5, for information about opening an existing wallet
- Creating a New Wallet on page 18-4, for information about creating a new wallet

Step 2: Specify Required Server Configuration (Wallet Location)

To specify required configuration parameters for the server:

1. Start 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 window, expand Local > Profile.

3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the SSL tab.
5. Select Configure SSL for Server.
6. In the Wallet Directory box, enter the directory in which the Oracle wallet is located, or click the Browse button to find it by searching the file system.

Important: There are two occasions during the client and the server configuration process when you set the location of the Oracle wallet. *Be sure to enter the same location on both occasions.*

- On the occasion described in this section, you set the location of the wallet either by using the Net8 Assistant or by modifying the `sqlnet.ora` file.
- Later, you use the Oracle Wallet Manager. See: Step 1: Create a Database Wallet on page 17-42.

---

7. Choose File > Save Network Configuration.

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

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

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

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

When you install Oracle Advanced Security, several SSL cipher suites are set for you by default. You can override the default by setting the SSL_CIPHER_SUITES parameter. For example, if you use 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 server must be compatible with those required by the client.

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

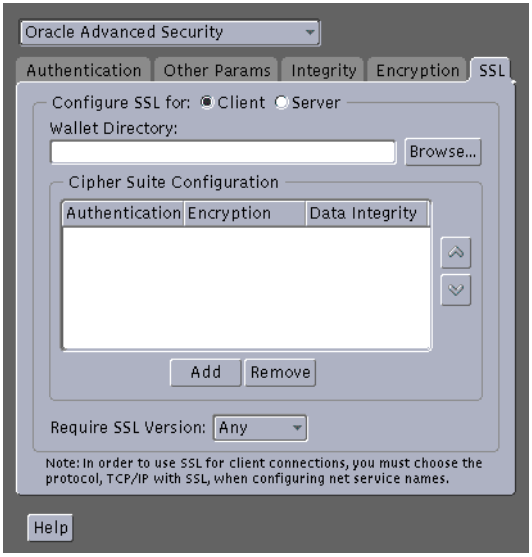
Note: In Oracle Advanced Security Release 8.1.7, if you set a cipher suite employing Diffie-Hellman anonymous authentication on the server, you must also set the same cipher suite on the client. Otherwise, the connection fails.

If you use a cipher suite employing Diffie-Hellman *anonymous*, you must set the SSL_CLIENT_AUTHENTICATION parameter to FALSE. See: Step 5: Set SSL Client Authentication (Optional) on page 9-25.

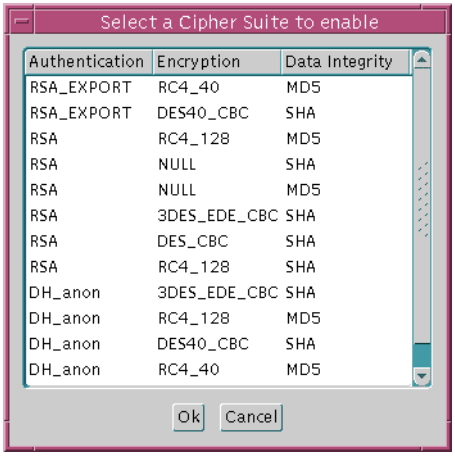
Table 9–1 lists the SSL cipher suites supported in the current release of Oracle Advanced Security. These cipher suites are set by default when you install Oracle Advanced Security. This table also lists the authentication, encryption, and data integrity types each cipher suite uses.

To specify cipher suites for the server:

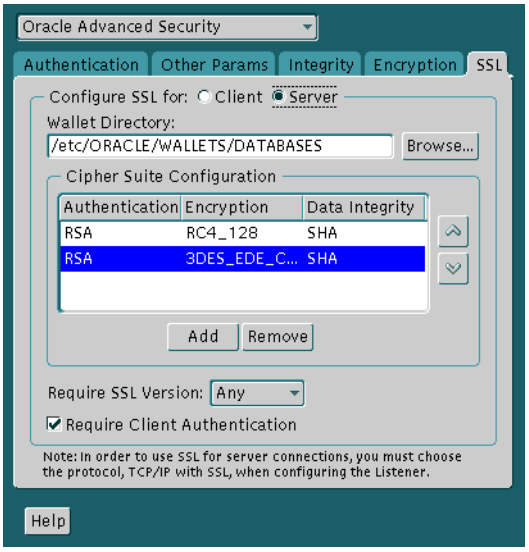
1. Start Net8 Assistant:
   - On UNIX, run `netasst` from `$ORACLE_HOME/bin`.
   - On Windows NT, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Net8 Assistant.
2. In the Navigator window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security;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



4. Choose the SSL tab.
5. Select Configure SSL for Server.
6. Choose the Add button; a secondary dialog box that lists available cipher suites appears:



7. Selecting a suite and choose OK; the Cipher Suite Configuration list is updated.



8. Use the up and down arrows to prioritize the cipher suites.
9. Choose File > Save Network Configuration.

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

   `SSL_CIPHER_SUITES= (SSL_cipher_suite1 [,SSL_cipher_suite2])`

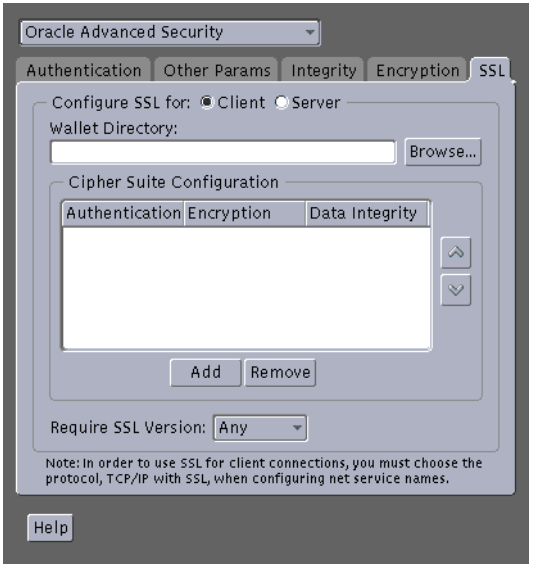
Step 4: Set the Required SSL Version (Optional)

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

To set the SSL version 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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the SSL tab.
5. Select Configure SSL for Server.
6. In the Require SSL Version scroll box the default is *Any*; accept this default or select the SSL version you want to configure.
7. Choose File > Save Network Configuration.

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

```
SSL_VERSION=UNDETERMINED
```

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

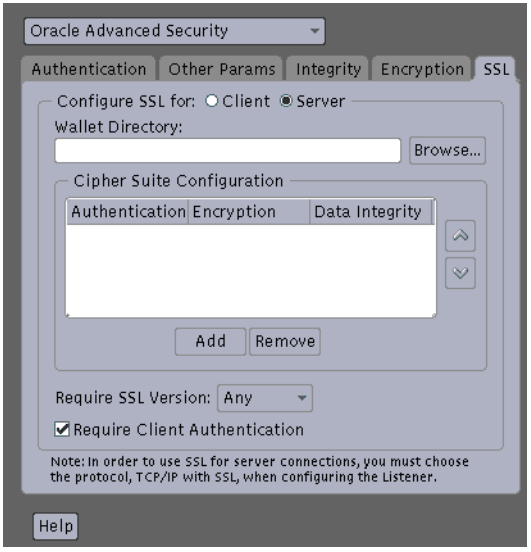
Step 5: Set SSL Client Authentication (Optional)

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

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

To set this parameter to FALSE:

1. Start 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 window, expand Local > Profile.
3. From the list in the right pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



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

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

```
SSL_CLIENT_AUTHENTICATION=FALSE
```

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

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

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

- You want to use SSL authentication in conjunction with another authentication method supported by Oracle Advanced Security. For example, you want the server to authenticate itself to the client by using SSL and the client to authenticate itself to the server by using Kerberos or SecurID.
- 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, do not set this parameter.

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

Configure the listener with a TCP/IP with SSL listening endpoint in the listener.ora file. Oracle Corporation recommends a port number 2484 for typical Net8 clients and 2482 for client connections to Oracle8*i* JServer.

See Also: *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
```

10

Configuring Entrust-Enabled SSL Authentication

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

- Overview
- System Components
- Entrust Authentication Process
- Enabling Entrust Authentication
- Issues and Restrictions

Overview

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

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

This section describes how PKI elements are provided by the following:

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

Oracle Advanced Security

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

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

Entrust/PKI

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

Entrust-Enabled Oracle Advanced Security

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

Entrust-enabled Oracle Advanced Security provides:

- ❏ Enhanced X.509-Based Authentication and Single Sign-On
- ❏ Integration with Entrust/PKI Key Management
- ❏ Integration with Entrust/PKI Certificate Revocation

Note: This release of Oracle Advanced Security is undergoing Entrust-Ready certification by Entrust Technologies, Inc. To check the status of Entrust-Ready certification of Oracle Advanced Security, refer to the Entrust Web site:

`http://www.entrust.com`

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

Entrust-enabled Oracle Advanced Security supports the use of Entrust credentials for X.509-based authentication and single sign-on. Instead of using an Oracle wallet to hold user PKI credentials, Oracle Advanced Security can access PKI credentials created by Entrust/Authority and held in an Entrust profile (an `.epf` file). Users who have deployed Entrust software within their enterprise are thus able to use it for authentication and single sign-on to Oracle8*i*.

Integration with Entrust/PKI Key Management

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

Integration with Entrust/PKI Certificate Revocation

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

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

System Components

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

- Entrust/PKI 5.0.2 for Oracle
- Entrust/Toolkit Server Login
- Entrust IPSEC Negotiator Toolkit

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

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

`http://www.entrust.com`

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

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

The URLs for downloading the Entrust/Toolkit Server Login and Entrust IPSEC Negotiator Toolkit from the Entrust Developer Network are provided in the related sections.

Entrust/PKI 5.0.2 for Oracle

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

Entrust/PKI 5.0.2 for Oracle is comprised of the following software components:

- Entrust/Authority
- Entrust/RA
- Entrust/Entelligence

Entrust/Authority

Entrust/Authority is the centerpiece of Entrust/PKI. It performs core certificate authority, certificate, and user management functions, such as creating users and user profiles containing the user's credentials.

Note: Oracle Corporation only supports the use of Entrust-enabled Oracle Advanced Security with versions of Entrust/Authority that run on Oracle8i.

See Also: Chapter 9, Configuring Secure Socket Layer Authentication, for information about certificate authorities.

Entrust/Authority supports unattended login, also called Server Login, which eliminates the need for a **Database Administrator** (DBA) to repeatedly enter a password for the Entrust profile on the server. With unattended login, the DBA need only enter a password once to open the Entrust profile for the server to authenticate itself to multiple incoming connections.

Entrust/RA

Entrust/RA is the administrator's secure interface to Entrust/Authority.

Entrust/Entelligence

Entrust/Entelligence provides support for user key management and single sign-on functionality on both clients and server by enabling Oracle8*i* server process access to incoming SSL connections.

Entrust/Toolkit Server Login

Entrust/Toolkit Server Login Toolkit Release 5.0.2 is required for single sign-on functionality on servers operating on UNIX platforms.

Entrust/Server Login Toolkit provides single sign-on by enabling Oracle8*i* server process access to incoming SSL connections. Without this capability, a database administrator or other privileged user would have to enter the password for the Entrust profile on the server for every incoming connection.

Entrust Developer Network members can download Entrust/Toolkit Server Login from the Entrust Web site:

http://developer.entrust.com/software/files/desc_serverlogin.cfm

Entrust IPSEC Negotiator Toolkit

The Entrust IPSEC Negotiator Toolkit Release 5.0.2 is required on both clients and servers for integrating the Oracle Advanced Security SSL stack with Entrust/PKI, enabling SSL authentication to use Entrust profiles.

Entrust Developer Network members can download the IPSEC Negotiator Toolkit from the Entrust Web site:

http://developer.entrust.com/software/index.htm

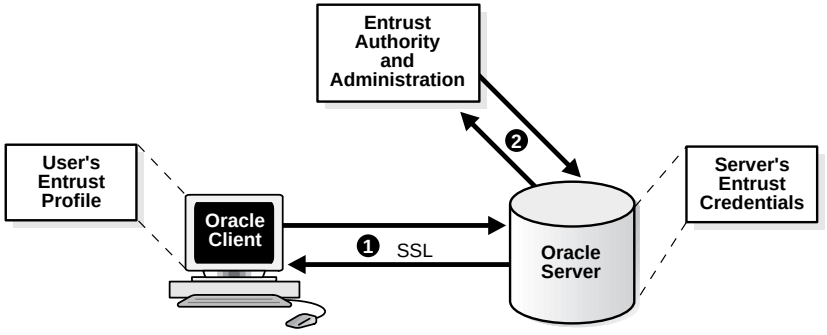
Entrust Authentication Process

Figure 10–1 illustrates the following Entrust authentication process:

1. The Entrust user on the Oracle client establishes a secure connection with the server using SSL and Entrust credentials.
2. The Oracle SSL adapter on the server communicates with the Entrust Authority to check the certificate revocation status of the Entrust user.

Note: Figure 10–1 does not include client and server profiles creation, which is presumed.

Figure 10–1 Entrust Authentication Process



See Also: How SSL Works in an Oracle Environment: The SSL Handshake on page 9-6

Enabling Entrust Authentication

This section describes the following tasks that enable Entrust-enabled Oracle Advanced Security SSL authentication:

- ❏ Creating Entrust Profiles
- ❏ Installing Oracle Advanced Security and Related Products
- ❏ Configuring SSL on the Client and Server
- ❏ Configuring Entrust on the Client
- ❏ Configuring Entrust on the Server
- ❏ Creating Database Users

Creating Entrust Profiles

This section describes how to create Entrust profiles. Entrust profiles can be created by either administrators or users.

Administrator-Created Entrust Profiles

Administrators create Entrust profiles as follows:

1. The Entrust administrator adds the Entrust user using the New User dialog box with the Create Profile option selected.

   See Also: The Entrust administration documentation for information on creating Entrust Users

2. The administrator enters the user's name and password.
3. The Entrust Authority creates the profile, or `.epf` file.
4. The administrator securely sends all profile-related files to the user with a preset password.

User-Created Entrust Profiles

Entrust users create their own Entrust profiles as follows:

1. The Entrust administrator adds the Entrust user using the New User dialog box with the Create Profile option deselected.
2. The user receives a secure e-mail notification from the administrator that contains a reference number, authorization code, and expiration date.

3. The user navigates to the Create Entrust Profiles screen in Entrust/Entelligence as follows:

   Start>Programs>Entrust>Entrust Profiles>Create Entrust Profiles

4. The user enters the reference number, authorization code, and expiration date provided in the e-mail notification, creating a profile, or `.epf` file, and the Entrust initialization file.

Installing Oracle Advanced Security and Related Products

Use the Custom installation option to install Oracle Advanced Security on both the client and server. Select the Entrust adapter from the Authentication Methods screen.

> **See Also:** The Oracle8*i* installation documentation for your platform.

Note: After the Entrust adapter is installed, the Net8 SSL protocol adapter does not operate with Oracle wallets.

Configuring SSL on the Client and Server

Configure SSL on the client and server.

> **See Also:** Chapter 9, Configuring Secure Socket Layer Authentication, for information about configuring SSL on the client and server; skip the section that describes the Oracle wallet location.

Configuring Entrust on the Client

The steps for configuring Entrust on the client vary according to the type of platform:

- Configuring Entrust on a UNIX Client
- Configuring Entrust on a Windows Client

Configuring Entrust on a UNIX Client

If the client resides on a non-Windows platform, perform the following steps:

1. Set the JAVA_HOME variable to JDK or JRE location.

   For example:

   ```
   >setenv JAVA_HOME $ORACLE_HOME/JRE
   ```

2. Either set the CLASSPATH environment variable or set the `ssl_entrust_classpath` parameter in the `sqlnet.ora` file to specify the required jar files for displaying the graphic user interface.

   For example:

   Set the CLASSPATH environment variable as follows:

   ```
   >setenv CLASSPATH $ORACLEHOME/JRE/lib/rt.jar:$ORACLE_
   HOME/JRE/lib/i18n.jar
   ```

3. Use SQL*Plus to connect to the Oracle instance as follows:

   ```
   sqlplus /@tns_service_name
   ```

   where `tns_service_name` is the service name of the Oracle instance.

   The Entrust_Login dialog box appears.

4. Enter the path to the profile and the password, as well as the path to the Entrust initialization file.

Configuring Entrust on a Windows Client

If the client resides on a Windows platform, ensure that the Entrust/Entelligence component is installed on the client and perform the following steps.

1. Choose the Entrust icon on the system tray to open the Entrust_Login dialog box.
2. Log on to Entrust by entering the profile name and password.
3. Use SQL*Plus to connect to the Oracle instance as follows:

   ```
   sqlplus /@tns_service_name
   ```

   where `tns_service_name` is the service name of the Oracle instance.

Configuring Entrust on the Server

The steps for configuring Entrust on the client vary according to the type of platform:

- ❏ Configuring Entrust on a UNIX Server
- ❏ Configuring Entrust on a Windows Server

Configuring Entrust on a UNIX Server

If the server is a UNIX platform, ensure that the Entrust/Server Login Toolkit component is installed on the server and perform the following steps:

> **See Also:** System Components on page 10-5 for information about downloading the Entrust/Toolkit Server Login.

1. Stop the Oracle database instance.
2. Set the `ssl_entrust_profile` and `ssl_entrust_ini_file` parameters in the `sqlnet.ora` and `listener.ora` files to specify the paths to the server's profile and the Entrust initialization file.
3. Enter the binder command to create unattended login credentials, or `.ual` files.

   For example:

   ```
   binder_sun
   ```

4. Enter the path to the profile, the password, and the path to the Entrust initialization file. A message informs you that you have successfully created a credential file.
5. Start the Oracle database instance.

Configuring Entrust on a Windows Server

If the server is a Windows platform, ensure that the Entrust/Entelligence component is installed on the client and perform the following steps:

> **See Also:** System Components for information on downloading Entrust/Entelligence.

1. Stop the Oracle database instance.
2. Choose the Entrust icon on the system tray to open the Entrust_Login dialog box.
3. Log on to Entrust by entering the profile name and password.
4. Start the Oracle database instance.

Creating Database Users

Create global user in the database based on the **distinguished name (DN)** of each Entrust user.

For example:

```
SQL> create user jdoe identified globally as
'cn=jdoe,o=oracle,c=us';
```

where `"cn=jdoe, o=oracle, c=us"` is the Entrust distinguished name of the user.

> **See Also:** Chapter 17, Configuring Entrust-Enabled SSL Authentication, for information about creating database users.

Issues and Restrictions

The Entrust-ready designation from Entrust typically requires that a partner product integration with Entrust is done using an Entrust toolkit. This means that an application must be specifically modified to work with Entrust.

For example, Oracle has modified its SSL libraries to access an Entrust profile instead of an Oracle wallet. Accordingly, the Entrust profile is not accessible from standard SSL libraries.

In addition, the following restrictions apply:

- The use of Entrust components for digital signatures in applications based on Oracle is not supported.
- The Entrust-enabled Oracle Advanced Security integration is only supported with versions of Entrust/PKI Release 5.0.2 running on Oracle8*i*.
- The use of earlier releases of Entrust/Authority with Entrust-enabled Oracle Advanced Security is not supported.
- Interoperability between Entrust and non-Entrust PKIs is not supported.
- Entrust has certified Oracle Internet Directory version 2.1 for 8.1.7

11

Configuring Multiple Authentication Methods

This chapter describes how to configure multiple authentication methods under Oracle Advanced Security Release 8.1.7, and how to use conventional user name and password authentication, even if you have configured another authentication method. In addition, this chapter describes how to configure your network so that Oracle clients can use a specific authentication method, and Oracle servers can accept any method specified.

This chapter contains the following topics:

- Connecting with User Name and Password
- Disabling Oracle Advanced Security Authentication
- Configuring Multiple Authentication Methods
- Configuring Oracle8i for External Authentication

Connecting with User Name and Password

To connect to an Oracle database server using a user name and password when an Oracle Advanced Security authentication method has been configured, disable the external authentication (See: Disabling Oracle Advanced Security Authentication, below).

With the external authentication disabled, a user can connect to a database using the following format:

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

For example:

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

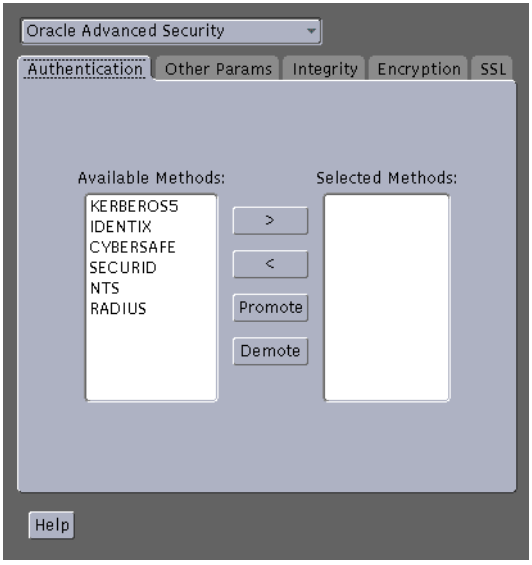
Disabling Oracle Advanced Security Authentication

To disable authentication methods:

1. Start Net8 Assistant:
   - On UNIX:

     Run `netasst` from $ORACLE_HOME/bin
   - On Windows NT:

     Select Start>Programs>Oracle-HOME_NAME>Network Administration>Net8 Assistant
2. In the Navigator window, expand Local > Profile.
3. From the list in the right window pane, select Oracle Advanced Security; the Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. Sequentially move all authentication methods from the Selected Method list to the Available Methods list by selecting a method and choosing the left arrow [<].
6. Choose File > Save Network Configuration.

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

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

Configuring Multiple Authentication Methods

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

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

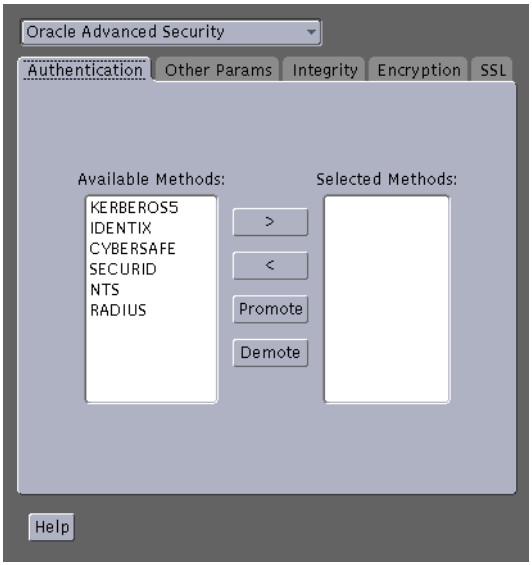
To add authentication methods to both clients and servers:

1. Start Net8 Assistant:
   - On UNIX:

     Run `netasst` from $ORACLE_HOME/bin
   - On Windows NT:

     Select Start>Programs>Oracle-HOME_NAME>Network Administration>Net8 Assistant
2. In the Navigator window, expand Local > Profile.
3. From the list in the right window pane, select Oracle Advanced Security.

The Oracle Advanced Security tabbed window appears:



4. Choose the Authentication tab.
5. Select a method listed in the Available Methods list.
6. Sequentially move selected methods to the Selected Methods list by choosing the right arrow [>].
7. Arrange the selected methods in order of desired use. To do this, select a method in the Selected Methods list, and choose Promote or Demote to position it in the list.
8. Choose File > Save Network Configuration.

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

```
SQLNET.AUTHENTICATION_SERVICES =
(RADIUS|CYBERSAFE|KERBEROS5|SECURID|IDENTIX)
```

Configuring Oracle8*i* for External Authentication

This section describes the parameters you must set to configure Oracle8*i* for network authentication, using the following tasks:

- ❏ Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora
- ❏ Verifying that REMOTE_OS_AUTHENT Is Not Set to TRUE
- ❏ Setting OS_AUTHENT_PREFIX to a Null Value

See Also:

- The corresponding chapter in this guide for information about configuring a particular authentication method
- Appendix B, Authentication Parameters

Setting the SQLNET.AUTHENTICATION_SERVICES Parameter in sqlnet.ora

The following parameter must be set in the `sqlnet.ora` file for all clients and servers to enable each to use a supported authentication method:

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

For example, for all clients and servers using Kerberos authentication, the `sqlnet.ora` parameter must be set as follows:

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

Verifying that REMOTE_OS_AUTHENT Is Not Set to TRUE

To verify that REMOVE_OS_AUTHENT is not set to TRUE, add the following parameter to the initialization file—in each database instance—when you configure the authentication method:

```
REMOTE_OS_AUTHENT=FALSE
```

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

If REMOTE_OS_AUTHENT is set to FALSE, and the server cannot support any of the authentication methods requested by the client, the authentication service negotiation fails and the connection terminates.

If the parameter is set as follows in the `sqlnet.ora` file on either the client or server, the database attempts to use the supplied user name and password to login the user:

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

If REMOTE_OS_AUTHENT is set to FALSE, however, the connection fails.

Setting OS_AUTHENT_PREFIX to a Null Value

Authentication service-based user names can be long, and Oracle user names are limited to 30 characters. Oracle Corporation strongly recommends that you enter a null value for the OS_AUTHENT_PREFIX parameter in the initialization file used for the database instance as follows:

```
OS_AUTHENT_PREFIX=""
```

Note: The default value for OS_AUTHENT_PREFIX is OPS$; however, you can set it to any string.

Attention: If a database already has the OS_AUTHENT_PREFIX set to a value other than NULL (" "), *do not change it*, since it can inhibit previously created, externally identified users from connecting to the Oracle server.

To create a user, launch SQL*Plus and enter the following:

```
SQL> CREATE USER os_authent_prefix username IDENTIFIED EXTERNALLY;
```

When OS_AUTHENT_PREFIX is set to a null value (" "), enter the following to create the user king:

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

The advantage of creating a user in this way is that the administrator no longer needs to maintain different user names for externally identified users. This is true for all supported authentication methods.

See Also:

- *Oracle8i Administrator's Guide*
- *Oracle8i Distributed Database Systems*

Part IV

Oracle DCE Integration

This part describes Oracle Distributed Computing Environment Integration (DCE), in the following sections:

- Chapter 12, Overview of Oracle DCE Integration
- Chapter 13, Configuring DCE for Oracle DCE Integration
- Chapter 14, Configuring Oracle8i 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 verify that Oracle Advanced Security supports Oracle DCE integration on your platform.

12

Overview of Oracle DCE Integration

Oracle DCE Integration enables Oracle applications and tools to access Oracle8*i* servers in a distributed computing environment. This chapter briefly describes the Distributed Computing Environment (DCE) and the Oracle DCE Integration product. It contains the following sections:

- Oracle DCE Integration Requirements
- The Distributed Computing Environment
- Components of Oracle DCE Integration
- Flexible DCE Deployment
- Release Limitations

See Also: Related Documents on page -xxv.

Oracle DCE Integration Requirements

System Requirements

Oracle DCE Integration requires Net8 and Oracle8*i*. It is based on the Open Software Foundation (OSF) DCE protocol (V1.1 and later).

Note that OSF has merged with X/OPEN, another standards group, to form The Open Group. This group is committed to continuing DCE support.

Backward Compatibility

Oracle servers running DCE Integration 2.3.2 and later are backward compatible with clients running SQL*Net/DCE 2.1.6 or 2.2.3; however, Release 2.1.6 clients cannot take advantage of external roles.

A client running DCE Integration 2.3.2 or later cannot connect to a SQL*Net/DCE 2.1.6 or 2.2.3 server. A DCE Integration Release 2.3.2 or later client requires a Release 2.3.2 or later server in order to connect to a database.

The Distributed Computing Environment

The Distributed Computing Environment (DCE) from the Open Group is a set of integrated network services that works across multiple systems to provide a distributed environment. The network services include remote procedure calls (RPCs), directory service, security service, threads, distributed file service, diskless support, and distributed time service.

DCE is the middleware between distributed applications and the operating system / network services and is based on a client / server model of computing. By using the services and tools that DCE provides, users can create, use, and maintain distributed applications that run across a heterogeneous environment.

Components of Oracle DCE Integration

Oracle DCE Integration has two components: DCE Communication/Security and DCE CDS Native Naming.

- DCE Communication/Security
- DCE Cell Directory Services Native Naming

DCE Communication/Security

This component has three principal features:

Authenticated RPC

Oracle DCE Integration provides authenticated Remote Procedure Call (RPC) as the transport mechanism that enables multi-vendor interoperability. RPC also uses some of the other DCE services, including directory and security services, to provide location transparency and secure distributed computing.

Integrated Security and Single Sign-On

Oracle DCE Integration works with the DCE Security service to provide security within DCE cells. It enables a user logged onto DCE to securely access any Oracle database without having to specify a user name or password. This is sometimes called **external authentication** to the database, or **single sign-on**. Clients and servers that are not running DCE authentication services can interoperate with systems that have DCE security by specifying an Oracle password.

Data Privacy and Integrity

Oracle DCE Integration uses the multiple levels of security that DCE provides to ensure data authenticity, privacy, and integrity. Users have a range of choices, from no protection to full encryption for each connection, with a guarantee that no data is modified in transit.

Note: For parts of the network that do not use DCE, you can use the other security and authentication services that are part of Oracle Advanced Security. These services work with SQL*Net release 2.1 and above or with 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 Cell Directory Services Native Naming

The DCE Cell Directory Service (CDS) Native Naming component includes naming and location transparency.

DCE Integration registers Oracle8*i* connect descriptors in the DCE 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)

See Also:

- 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 Oracle8i 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).
- 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.
- 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 Release 8.1.7 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 how to configure the Distributed Computing Environment (DCE) to use Oracle DCE Integration—after Oracle DCE Integration has been installed.

See Also: Chapter 12, Overview of Oracle DCE Integration

To Configure DCE for Oracle DCE Integration:

The following tasks, performed by the DCE cell administrator, assume that a DCE cell has been configured and the systems being used are part of that cell:

- ❏ Task 1: Create New Principals and Accounts
- ❏ Task 2: Install the Key of the Server into a Keytab File
- ❏ Task 3: Configure DCE CDS for Use by Oracle DCE Integration

Task 1: Create New Principals and Accounts

Use the following procedure model to add server principals:

```
% dce_login cell_admin password
% rgy_edit
Current site is: registry server at /.../cell1/subsys/dce/sec/master
rgy_edit=>do p
Domain changed to: principal
rgy_edit=> add oracle
rgy_edit=> do a
Domain changed to: account
rgy_edit=> add oracle -g none -o none -pw oracle_password -mp cell_admin_
password
rgy_edit=> quit
bye
```

In this example, a DCE principal named `oracle` is created. The principal has a corresponding account with a password set to *password*. The account does not belong to any DCE group or DCE profile.

Note: Perform this task on the server only once after DCE Integration has been installed; *do not perform this task on client systems.*

Task 2: Install the Key of the Server into a Keytab File

Install the key of the server into a keytab file, `dcepa.key`. This file contains the password of the principal under which the Net8 listener starts. The Net8 listener reads this file to authenticate itself to DCE. To generate the keytab file, enter the following:

```
% dce_login cell_admin password
% rgy_edit
```

```
Current site is: registry server at /.../cell1/subsys/dce/sec/master
rgy_edit=> ktadd -p oracle -pw Oracle_password -f
$ORACLE_HOME/dcepa/admin/dcepa.key
rgy_edit=>quit
bye
```

Note:

- Perform this task on the server only once after DCE Integration has been installed; *do not perform this task on client systems.*
- Remember to substitute the full pathname for the $ORACLE_HOME variable. If the specified directories do not exist, create them before running the command; to create the directories. enter the following:

  ```
  mkdir $ORACLE_HOME/dcepa
  mkdir $ORACLE_HOME/dcepa/admin
  ```

Task 3: Configure DCE CDS for Use by Oracle DCE Integration

Step 1: Create Oracle Directories in the CDS Namespace

Enter the following after installing DCE Integration for the first time in a cell; create directories on all CDS replicas:

```
% dce_login cell_admin

Enter Password:(password not displayed)
$ cdscp
cdscp> create dir /.:/subsys/oracle
cdscp> create dir /.:/subsys/oracle/names
cdscp> create dir /.:/subsys/oracle/service_registry
cdscp> exit
```

Note:

- The directory `/.:/subsys/oracle/names` contains objects that map Net8 service names to connect descriptors, which are used by the CDS naming adapter.
- The directory `/.:/subsys/oracle/service_registry` contains objects that map the service name in DCE addresses to the network endpoint that is used by both DCE protocol adapter clients and servers.

Step 2: Give Servers Permission to Create Objects in the CDS Namespace

Enter the following to add the principal `oracle` to the CDS-server group:

```
$ dce_login cell_admin
Enter Password:   (password not displayed)
$ rgy_edit
rgy_edit=> domain group
Domain changed to: group
rgy_edit=> member subsys/dce/cds-server -a oracle
rgy_edit=> exit
```

Step 3: Load Oracle Service Names into CDS

Load Oracle service names into the Cell Directory Service, as described in Chapter 14, **Configuring Oracle8i for Oracle DCE Integration**.

14

Configuring Oracle8*i* for Oracle DCE Integration

This chapter describes how to configure Oracle8*i* and Net8 to use Oracle DCE Integration after it has been successfully installed.

DCE Address Parameters

DCE addresses in the `listener.ora` and `tnsnames.ora` configuration files are defined by DCE parameters, illustrated below:

ADDRESS=(PROTOCOL=DCE)(SERVER_PRINCIPAL=*server_name*)(CELL_NAME=*cell_name*)
(SERVICE=*dce_service_name*))

These parameters are described by Table 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
```

Alternatively, you can specify:

```
SERVICE=dce_service_name
```

provided that `CELL_NAME=cell_name` is also specified.

In this case, the cell name defaults to the local cell. However, this way of specifying service names only works if you are operating within a single cell.

Note: The *dce_service_name* in the service field might not be the same as that used by 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.

Configuring Oracle 8*i* and Net8:

To configure Oracle 8*i* and Net8 to use Oracle DCE Integration, perform the following tasks:

- ❏ Task 1: Configure the Server
- ❏ Task 2: Create and Name Externally-Authenticated Accounts
- ❏ Task 3: Set up DCE Integration External Roles
- ❏ Task 4: Configure DCE for SYSDBA and SYSOPER Connections to Oracle Databases
- ❏ Task 5: Configure the Client
- ❏ Task 6: Configure Clients to Use DCE CDS Naming

Task 1: Configure the Server

To configure a server for DCE Integration, do the following:

1. Configure the listener configuration file (`listener.ora`) with DCE address information for all servers.
2. For servers in distributed systems that require database link connections to other servers, configure the `sqlnet.ora` and `protocol.ora` files with DCE address information.

Note: In this release, the configuration files `listener.ora`, `sqlnet.ora`, `tnsnames.ora`, and `protocol.ora` are located in the $ORACLE_HOME/network/admin directory.

For a database server to receive connections from Net8 clients in a DCE environment, there must be a Net8 listener active on the server platform. This process listens for connections on a network address that is defined in the `listener.ora` configuration file.

The SERVER_PRINCIPAL parameter designates what DCE principal the listener should be running under. In the sample below, the listener is running under principal oracle.

The following is a sample DCE address as it would appear in the `listener.ora` file.

```
LSNR_DCE=
  (ADDRESS=
       (PROTOCOL=DCE)
       (SERVER_PRINCIPAL=oracle)
       (CELL_NAME=cell1)
       (SERVICE=dce_svc))
SID_LIST_LSNR_DCE=
       (SID_DESC=
       (SID_NAME=ORASID)
        (ORACLE_HOME=/private/oracle8))
```

Task 2: Create and Name Externally-Authenticated Accounts

To use DCE authentication for logging onto an Oracle database, you must create database accounts that are authenticated externally. To enable secure external authentication, do the following:

> **Note:** The privileges shown in this this section are the *minimum access privileges necessary*. The actual set of privileges needed depends upon the instance or application.

1. Verify that these lines are in the initialization parameter file:

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

2. Verify that the initialization parameter file does not have a multi-threaded server (MTS) entry for DCE. For example, an entry such as the following is not allowed:

   ```
   mts_dispatchers="(PROTOCOL=dce)(DISPATCHERS=3)"
   ```

3. Ensure that you are logged on as a member of the DBA group. Restart the database instance for the changes to take effect.

4. At the SQL*Plus prompt, define users. Before doing so, decide whether you are, or ever will be, operating in a multi-cell DCE environment in which you 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.

Local Cell:

If users are connecting within a local cell, use the following format:

```
SQL> CREATE USER server_principal IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO server_principal;
```

For example:

```
SQL> CREATE USER oracle IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO oracle;
```

The entire CELL_NAME/SERVER_PRINCIPAL string must be 30 characters or les *(this is an Oracle8i restriction—not a restriction of the DCE adapter).*

For example:

```
SQL> CREATE USER "CELL1/ORACLE" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL1/ORACLE";
```

Multiple Cells:

If connecting to the database across multiple cells, specify both the *cell_name* and the *server_principal*, as illustrated below:

```
SQL> CREATE USER "CELL_NAME/SERVER_PRINCIPAL" IDENTIFIED
EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL_NAME/SERVER_PRINCIPAL";
```

You must enclose the externally-identified account name in double quotation marks, because the slash is a reserved character. Also, if the account (user) name is double-quoted, it must be capitalized.

For example:

```
SQL> CREATE USER "CELL1/ORACLE" IDENTIFIED EXTERNALLY;
SQL> GRANT CREATE SESSION TO "CELL1/ORACLE";
```

When using this format, set the following parameter in the `protocol.ora` configuration file to FALSE:

```
dce.local_cell_usernames=false
```

References to an Oracle account created in this manner must include the schema / account in the correct format. Consider requests for access to tables from another account. When a user references the tables in another account created within a local cell, the command might appear as follows:

```
SQL> SELECT * FROM oracle.emp
```

If a user wants to access tables in another account created for connections across cells, the command might appear as follows:

```
SQL> SELECT * FROM "CELL1/ORACLE" .emp
```

See Also: *Oracle8i Distributed Database Systems,* for more information about external authentication

Task 3: Set up DCE Integration External Roles

To set up external roles for DCE Integration, and enable connection to an Oracle database as SYSOPER or SYSDBA with DCE credentials, do the following:

1. Set the following parameter in the initialization parameter file:

   ```
   OS_ROLES=TRUE
   ```

2. Restart the database.

3. Ensure that the DCE groups that map to Oracle roles adhere to the following syntax:

   ```
   ORA_global_name_role[_[a][d]]
   ```

   Table 14–2 describes 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

Table 14–2 Setting Up External Role Syntax Components

Component	Definition
A or a	Optional character indicating that the user has admin privileges for this role
D or d	Optional character indicating the role is to be enabled by default at connect time

See Also: *Oracle8i Administrator's Guide for more information about external roles*

4. Authenticate to DCE a user who is a member of a DCE group by entering the following commands:

```
dce_login
klist
```

Sample Output:

```
% dce_login oracle
Enter Password:
% klist
dce identity information:
   Warning: Identity information is not certified
   Global Principal: /.../ilab1/oracle
   Cell:      001c3f90-01f5-1f72-ba65-02608c2c84f3
   /.../ilab1
   Principal: 00000068-0568-2f72-bd00-02608c2c84f3 oracle
   Group:     0000000c-01f5-2f72-ba01-02608c2c84f3 none
   Local Groups:
   0000000c-01f5-2f72-ba01-02608c2c84f3 none
   0000006a-0204-2f72-b901-02608c2c84f3
   subsys/dce/cds-server
   00000078-daf4-2fe1-a201-02608c2c84f3 ora_dce222_dba
```

```
   00000084-89c8-2fe8-a201-02608c2c84f3 ora_dce222_connect_
   d
00000087-8a13-2fe8-a201-02608c2c84f3 ora_dce222_resource_d
00000080-f681-2fe1-a201-02608c2c84f3 ora_dce222_role1_ad
.
.
.
```

5. Connect to the database as usual.

The following sample output lists external roles (DBA, CONNECT, RESOURCE, and ROLE1) that have been mapped to DCE groups:

```
SQL> SELECT * FROM session_roles;

ROLE
------------------------------
CONNECT
RESOURCE
ROLE1

SQL> SET ROLE all;

Role set.

SQL> SELECT * FROM session_roles;

ROLE
------------------------------
DBA
EXP_FULL_DATABASE
IMP_FULL_DATABASE
CONNECT
RESOURCE
ROLE1

6 rows selected.
```

```
SQL> EXIT
```

Task 4: Configure DCE for SYSDBA and SYSOPER Connections to Oracle Databases

To configure DCE so that you can connect to an Oracle database as SYSOPER or SYSDBA with DCE credentials, do the following:

1. Create DCE groups that map to Oracle DBA and OPERATOR roles. DCE group names should adhere to the syntax described by Task 3: Set up DCE Integration External Roles on page 14-7. Add the externally authenticated user `oracle` as a member of the group(s).

```
$ dce_login cell_admin cell_admin_password
$ rgy_edit
rgy_edit=> domain group
Domain changed to: group
rgy_edit=> add ora_dce222_dba_ad
rgy_edit=> add ora_dce222_operator_ad
rgy_edit=> member ora_dce222_dba_ad -a oracle
rgy_edit=> member ora_dce222_operator_ad -a oracle
```

2. Add the GLOBAL_NAME parameter to the DCE address or TNS service name in the local configuration file `tnsnames.ora`.

```
ORADCE=
    (ADDRESS=
               (PROTOCOL=DCE)
               (SERVER_PRINCIPAL=oracle)
               (CELL_NAME=cell1)
               (SERVICE=dce_svc))
 (CONNECT_DATA=
              (SID=ORASID)
              (GLOBAL_NAME=dce222)))
```

3. Create the database user `oracle` as described by Task 2: Create and Name Externally-Authenticated Accounts on page 14-5.

4. Get DCE credentials for the externally authenticated user:

```
$ dce_login oracle oracle_password
$klist
DCE Identity Information:
        Warning: Identity information is not certified
        Global Principal: /.../dce.dlsun685.us.oracle.com/oracle
        Cell:      00af8052-7e94-11d2-b261-9019b88baa77
```

```
/.../dce.dlsun685.us.ora
cle.com
        Principal: 0000006d-88b9-21d2-9300-9019b88baa77 oracle
        Group:     0000000c-7e94-21d2-b201-9019b88baa77 none
        Local Groups:
                0000000c-7e94-21d2-b201-9019b88baa77 none
                0000006a-7e94-21d2-ad01-9019b88baa77 subsys/dce/cds-server
                00000076-8b53-21d2-9301-9019b88baa77 ora_dce222_dba_ad
                00000077-8b53-21d2-9301-9019b88baa77 ora_dce222_operator_ad

Identity Info Expires: 1999-12-04-10:28:22
Account Expires:       never
Passwd Expires:        never

Kerberos Ticket Information:
Ticket cache: /opt/dcelocal/var/security/creds/dcecred_43ae2600
Default principal: oracle@dce.dlsun685.us.oracle.com
Server: krbtgt/dce.dlsun685.us.oracle.com@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:22 to 1999-12-04-10:28:22
Server: dce-rgy@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:22 to 1999-12-04-10:28:22
Server: dce-ptgt@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:26 to 1999-12-04-02:28:26
Client: dce-ptgt@dce.dlsun685.us.oracle.com     Server:
krbtgt/dce.dlsun685.us.o
racle.com@dce.dlsun685.us.oracle.com
        valid 1999-12-04-00:28:26 to 1999-12-04-02:28:26
Client: dce-ptgt@dce.dlsun685.us.oracle.com     Server:
dce-rgy@dce.dlsun685.us.
oracle.com
        valid 1999-12-04-00:28:27 to 1999-12-04-02:28:26
```

Note: List output shows the DCE group membership of `oracle`.

5. Connect to the Oracle database as SYSBDA or SYSOPER.

For example:

```
SQL> connect /@oradce as SYSDBA
```

Task 5: Configure the Client

To configure a client for DCE Integration, you must configure the following Net8 files with DCE address and parameter information:

- `protocol.ora`
- `sqlnet.ora`

Typically, CDS is used for name resolution. Thus, a local naming configuration file (`tnsnames.ora`) is not used, except when loading names and addresses into CDS.

Parameters in protocol.ora

There are four DCE parameters located in the `protocol.ora` file. Each parameter begins with the prefix `DCE.` to distinguish it from parameters relevant to other protocols. If default values are used for these four parameters, DCE Integration does not require a `protocol.ora` file. The parameters and their current defaults follow:

- DCE.AUTHENTICATION=*dce_secret*
- DCE.PROTECTION=*pkt_integ*
- DCE.TNS_ADDRESS_OID=1.3.22.1.5.1
- DCE.LOCAL_CELL_USERNAMES=TRUE

Configuration parameters are not case-sensitive: you can enter them in either uppercase or lowercase.

DCE.AUTHENTICATION

The DCE.AUTHENTICATION parameter is optional. It indicates the authentication value to be used for each DCE RPC. The client DCE_AUTHENTICATION value must be the same as the server DEC_AUTHENTICATION value. If this entry is not specified, cell-wide default authentication is used. The options follow:

Option	Description
NONE	No authentication
DCE_SECRET	DCE shared-secret key authentication (Kerberos)
DCE_SECRET	Default authentication level and recommended value
DEFAULT	Cell default

DCE.PROTECTION

DCE.PROTECTION is an optional field that specifies the data integrity protection levels for data transmission. The client DCE_PROTECTION level must be equal to or greater than the server DCE_PROTECTION level. If this entry is not specified, cell-wide default protection is used. The options follow:

Option	Description
NONE	Perform no protection for the current connection
DEFAULT	Use the default cell-wide protection level
CONNECT	Perform protection only when the client establishes a relationship with the server
CALL	Perform protection only at the beginning of each remote procedure call when the server receives the request
PKT	Ensure that all data received is from the expected client
PKT_INTEG	Ensure and verify that none of the data transferred between the client and server has been modified
PRIVACY	Perform protection as specified by all of the previous levels and also encrypt each RPC argument value and all user data in each call

DCE.TNS_ADDRESS_OID

DCE.TNS_ADDRESS_OID is an optional parameter that enables you to specify an alternative to the default value as follows:

```
DCE.TNS_ADDRESS_OID=1.3.22.1.x.x
```

See Also: Step 2: Modify the CDS Attributes File and Restart the CDS on page 14-15.

DCE.LOCAL_CELL_USERNAMES

DCE.LOCAL_CELL_USERNAMES is an optional parameter that defines the format used to specify the principal name (username), with or without the cell name. The choice you make for this parameter should be determined by whether or not users are making connections across cells—with unique names. The default for DCE.LOCAL_CELL_USERNAMES is now TRUE (it was set to FALSE in the DCE Integration 2.1.6 release).

The associated options follow:

Option	Description
TRUE	The default value. Select TRUE if using just the SERVER_PRINCIPAL format, without the CELL_NAME. An example of a user specified in this format is as follows: oracle TRUE is an appropriate option if users are making connections within a single cell, or if naming conventions in the network assure that users in different cells do not have duplicate names.
FALSE	Select FALSE when using the CELLNAME/SERVER_PRINCIPAL format. An example of a user specified in this format is as follows: CELL1/ORACLE FALSE is an appropriate option if users are making connections across cells and there can be users in different cells with identical name

Task 6: Configure Clients to Use DCE CDS Naming

Clients typically use CDS to resolve Oracle service names to addresses. Perform the following steps to configure CDS:

- ❏ Step 1: Enable CDS for use in Performing Name Lookup
- ❏ Step 2: Modify the CDS Attributes File and Restart the CDS
- ❏ Step 3: Create a tnsnames.ora File for Loading Oracle Connect Descriptors into CDS
- ❏ Step 4: Load Oracle Connect Descriptors into CDS
- ❏ Step 5: Delete or Rename the tnsnames.ora File
- ❏ Step 6: Modify the sqlnet.ora File to Resolve Names in CDS

Note: Upon completion of this task, you can connect to an Oracle database in your DCE environment.

Step 1: Enable CDS for use in Performing Name Lookup

To use CDS for name resolution, the DCE Integration CDS Naming Adapter must be installed on all clients and servers that use CDS. Also, the CDS namespace must have been configured for use by DCE Integration.

> **See Also:** DCE Integration installation instructions, and Task 3: Configure DCE CDS for Use by Oracle DCE Integration on page 13-3.

For example, a service name such as `ORADCE` and its network address can be stored in `DCE CDS`.

Users can typically connect to Oracle services using the familiar Oracle service name if there are no domains or the database is in the user's default domain, as in the following example:

```
sqlplus /@ORADCE
```

This example assumes that DCE externally-authenticated accounts are in use.

As an alternative name resolution service, use a local naming configuration file, `tnsnames.ora`, when CDS is inaccessible. To do so, locate names and addresses of all Oracle servers in the local `tnsnames.ora` file.

Step 2: Modify the CDS Attributes File and Restart the CDS

On all DCE machines where CDS naming will be used, add the object ID (OID) for the CDS attribute TNS_Address to the CDS attributes file. (The object ID must be the same across all machines.)

1. Add a line in the following format to the `/opt/dcelocal/etc/cds_attributes` file:

   ```
   1.3.22.1.5.1    TNS_Address    char
   ```

   The first four digits of this TNS_Address attribute value, `1.3.22.1.x.y`, are fixed, under DCE naming conventions. If the default TNS_Address object ID value `1.3.22.1.5.1` already exists in the `cds_attributes` file, you must specify a value for the object ID that is not already in use.

If you are unable to use the default value for the Object ID, you must specify the object ID in the protocol.ora file on the client.

If you had to specify a value other than the default value 1.3.22.1.5.1, you must add the following parameter to the protocol.ora file:

```
DCE.TNS_ADDRESS_OID=1.3.22.1.x.y
```

Make sure that the object ID value in the cds_attributes file matches the value specified in the DCE.TNS_ADDRESS_OID parameter in the protocol.ora file.

2. Restart CDS on the system.

   The command to restart CDS varies between different operating systems. On the Solaris platform, for example, you can use the following command to restart CDS:

   ```
   /opt/dcelocal/etc/rc.dce restart
   ```

Step 3: Create a tnsnames.ora File for Loading Oracle Connect Descriptors into CDS

To load the Oracle service names and addresses into CDS, create or modify a local naming configuration file, tnsnames.ora. This file is used to map service names to addresses for use by 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, although they are frequently the same.

Parameter			
Name	**Type**	**Mandatory ?**	**Description**
PROTOCOL=DCE	keyword value pair	Yes	Appears in the address sections of (i) `listener.ora`, a listener configuration file, and (ii) `tnsnames.ora`, a local naming configuration file.
SERVER_ PRINCIPAL	DCE Parameter	No	Appears in `tnsnames.ora`
SERVICE	DCE Parameter	Yes	The value given for the DCE parameter (SERVICE=*dce_service_name*) must be the same in `listener.ora` and `tnsnames.ora`
SID	Oracle Parameter	Yes	Identifies the Oracle system ID; each SID value must be unique on a node. This parameter is used locally only, and is not used in DCE CDS.

See Also: *Net8 Administrator's Guide,* for information on `tnsnames.ora`, the local naming configuration file.

Step 4: Load Oracle Connect Descriptors into CDS

A separate utility called `tnnfg` is provided with Oracle DCE Integration to load connect descriptors into CDS. If you configure a new service name and address in `tnsnames.ora`, `tnnfg` adds the new service name and address to CDS. If you change the address for a particular service name, `tnnfg` updates the address for a particular service name.

To load the Oracle service names or aliases from `tnsnames.ora` into CDS, enter the following at the system prompt:

```
% dce_login cell_admin
% tnnfg dceload full_pathname_to_tnsnames.ora
% Enter Password:(password will not display)
```

Be sure to enter the full pathname of the `tnsnames.ora` file, and ensure that the `sqlnet.ora` file exists in the same directory as the `tnsnames.ora` file.

Step 5: Delete or Rename the tnsnames.ora File

You can keep `tnsnames.ora` available as a backup in case CDS becomes unavailable. To assure that CDS is routinely searched instead of `tnsnames.ora`,

configure the NAMES.DIRECTORY_PATH parameter in a profile (sqlnet.ora), as described by Step 6: Modify the sqlnet.ora File to Resolve Names in CDS (the next section).

Step 6: Modify the sqlnet.ora File to Resolve Names in CDS

The parameters required in a profile (sqlnet.ora) depend upon the version of SQL*Net or Net8 you are using.

For a client or server to use DCE CDS Naming, the administrator must do the following:

1. Ensure that the CDS Naming Adapter has been installed on that node.
2. Add the following parameter to the sqlnet.ora file:

   ```
   NAMES.DIRECTORY_PATH=(dce, tnsnames, onames)
   ```

   The first name resolution service listed as a value for this parameter is used. If it is unavailable for any reason, the next name resolution service is used, and so forth.

15

Connecting to an Oracle Database in DCE

This chapter describes 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 contains the following sections:

- Starting the Listener
- Connecting to an Oracle Database Server in the DCE Environment

Starting the Listener

To start the listener, do the following:

1. Enter the following commands:

```
% dce_login principal_name password
% lsnrctl start listener_name
```

For example, if the listener name is LSNR_DCE in the `listener.ora` file, enter the following:

```
% dce_login oracle orapwd
% lsnrctl start LSNR_DCE
```

2. Verify that the server has registered its binding handler with rpcd:

```
% rpccp show mapping
```

Look for the line that includes the `dce_service_name` that is part of the listener address.

3. Verify that the service has been created by searching for the `dce_service_name` as follows:

```
% cdscp show object "/.:/subsys/oracle/service_registry/dce_service_name"
```

For example:

The following command shows you the mapping in the CDS namespace that the listener has chosen for the endpoint:

```
% cdscp show object "/.:/subsys/oracle/service_registry/dce_svc"

            SHOW
          OBJECT    /.../subsys/oracle/service_registry/dce_svc
             AT   1999-05-15-17:10:52
RPC_ClassVersion = 0100
         CDS_CTS = 1999-05-16-00:05:01.221106100/aa-00-04-00-3e-8c
         CDS_UTS = 1999-05-16-00:05:01.443343100/aa-00-04-00-3e-8c
       CDS_Class = RPC_Server
CDS_ClassVersion = 1.0
      CDS_Towers = :
           Tower = ncacn_ip_tcp:144.25.23.57[]
```

Connecting to an Oracle Database Server in the DCE Environment

Connect to an Oracle server in the DCE environment using one of the following methods:

- ❑ Method 1
- ❑ Method 2

Method 1

After externally-identified accounts have been set up, you can take advantage of DCE authentication to log into Oracle without providing any user name / password information. To use this single sign-on capability, just log in to DCE using a command like the following:

```
% dce_login principal_name password
```

For example:

```
% dce_login oracle orapwd
```

> **Note:** You only need to enter the `dce_login` command once. If you are already logged into DCE, you do not need to log in again.

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

```
% sqlplus /@net_service_name
```

where *net_service_name* is the database service name.

For example:

```
% sqlplus /@ORADCE
```

Method 2

From a client, you can still connect with a user name / password:

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

where *net_service_name* is the 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 `tnsnames.ora`, a local naming configuration file, can be used for name lookup when CDS is accessible.

This chapter contains the following sections:

- 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 Oracle8*i* 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.

The following section includes samples of `listener.ora` and `tnsnames.ora` files as they would be configured if a client from outside of DCE wanted to connect to Oracle database servers in a DCE environment.

Sample Parameter Files

At least the following two Oracle parameter files are needed for successful client/server communications; create and modify these files using a text editor:

The parameter files are described in the following sections:

- The listener.ora File
- The tnsnames.ora File

The listener.ora File

The `listener.ora` file resides on the listener node. It defines listener characteristics and the addresses at which the listener listens.

In the following example, each element is displayed on a separate line, to show the file's structure. This is the recommended format, but you do not have to put each element on a separate line. Be sure to include all the appropriate parentheses, and to indent if you must continue an element on the next line.

This example assumes the UNIX operating system and the TCP/IP protocol for one listener, and the DCE protocol for another listener. A single listener can have multiple addresses. For example, instead of having two separate listeners for different database instances on a server node, you could have *one listener for both,*

listening on both TCP/IP and on DCE. However, performance is improved with separate listeners.

```
LSNR_TCP=
      (ADDRESS_LIST=
          (ADDRESS=
                 (PROTOCOL=IPC)
                 (KEY=DB1)
          )
          (ADDRESS=
                 (PROTOCOL=tcp)
                 (HOST=rose)
                 (PORT=1521)
          ))

  SID_LIST_LSNR_TCP=
         (SID_DESC=
                (SID_NAME=ORASID)
                (ORACLE_HOME=/usr/jprod/oracle8i)
         )
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/oracle8i/network/trace
TRACE File_LISTENER=listener.trc
LOG_DIRECTORY_LISTENER=/usr/prod/oracle8i/network/log
LOG_FILE_LISTENER=listener.log
```

The tnsnames.ora File

This file resides on both the client and the server nodes. It lists the service names and addresses of all services on the network.

The following sample `tnsnames.ora` file maps the service name ORATCP to the connect descriptor that includes a TCP/IP address and the service name ORADCE to a connect descriptor that includes a DCE address.

```
ORATCP = (DESCRIPTION=
          (ADDRESS=
            (PROTOCOL=TCP)
            (HOST=rose)
            (PORT=1521)
           )
           (CONNECT_DATA=
             (SID=DB1)
           )
          )
ORADCE=(DESCRIPTION=
         (ADDRESS=
           (PROTOCOL=DCE)
           (SERVER_PRINCIPAL=oracle)
           (CELL_NAME=cell1)
           (SERVICE=dce_svc)
          )
          (CONNECT_DATA=
                (SID=ORASID)
          )
          )
```

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 of the `tnsnames.ora` file (in the context of native naming adapters) is to load Oracle service names and network addresses into CDS, it could be used temporarily as a backup name resolution service if CDS is inaccessible.

SQL*Net Release 2.2 and Earlier

To use the `tnsnames.ora` file for name lookup and resolution, remove (or comment out) the "native name" parameters from the `sqlnet.ora` file on the client. To comment out the lines, add a pound sign (#) at the beginning of each line.

For example:

```
#native_names.use_native=true
#native_names.directory_path=(dce)
```

SQL*Net Release 2.3 and 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 V

Oracle8*i* Enterprise User Security

This part describes Oracle8*i* directory and security integration functionality, which enables single sign-on in a client/server environment.

This part contains the following chapters, which describe how to set up enterprise user security in an Oracle database environment:

- Chapter 17, Managing Enterprise User Security
- Chapter 18, Using Oracle Wallet Manager
- Chapter 19, Using Oracle Enterprise Login Assistant
- Chapter 20, Using Oracle Enterprise Security Manager

17

Managing Enterprise User Security

Enterprise user security management lets you create and administer large numbers of users in a secure, LDAP-compliant directory service. This chapter describes the configuration and setup of enterprise user security management, in the following sections:

Part I: Overview / Concepts:

- ❏ Overview of Enterprise User Security
- ❏ Shared Schemas
- ❏ Current User Database Links
- ❏ Oracle Enterprise User Security Components

Part II: Procedure

- ❏ Installing and Configuring Enterprise User Security
- ❏ Troubleshooting Enterprise User Login

See Also: Chapter 20, Using Oracle Enterprise Security Manager

Part I: Overview / Concepts

- **Overview of Enterprise User Security**
- **Shared Schemas**
- **Current User Database Links**
- **Oracle Enterprise User Security Components**

Overview of Enterprise User Security

This section describes fundamental concepts related to enterprise user security management:

- Introduction to Enterprise User Security
- About Directories
- Elements of Enterprise User Security Management
- The Enterprise User Security Process

Introduction to Enterprise User Security

Administrators 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, individual users can have multiple accounts on multiple databases. This can produce 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 be changed the same day in order to guard against misuse of that user's accounts and privileges. However, in a large enterprise, with user accounts and passwords distributed over multiple databases, an administrator may not be able to make the timely changes required by good security practices.

Enterprise user security management addresses these user, administrative, and security challenges by centralizing storage and management of user-related information in an LDAP-compliant directory service. When an employee changes jobs in such an environment, the administrator need only modify information in one location—the directory—to make effective changes in multiple databases and systems. This centralization lowers administrative costs and improves enterprise security.

Enterprise user security provides single sign-on to Oracle8*i* using interoperable X.509 v3 certificates over Secure Sockets Layer (SSL) v3, and supports the following LDAP-compliant directory services:

- Oracle Internet Directory Release 2.0.5 or later
- Microsoft Active Directory

It also provides a tool, Oracle Enterprise Security Manager, to create user entries in the directory and manage authorizations for those users.

See Also: Chapter 20, Using Oracle Enterprise Security Manager

About Directories

A directory is an index to information, organized 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. Examples of directories include a telephone book, a library card catalog, and a department store catalog.

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 include:

- Entries
- Distinguished Names and Directory Information Trees
- Naming Contexts
- Authorization and Access Control

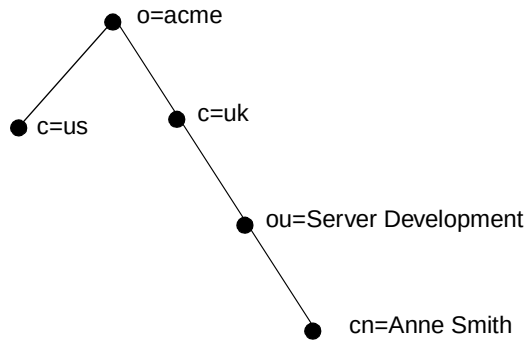
Entries

In a directory, each collection of information about an object is called an **entry**. Just as a 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 where the entry resides in the directory's hierarchy, called a **directory information tree (DIT)**, illustrated by 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 organizational unit (ou) Server Development, in the country (c) of Great Britain (uk), in the organization (o) Acme.

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 starts with the least significant component (that naming the entry itself) and proceeds to the most significant component (that just below the root).

Note: The example in Figure 17–1 uses the following notation to define distinguished name components:

- o = organization
- c = country
- ou = organizational unit
- cn = common name

Naming Contexts

A directory's information is divided into units called **directory naming context**s. A directory naming context is a subtree that resides entirely on one server. It must be contiguous, beginning at an entry that serves as the top of the subtree, and

extending 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 8.1.7, as described later in this chapter, you choose one or more naming contexts to contain Oracle enterprise information. These are called **administrative context**s, and within each one, you can create a container to hold Oracle enterprise information—called an **Oracle Context**.

Authorization and Access Control

The **authorization** process ensures that a user has access to only that 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 an **Access Control List (ACL)**. An 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

Oracle Internet Directory

Combining the flexibility of the Internet's LDAP standard with the robustness of the Oracle8*i* platform, the Oracle Internet Directory provides a scalable, reliable and secure LDAPv3 directory service for mission critical applications.

Oracle Internet Directory is an LDAP Version 3 service that combines the mission critical strength of Oracle's database technology with the flexibility and compatibility of the LDAP directory standard. Oracle Internet Directory is tightly integrated with the Oracle management environment, making it the enterprise directory of choice for Oracle using organizations. Its scalability, high availability and security features make it the ideal customer choice for internet service provider (ISP) and telecommunications carrier implementations.

The Oracle Internet Directory server is implemented as an application running on an Oracle8*i* database. Through its tight integration, Oracle Internet Directory

effectively leverages the features of the Oracle platform to make it the compelling choice for mission-critical applications.

Oracle Internet Directory provides comprehensive and flexible support for LDAP v3 directory access control. This includes entry level, attribute level, and prescriptive access controls to provide varying levels of security to custom fit enterprise and service provider needs. An administrator can grant or control access to a specific directory object or an entire directory subtree. Oracle Internet Directory implements three levels of user authentication: anonymous, password-based, and certificate-based, using the Secure Socket Layer (SSL) v3 protocol for authenticated access and data privacy.

Oracle Internet Directory and Oracle Enterprise Manager both include Oracle Directory Manager, a graphical directory administrative tool for managing and administering directory information from anywhere in the distributed environment. It also manages directory schema, replication agreements, and access control information. Oracle Directory Manager provides *administrative transparency* for Oracle using organizations deploying both Enterprise Manager and Oracle Internet Directory. Written entirely in Java, Oracle Directory Manager is portable to all Oracle platforms.

See Also: *Oracle Internet Directory Administrator's Guide*

Elements of Enterprise User Security Management

The principal directory entries that relate to enterprise user security management include the following:

- Enterprise Users
- Enterprise Roles and Global Roles
- Local Roles and Privileges
- Enterprise Domains
- Oracle Context
- Administrative Context
- Administrative Groups
- Server-Related Objects

Enterprise Users

An **enterprise user** is one that is defined and managed in a directory. Each **enterprise user** has a unique identity across an enterprise.

Enterprise Roles and Global 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** includes privileges contained in a database, but the global role is managed in a directory. An enterprise role is thus a container of **global role**s. 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 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.

An enterprise domain subtree includes an enterprise role object that 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.

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.

Local Roles and Privileges

Local roles are stored in the database, and can be used in combination with enterprise roles. Local roles and privileges can also be created in the database. They can be granted directly to a shared schema, or they can be granted to a global role—that is part of an enterprise role and granted to a user.

See Also:

- Shared Schema Functionality And SSL on page 17-18.
- Note on page 17-20.

Enterprise Domains

An **enterprise domain** is a group of databases and **enterprise role**s. 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. An enterprise domain subtree in a directory is composed of two objects: enterprise role objects (discussed by Enterprise Roles and Global Roles on page 17-8), and 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 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 17-21.

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 and a product subtree, and can also include server and Net8 objects.

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. The Oracle Context subtree is created by Oracle Net8 Configuration Assistant (Net8CA), and is populated with server entries by Oracle Database Configuration Assistant (DBCA) during the database install, and by Oracle Enterprise Security Manager during administration. Databases (and other Oracle LDAP clients) refer to entries in the context to determine enterprise user authorization at login.

During database installation, a default enterprise domain (`cn=OracleDefaultDomain`) is established. The domain administrator can later add

additional enterprise domains (represented in Figure 17–2 as `cn=Domain1`) by using Oracle Enterprise Security Manager.

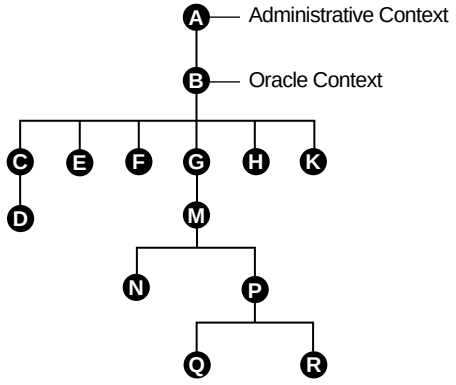
Note: Do not remove the default enterprise domain (cn=OracleDefaultDomain); it is required when using the Oracle Database Configuration Assistant to register a database.

See Also: Creating an Enterprise Domain on page 20-15.

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 Oracle Net8 Configuration Assistant during or after installation, you select an administrative context. An administrative context, an Oracle Context, and its subtrees are illustrated by Figure 17–2:

Figure 17–2 An Administrative Context



A o=acme
B cn=OracleContext
C cn=server1
D cn=mapping2
E Net8 objects
F OracleDBSecurityAdmins
G cn=products
H cn=OracleNetAdmins
K cn=OracleDBCreators
M cn=OracleDBSecurity
N cn=OracleDefaultDomain
P cn=Domain1
Q cn=mapping1
R cn=EntRole1

Note: Do not modify the ACLs for the objects contained in an Oracle Context; doing so breaks the security configuration for these objects—and may break enterprise user functionality as well.

Administrative Groups

The Oracle Context contains three administrative groups, each with its associated ACL. The user who creates the Oracle Context with Net8CA automatically becomes the first member of each of these groups. The three administrative groups in an Oracle Context are described in Table 17–1:

Table 17–1 Administrative Groups in an Oracle Context

Administrative Group	Description
OracleNetAdmins	Members of the OracleNetAdmins group (`cn=OracleNetAdmins,cn=OracleContext`) have `create`, `modify`, and `read` access to Net8 objects and attributes. Net8CA 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. Net8CA 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.

Table 17–1 Administrative Groups in an Oracle Context

Administrative Group	Description
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 Net8CA 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.

You can also have a Domain Administrator responsible for managing a single domain. This administrator is less privileged than the **Database Security Administrator**. Similarly, you can have a **Database Administrator** responsible for a single database directory entry.

> **See Also:** *Net8 Administrator's Guide* for information about adding members to the OracleNetAdmins group

Server-Related Objects

In addition to server-related objects, an Oracle Context may also include other types of objects (Table 17–2):

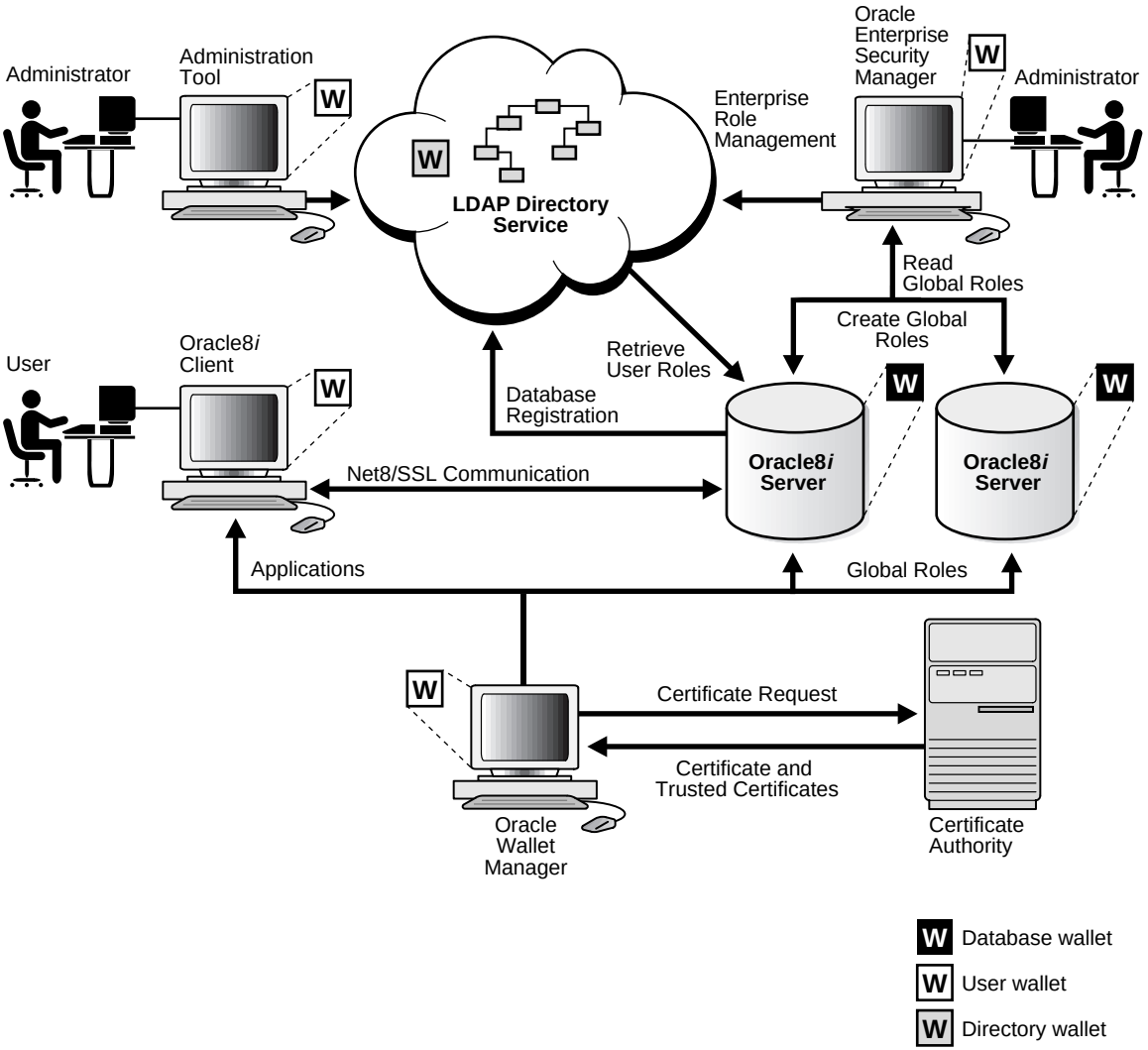
Table 17–2 Server-Related Objects in an Oracle Context

Object	Description
Database server object	A database server object (represented as `cn=server1` in Figure 17–2) contains information about a database server. It is created by the Oracle Database Configuration Assistant during the database installation (or later, using the DBCA Modify Database command) 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. They can also be created later by members of the OracleNetAdmins group.

Overview of Enterprise User Security Management

Figure 17–3 provides an overview of the Enterprise User Security Management process:

Figure 17–3 Overview of Enterprise User Security

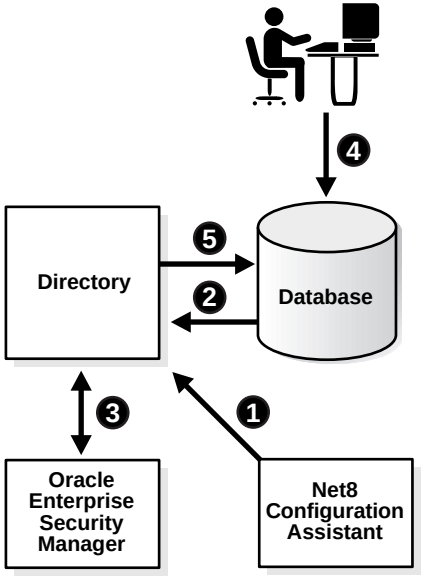


The Enterprise User Security Process

Figure 17–4 describes the operation of the Enterprise User Security Management process, assuming:

- A wallet has been set up and configured for the user.
- The user is authenticated to the database through SSL.

Figure 17–4 How Enterprise User Security Management Works



1. An administrator uses Net8CA to (i) select the administrative context in the directory, and (ii) and create an Oracle Context.
2. A member of the OracleDBCreators group uses the Oracle Database Configuration Assistant to register the database with the directory.
3. An administrator uses Oracle Enterprise Security Manager to set up both enterprise users and enterprise roles in the directory and relevant domains.
4. A user initiates an SSL connection to the database (logs on), and the database uses SSL to authenticate the user.

5. The database retrieves the user's enterprise roles from the directory, and authorizes any associated global roles applicable to that database.

Shared Schemas

The following sections describe shared schema features, and how to set them up:

- Overview
- Setting Up Shared Schemas
- Shared Schema Functionality And SSL
- Creating a Shared Schema
- Creating an Enterprise User in the Directory
- Mapping an Enterprise User to a Shared Schema

Overview

Users do not necessarily require individual accounts or schemas set up in a database they wish to access. Alternatively, they can be granted access to common, **shared schema**s (also called **user/schema separation**) associated with target applications. For example, suppose that users *Tom*, *Dick*, and *Harriet* require access to the Payroll application on the Finance database. They do not need to create unique objects in the database, and therefore do not need their own schemas—they do need access to the Payroll schema.

Oracle8*i* Release 8.1.7 supports mapping multiple users stored in an enterprise directory to shared schema on an individual database. This separation of users from schemas reduces administration costs by reducing the number of user accounts. It means that you do not need to create an account for each user—a user schema—in multiple databases, in addition to creating the user in the directory. Instead, you can create a user in one location, the enterprise directory, and map the user to a shared schema that other enterprise users can also be mapped to. For example, if Tom, Dick and Harriet all access both the Sales and the Finance databases, you do not need to create an account for each user on each of these databases. Instead, you can create a single shared schema on each database, such as `SALES_APPLICATION` and `FINANCE_APPLICATION`, respectively, that all three users can access. A typical environment might have some 5,000 enterprise users mapped to just one of three or four shared schemas.

Summary:

- Shared schemas eliminate the need to have a dedicated database schema on each database for each enterprise user.

- Each enterprise user can be mapped to a shared schema on each database the user needs to access. The user is granted access to each such shared schema when the user connects to a database.
- Shared schemas lower the cost of managing users in an enterprise.

Setting Up Shared SchemaS

To configure shared schemas, the local Database Administrator must create at least one database schema in a database. Enterprise users can be mapped to this schema.

In the following example, the administrator creates a shared schema and maps users to it:

- The administrator creates a global shared schema called EMPLOYEE and the global role HRMANAGER on the HR database.
- The administrator uses Oracle Enterprise Security Manager to create and manage enterprise users and roles in the directory. For example, the administrator creates enterprise user Harriet and an enterprise role named MANAGER. The administrator then assigns the global role HRMANAGER to the enterprise role MANAGER.
- The administrator assigns enterprise roles to enterprise users in the directory. For example, the administrator assigns the enterprise role MANAGER to Harriet.
- The administrator uses Oracle Enterprise Security Manager to map the user Harriet in the directory to the shared schema *EMPLOYEE* on the HR database.

When Harriet connects to the database, she is automatically connected to the EMPLOYEE schema and is given the global role HRMANAGER. Multiple enterprise users can be mapped to the same shared schema. For example, the enterprise security administrator can create another enterprise user *Scott* and map Scott to the EMPLOYEE schema. From that point on, both Harriet and Scott automatically use the EMPLOYEE schema when connecting to the HR database, but each can have different roles—and can be individually audited.

Shared Schema Functionality And SSL

Shared schema functionality relies on SSL for authentication to the database. SSL authentication occurs as follows:

- Prior to connecting to a database, an enterprise user opens a wallet by providing a password to Oracle Wallet Manager or Oracle Enterprise Login Assistant.

- When connecting, Oracle Advanced Security performs an SSL handshake with the database, during which it passes the user's unique certificate to the server; this handshake authenticates the user to the server.
- The database extracts the user's DN from the user's certificate and looks it up 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 with 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 lets multiple enterprise users share part of their DN to access the same shared schema. This method is useful if multiple enterprise users are already grouped under some common root in the directory tree. The subtree that these users share can be mapped to a shared schema on a database. For example, you can map all enterprise users in the subtree for the engineering division to one shared schema, `BUG_APPLICATION`, on the bug database.
  - No mapping at all
- If the database does *not* find either the DN locally or an appropriate DN mapping object in the directory, it refuses the user's connection to the database.

  If the database *does* find either the DN locally or the appropriate DN mapping object in the directory, the database allows the user to log on.
- The database maps the user to the associated schema.

  For example, suppose that Harriet is trying to connect to the HR database, but the database does not find Harriet's DN. In this case, the HR database looks up the Harriet's DN in the directory. The directory has a mapping of Harriet to the shared schema `EMPLOYEE` and returns this schema. The database logs Harriet in and maps her to the `EMPLOYEE` schema.

- The database retrieves this user's global roles for this database from the directory.

  The database also retrieves from its own tables any local roles and privileges associated with the database schema to which the user is mapped.

  The database uses both the global and the local roles to determine the information that the user can access.

  Note: You can configure the database so that it uses *local roles only* and does not look up global roles in the directory. To do this, do not register the database in the directory with the Oracle Database Configuration Assistant. If this configuration is set, the database uses only local roles to determine what the user can access. This lets customers use SSL for client authentication, without having to manage user privileges centrally. *This configuration does not work with mapped users and shared schemas—because the mapping information must be shared in the directory.*

  See Also:

  - Chapter 9, Configuring Secure Socket Layer Authentication, for information about SSL
  - Chapter 18, Using Oracle Wallet Manager, for information about wallets and Oracle tools for managing them

Continuing this example, assume that the enterprise role MANAGER contains the global roles ANALYST on the HR database, and CLERK on the Payroll database. When Harriet, who has the enterprise role MANAGER, connects to the HR database, she uses the schema *EMPLOYEE* on that database. Her privileges on that database are determined by:

- The global role ANALYST
- Any local roles and privileges associated with the *EMPLOYEE* schema on the HR database

When Harriet connects to the Payroll database, her privileges are determined by:

- The global role 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 [shared schema name] IDENTIFIED GLOBALLY AS ''
```

For example, the administrator for the HR database creates a shared schema for the user SALES_APPLICATION as follows:

```
CREATE USER sales_application IDENTIFIED GLOBALLY AS ''
```

Note: *There is no space* between the single quotation marks in the syntax for creating a shared schema.

Creating an Enterprise User in the Directory

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 bulk load tool.

See Also:

- Chapter 20, Managing Enterprise User Security.
- Your LDAP directory documentation (such as for Oracle Internet Directory or Microsoft Active Directory)

Mapping an Enterprise User to a Shared Schema

The mapping between enterprise users and a schema can be done in either the database or the directory.

The mapping is done in the directory by means of one or more mapping objects. A mapping object is used to map the Distinguished Name (DN) of a user, 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 of the following:

- A Full DN (entry-level) mapping
- Partial DN (subtree-level) mapping

When determining the schema to which it should connect the user, the database uses the following precedence rules:

- It first looks for that schema locally.

- If it does not find it locally, it searches the directory. Within the directory, it looks under the *server* object, first for a full DN mapping, then for a partial DN mapping.
- If it does not find a mapping object under the server object, it looks under the *domain* object, first for a full DN mapping, then for a partial DN mapping.
- If it does not find a mapping object in the domain object, the database refuses the connection.

You can grant privileges to a specified group of users by granting roles and privileges to a database schema. Every user sharing such a schema gets these local roles and privileges in addition to personal enterprise roles. However, you should exercise caution when doing this, because every user who is mapped to this shared schema can exercise the privileges assigned to it.

Current User Database Links

Oracle8*i* supports current user database links, which let you make a procedural connection to a second database as another user and with that user's privileges—though it does not require that the second user's credentials be stored in the database link definition. Such access is limited to the scope of the database link procedure.

For example, a current user database link lets Harriet, a user of the Accounts Payable database, procedurally access the Human Resources database by connecting as *Scott*, and using Scott's credentials.

For Harriet to access a current user database link to connect to the schema Scott, Scott must be a schema created as `IDENTIFIED GLOBALLY` in both databases. Harriet, however, can be a user identified in one of three ways:

- By a password
- `GLOBALLY`
- `EXTERNALLY`

To create Scott as a global user in both the Accounts Payable and Human Resources databases, you must enter 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 described in Creating a Shared Schema on page 17-21. In this case, the schema is Scott's alone. In order for the current user database link to work, the schema created for Scott cannot be shared with other users.

Current user database links operate only between 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. To specify a database as untrusted that is part of a trusted enterprise domain, use the PL/SQL package `DBMS_DISTRIBUTED_TRUST_ADMIN`. To obtain a list of trusted servers, use the `TRUSTED_SERVERS` view.

See Also:

- *Oracle8i Distributed Database Systems*, for additional information about current user database links
- *Oracle8i SQL Reference*, for more information about syntax
- *Oracle8i Supplied PL/SQL Packages Reference*, for information about 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 functionality uses the following administration tools:

- Oracle Wallet Manager
- Oracle Enterprise Login Assistant
- Oracle Enterprise Security Manager

Oracle Wallet Manager

Oracle Wallet Manager is a standalone Java application that wallet owners and security administrators use to manage and edit the security credentials in their Oracle wallets. Wallet Manager tasks include:

- Generating a public/private key pair
- Creating a certificate request for submission to a **certificate authority** (CA)
- Installing a certificate for the entity
- Configuring trusted certificates for the entity
- Creating a wallet that can be opened later using the Oracle Enterprise Login Assistant
- Opening a wallet to enable access to PKI-based services

> **See Also:** Chapter 18, Using Oracle Wallet Manager, for detailed information about using 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. A functional subset of Oracle Wallet manager, this easy-to-use tool lets users connect to multiple services with a single sign-on. Oracle Enterprise Login Assistant masks the complexity of SSL, wallets, enterprise users, and the process of authenticating to multiple databases. It lets users securely access multiple databases and applications using a single password, entered only once per session.

> **See Also:** Chapter 19, Using Oracle Enterprise Login Assistant, for additional information about this tool

Oracle Enterprise Security Manager

Oracle Enterprise Security Manager is an administration tool that provides a graphical user interface to help you manage enterprise users, enterprise domains, databases, and enterprise roles that are stored in a directory server. Use Oracle Enterprise Security Manager to:

- Administer databases
- Administer enterprise domains
- Administer enterprise users
- Administer database, security, and domain administrators
- Manage the administration groups in the Oracle Context
- Create new user-schema mappings

See Also: Chapter 20, Using Oracle Enterprise Security Manager, for detailed instructions about using this application

Part II: Procedure

- **Installing and Configuring Enterprise User Security**
- **Troubleshooting Enterprise User Login**

Installing and Configuring Enterprise User Security

This section describes how to set up enterprise user security. The required tasks follow:

- ❏ Task 1: Install or Identify a Certificate Service
- ❏ Task 2: Install and Configure a Directory Service
- ❏ Task 3: Install and Configure the Database
- ❏ Task 4: Configure the Database for SSL
- ❏ Task 5: Create and Configure the Wallet
- ❏ Task 6: Create Global Schemas and Roles
- ❏ Task 7: Configure Database Clients
- ❏ Task 8: Configure an Enterprise Domain
- ❏ Task 9: Configure Enterprise Users
- ❏ Task 10: Log In as the Enterprise User

Task 1: Install or Identify a Certificate Service

Oracle Wallet Manager requires you to have a **certificate authority** (CA) in your environment. You can use a CA vendor's certificates, or you can use your own CA that can process PKCS#10 certificate requests in Base 64 format and return X509v3 certificates—also in Base 64 format.

> **See Also:** Chapter 18, Using Oracle Wallet Manager, for a description of certificate authorities and Oracle Wallet Manager

Task 2: Install and Configure a Directory Service

Conceptually, there are four major prerequisites for an Oracle RDBMS to communicate with the directory:

- The Oracle Schema must be installed in the directory, if it is not there already. This includes all the Oracle-specific objectclasses and attributes, but no specific objects. Oracle Internet Directory comes with the Oracle Schema pre-installed.
- An Oracle Context must be created in the directory, if one does not already exist. This is the subtree where all the Oracle-specific objects are stored. Oracle Internet Directory does NOT currently come with a pre-installed Oracle Context.

- There must be a local file (`ldap.ora`) that tells the RDBMS how to connect to the directory, including host, port, and directory type— and the location in the directory of the Oracle Context.
- There must be an entry in the directory representing the database, so it can bind (log in) to the directory.

Net8CA performs the first three steps (called *directory service access configuration),* and the Oracle Database Configuration Assistant (DBCA) performs the fourth. Note that you must create the Oracle Context once per directory, but you need to create an `ldap.ora` file for each `ORACLE_HOME` used to access the directory. If there is no context, Net8CA asks if you want to create one; if one already exists, it asks which one you want to use. If there is no Oracle schema installed in the directory (which is done automatically for Oracle Internet Directory), Net8CA prompts you to install it.

If you run the recommended *custom* database install, Net8CA and DBCA automatically complete the directory-related configuration.

If you install the Oracle8*i* database using a *typical* install, Net8CA and DBCA do not complete the directory-related configuration. In this case, you must run both Net8CA and DBCA in standalone mode.

To install and configure a directory service:

- ❏ Step 1: Install an LDAP v3-Compliant Directory Service
- ❏ Step 2: Set Up the Directory for Two-way SSL Authentication
- ❏ Step 3: Start the Directory
- ❏ Step 4: Create Directory Naming Contexts
- ❏ Step 5: Create Administrative Contexts
- ❏ Step 6: Create Administrative Users
- ❏ Step 7: Create Oracle Contexts

Step 1: Install an LDAP v3-Compliant Directory Service

Oracle8*i* Release 8.1.7 supports the following:

- Oracle Internet Directory Release 2.0.5 or later
- Microsoft Active Directory

Step 2: Set Up the Directory for Two-way SSL Authentication

This requires the creation of 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, see the product-specific documentation for instructions about enabling two-way SSL authentication.

Step 3: Start the Directory

Step 4: Create Directory Naming Contexts

Step 5: Create Administrative Contexts

Create administrative contexts beneath the directory naming contexts in the DIT.

For example:

If your naming context is `dc=my_company,dc=com`, you can enter `dc=us,dc=my_company,dc=com` as an administrative context.

Step 6: Create Administrative Users

Create administrative users—especially those authorized to register databases.

Step 7: Create Oracle Contexts

If you are using Oracle Internet Directory, use Net8CA to create Oracle Contexts (the Oracle schema is installed automatically).

If you are not using Oracle Internet Directory, you can use Net8CA later to install the Oracle schema and create Oracle Contexts.

See Also:

- Administrative Context on page 17-10
- The chapter about configuring naming methods in *Net8 Administrator's Guide* for instructions about installing the Oracle directory schema into the directory and creating an Oracle Context
- *Oracle Internet Directory Administrator's Guide* for instructions about using Oracle Wallet Manager and Oracle Directory Manager to create a wallet for the directory

Task 3: Install and Configure the Database

To install and configure one or more databases:

- ❏ Step 1: Install Oracle8i Release 8.1.7 Database Software
- ❏ Step 2: Set Up Directory Access for ORACLE_HOME
- ❏ Step 3: Authorize Users for Administrative Functions
- ❏ Step 4: Use Oracle Database Configuration Assistant to Register the Database in the Directory

Step 1: Install Oracle8*i* Release 8.1.7 Database Software

Use the Oracle Universal Installer to install Oracle8*i* databases. Before installation, obtain the following from the directory administrator:

- A directory login name and password. The login name must belong to someone in the OracleDBCreators group (See: Note on page 17-34). 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 8.1.7 installation documentation for your platform, for detailed instructions about how to install and create a database.

Net8CA runs automatically at the end of the Oracle8*i* installation process; see Step 2 for related instructions.

Step 2: Set Up Directory Access for ORACLE_HOME

To configure directory service access configuration, use Net8CA:

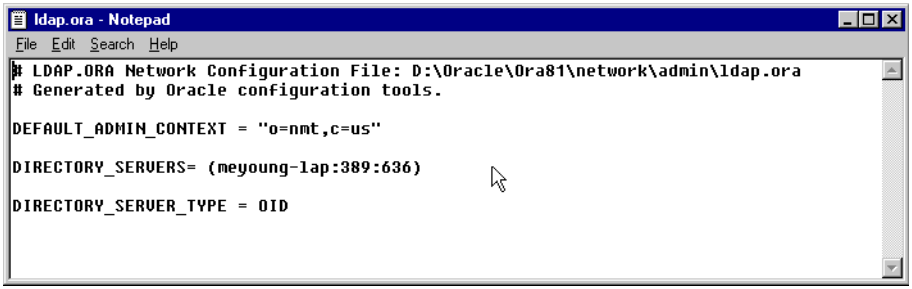
1. Run Net8CA:
   - Microsoft NT: Select `Start–>Programs–>Oracle - OraHome81–>Network Administration–>Net8 Configuration Assistant`

- UNIX: Type `net8ca` at the command line.

2. Select `Directory Service Access Configuration`; choose `Next`.
3. Select `Perform directory access configuration for a server`; choose `Next`.
4. In the Directory Type window, select your Directory Type (Oracle Internet Directory, for example).
5. In the Directory Location window:
   - Enter the name of your directory host system in the Hostname field.
   - Enter the non-SSL port number in the Port field; the default is `389`.
   - Enter the SSL port number in the SSL Port field; the default is `636`.
   - Choose `Next`.
6. If an Oracle Context already exists in the directory, select it—and proceed to Step 11.
7. In the next window, Net8 Configuration Assistant: Directory Access, No Oracle Context: choose `Yes, I want to create a new Oracle Context;` choose `Next`.
8. The next window, Net8 Configuration Assistant: Directory Access, Create Oracle Context, asks where to create your context, and presents a list of the known naming contexts in the directory.

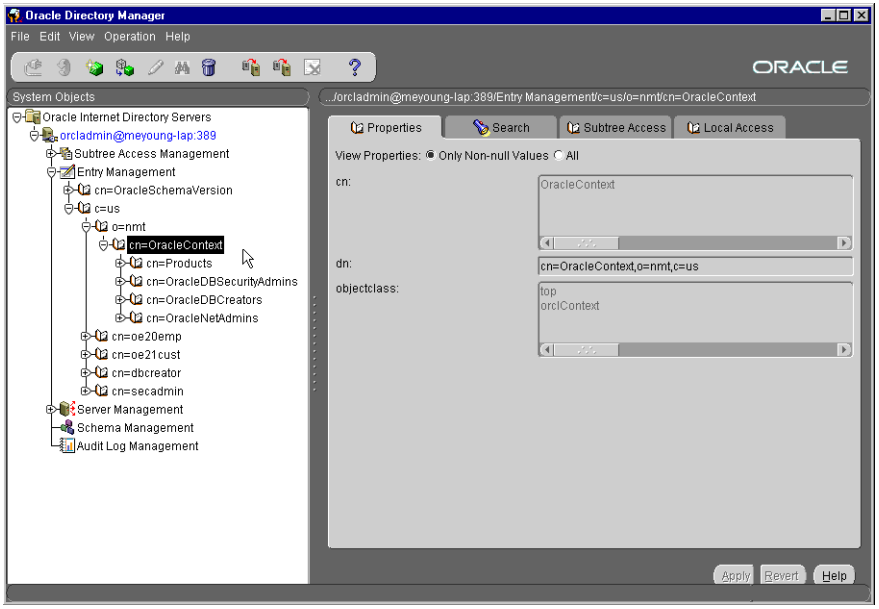
   Either select an existing naming context, or enter the **distinguished name (DN)** of a directory entry under which you want to create your new context; choose `Next`.
9. In the next window, Net8 Configuration Assistant: Directory Access, Credentials, enter the directory credentials (User DN, Password) of a user authorized to create objects under the chosen administrative context in the directory; choose `Next`.
10. These steps create a complete Oracle Context in the directory, and a local `ldap.ora` file that tells the database where to find the context and the directory, and which port to use. Choose `Finish` to exit Net8CA.
11. Verify that an `ldap.ora` file is located in one of the following directory paths:
    - Microsoft NT: `X:\Oracle\Ora81\network\admin` (Figure 17–5)
    - UNIX: `$ORACLE_HOME/network/admin`

Figure 17–5 Example: The ldap.ora File (Microsoft NT)



12. Use Oracle Directory Manager to verify that there is a new Oracle context subtree in your directory. In the example (Figure 17–6), it is rooted at `cn=OracleContext,o=nmt,c=us`.

Figure 17–6 Example: Oracle Context Subtree



See Also: *Net8 Administrator's Guide*, for instructions about setting up directory access for the database (*Configuring Naming Methods*).

Step 3: Authorize Users for Administrative Functions

To register your database in the directory (using DBCA), you must provide directory credentials for either (i) a user in the database Installation Administrator's group, or (ii) the directory superuser. If you choose the first approach, you may need to add appropriate users to that group *before* running DBCA. Use Oracle Enterprise Security Manager to put the appropriate directory users into the *Database Installation Admin group*—so they can register the database in the directory.

Note: This group is called the Database Installation Admin group by Oracle Enterprise Security Manager, but the actual group name in the directory is `ORACLEDBCREATORS`.

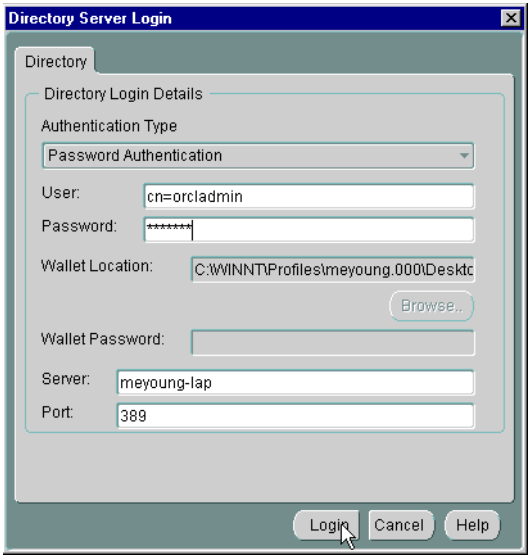
See Also: Chapter 20, **Managing Enterprise User Security**

To authorize users for administrative functions:

1. Start Oracle Enterprise Security Manager:
   - Microsoft NT: Select `Start—>Programs—>Oracle - OraHome81—>Extended Administration—>Enterprise Security Manager`.
   - UNIX: Type `oemapp esm` at the command line.
2. Log out:
   - Microsoft NT: Select Directory—>Logout
   - UNIX: Select Directory—>Logout

   This step is necessary because Oracle Enterprise Security Manager logs you into the directory anonymously by default.
3. Log back in. Select `Directory—>Login` and log in as a directory administrator, or as a user with permission to modify the context. If it is a new context, this user is the one who created the context. If it is a pre-existing context, any user in the database security admin group is acceptable; the Directory Server Login window appears (Figure 17–7); choose `Login`.

Figure 17–7 Example: The OID Directory Server Login Window



4. Use the User Search Base window to define the entry the directory is to search under (See: Chapter 20, Managing Enterprise User Security for details).
5. Using Oracle Enterprise Security Manager, add appropriate users to the following groups:
   - Database Installation Administrators
   - Database Security Administrators

Step 4: Use Oracle Database Configuration Assistant to Register the Database in the Directory

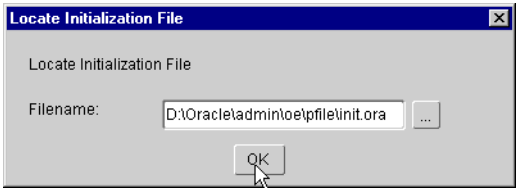
1. If you performed a *typical* database installation, start DBCA in standalone mode as follows:
   - Microsoft NT: Select `Start–>Programs–>Oracle - OraHome81–>Database Administration–>Database Configuration Assistant`
   - UNIX: Enter `dbassist` at the command line.
2. If you performed a *custom* database installation, Oracle Database Configuration Assistant asks if you want to register the database in the directory. Choose `Yes`.

3. If you are running DBCA in a standalone mode, select `Change database configuration` and choose `Next`.
   - Select a database and choose `Next` (this process typically takes about a minute to complete).
   - Accept the correct server mode and choose `Next`.
   - Select any option and choose `Next`; the final Oracle Database Configuration Assistant window appears (Figure 17–8):

Figure 17–8 Oracle Database Configuration Assistant Window (Finish)

   - Choose `Yes, Register the Database`, and enter the directory credentials for a user in the Database Installation Administrators group.
   - Choose `Finish`; the Locate Initialization File window appears (Figure 17–9):

Figure 17–9 DBCA Locate Initialization File Window



- Select the appropriate initialization file and choose OK. Oracle Database Configuration Assistant creates a new database service object underneath your chosen Oracle Context in the directory. The credentialed user is the one with the correct access permissions to create the new database entry in the directory. This user obtained these permissions by being placed into the Database Installation Administrators group by Oracle Enterprise Security Manager. When DBCA registers the database in the directory, it also adds the `RDBMS_SERVER_DN` initialization parameter to the `init.ora` file for the database. This parameter represents the **distinguished name (DN)** of the database as registered in the directory.

Notes:

- Do not modify the `RDBMS_SERVER_DN` parameter manually by editing the `init.ora` file; this can make the locally stored DN for the database inconsistent with the actual DN in the directory. If this happens, change the `init.ora` parameter to match the DN for the database in the directory.
- Use your directory administration tool to verify that a database entry exists under your Oracle context in the directory.
- If you get an error message, use Oracle Enterprise Security Manager to check that the user you are providing credentials for is in the Database Installation Administrators group.

Task 4: Configure the Database for SSL

To configure the database for SSL:

- ❏ Step 1: Configure Net8 for Listener and Database SSL Support
- ❏ Step 2: Configure SSL Service Name
- ❏ Step 3: Configure the Listener

- ❑ Step 4: Review the .ORA Files

Step 1: Configure Net8 for Listener and Database SSL Support

If you are using an LDAP directory service for enterprise security, you can use Net8 directory naming. This lets the client connect to the database using the database entry registered with the directory by Oracle Database Configuration Assistant. You can alternatively use one of the other Net8 naming methods, such as local naming (`tnsnames.ora` file), to configure a net service name for the database. See the *Oracle Net8 Administrator's Guide* for more information.

Net8 must be configured for SSL on both the **listener** and the database. The listener must have a listening endpoint that is configured for the TCP/IP with SSL protocol, and the location of the database wallet must be specified. Use Net8 Assistant to do this (See: Enabling SSL on page 9-10):

1. Run Net8 Assistant:
   - Microsoft NT: Select `Start–>Programs–>Oracle - OraHome81–>Network Administration–>Net8 Assistant.`
   - UNIX: Enter `netasst` at the command line.
2. Configure Profile:
   - Select `Profile`.
   - Select `Oracle Advanced Security` from the drop-down list at the top of the right window region.
   - Choose the `SSL` tab.
   - Choose the `Server` button.
   - Add the database name to the end of the wallet location.
   - Select `File–>Save the network configuration`; your `sqlnet.ora` file is updated.

Suggested Wallet Locations for a database:

- Microsoft NT: `C:\WINNT\Profiles\DATABASES\database_name`
- UNIX: `/etc/ORACLE/WALLETS/DATABASES/database_name`

Step 2: Configure SSL Service Name

To configure the SSL service name:

1. In the Net8 Assistant Service Naming window, select the Service Naming icon and choose the Create icon (the big green plus sign at the upper-left side of the window).
2. In the Net8 Service Name Wizard window (*Welcome*), enter your chosen Net Service Name; this is the name you use to access your database as an enterprise user. Choose `Next`.
3. In the Net8 Net Service Name Wizard window (*page 2 of 5: Protocol*), select `TCP/IP with SSL`; choose `Next`.
4. In the Net8 Net Service Name Wizard window (*page 3 of 5: Protocol Settings*), enter your database Host Name and Port Number—that you will use for the SSL connection.
5. In the Net8 Net Service Name Wizard window (page 4 of 5: Service), enter the Oracle 8i Service Name; choose `Next`.
6. Do not test this connection when asked. It will fail because (i) you have not set up the listener to listen for SSL connections, and (ii) you have not set up the database wallet; choose `Finish`.
7. In the Net8 Assistant window, select `File–>Save the network configuration`; your `TNSNAMES.ORA` file is updated, and can be reviewed later.

Step 3: Configure the Listener

To configure the **listener**:

1. In the Net8 Assistant window, expand `Listeners` and select `Listener` (in the expandable tree menu on the left side of the window).

2. Select `Listening Locations` from the drop-down menu at the top right side of the window.
3. Choose the `Add Address` button at the bottom right side of the window.
4. Using the Protocol drop-down list, select TCI/IP with SSL; enter your host name and your chosen SSL port number.
5. Select `File–>Save Network Configuration`; your `listener.ora` file is updated. Exit Net8 Assistant.

Notes:

- Do not attempt to start the listener—until you have set up a wallet for SSL connections.
- Do not modify the value of `SSL_CLIENT_AUTHENTICATION` in `listener.ora`, which should be `FALSE` *(the listener is not doing the authentication—the database uses SSL to authenticate the client).*

---

6. For Microsoft NT only: you must manually edit `tnsnames.ora` for the client; edit `$ORACLE_HOME/network/admin/tnsnames.ora` by adding the following to your SSL service name:

   ```
   SECURITY = (AUTHENTICATION_SERVICE = TCPS)
   ```

Step 4: Review the .ORA Files

To facilitate review of your .ora files, some Microsoft NT examples follow:

Example: The SQLNET.ORA File

```
NAMES.DEFAULT_DOMAIN = WORLD
OSS.SOURCE.MY_WALLET =
    (SOURCE =
        (METHOD = FILE)
        (METHOD_DATA =
            (DIRECTORY = C:\WINNT\Profiles\DATABASES\oe)
        )
    )

SQLNET_AUTHENTICATION_SERVICES = (TCPS,NTS)
SSL_CLIENT_AUTHENTICATION = TRUE
```

```
SSL_VERSION = 0

SQLNET.CRYPTO_SEED = 4fhfquweotcadsfdsafjkdsfqp5f201p45mxskdlfdasf
```

> **Note:** The wallet location matches the one you entered in Net8 Assistant for the database.

Example: The TNSNAMES.ORA File:

```
OESSL.WORLD =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCPS) (HOST = host1) (PORT = 5000)
        )
        (CONNECT_DATE =
            (SERVICE_NAME = oe.world)
        )
        (SECURITY = (AUTHENTICATION_SERVICE = TCPS))
OE.WORLD =
    (DESCRIPTION =
        (ADDRESS_LIST =
            (ADDRESS = (PROTOCOL = TCP) (HOST = host1) (PORT = 1521)
        )
        (CONNECT_DATA =
            (SERVICE_NAME = oe.world)
        )
    )
```

Example: The LISTENER.ORA File:

```
OSS.SOURCE.MY_WALLET =
    (SOURCE =
        (METHOD = FILE)
        (METHOD_DATA =
            (DIRECTORY = C:\WINNT\Profiles\DATABASES\oe)
        )
    )


LISTENER =
    (DESCRIPTION_LIST =
        (DESCRIPTION =
            (ADDRESS = (PROTOCOL = TCP) (host = HOST1) (port = 1521)
```

```
        )
        (DESCRIPTION =
            (ADDRESS = (PROTOCOL = TCPS) (HOST = host1) (PORT = 5000))
        )
    )

SID_LIST_LISTENER =
    (SID_LIST =
        (SID_DESC =
            (GLOBAL_DBNAME = oe.world)
            (ORACLE_HOME = D:\Oracle\Ora81)
            (SID_NAME = oe)
        )
    )

SSL_CLIENT_AUTHENTICATION = FALSE
```

Note: SSL client authentication is FALSE because the listener does not need to authenticate the client; the database authenticates the client.

Task 5: Create and Configure the Wallet

To create and configure the wallet:

- ❏ Step 1: Create a Database Wallet
- ❏ Step 2: Open the Wallet
- ❏ Step 3: Perform Database Logout for Security
- ❏ Step 4: Verify Database Installation

Step 1: Create a Database Wallet

To create a database wallet:

1. Create a directory for the wallet for the location you specified in Step 1 on page 17-38.
2. Run Oracle Wallet Manager to create a new wallet for the database:

- Microsoft NT: Select
  `Start—>Programs—>Oracle-OracleHome81—>Network Administration—>Wallet Manager`
- UNIX: Enter `owm` at the command line.

3. Select `New` from the wallet menu. Do not create a new default directory when asked—this is for user wallets. During certificate request creation, type the **distinguished name (DN)** of the database *exactly*:

   `cn=`*`database_name`*`,cn=OracleContext,`*`administrative_context`*

   It is found in the initialization parameter file, in the parameter

   `RDBMS_SERVER_DN.`

   Note: The Distinguished Name is case-sensitive.

   For example:

   If the global database name chosen during installation is `sales.us.acme.com`, and the administrative context selected within Net8CA is `ou=division1,c=us,o=acme,` the complete DN of the database that you enter into Oracle Wallet Manager is:

   `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** (CA).
5. Get the certificate text for the CA ***trusted certificate***. The CA trusted certificate is sometimes called a **root key certificate**.

   The certificate text includes the lines `BEGIN CERTIFICATE` and `END CERTIFICATE` and the text between them.

6. Import or paste the trusted 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 between them.

8. Import or paste the database certificate text into the wallet using the Oracle Wallet Manager Import User Certificate function.

Step 2: Open the Wallet

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 the following at the command line:
   - Microsoft NT: `lsnrctl stop`
   - UNIX: `lsnrctl stop`

   Note for NT: You can alternatively stop Oracle Listener Services in the Services Control Panel. See: Notes on page 17-46.

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 completed Step 1 on page 17-38. For verification, check that there is a `cwallet.sso` file in the wallet directory.

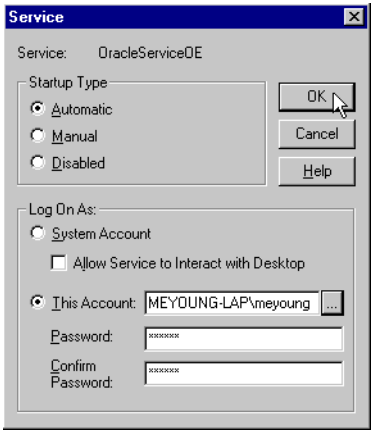
   Note: End users never have to use Oracle Wallet Manager, because Oracle Enterprise Login Assistant can be used to enable and disable autologin.

4. **Change Oracle Services Login (Microsoft NT only)**. Because the database and the listener services are running as system (with few privileges in NT), and the wallets are opened under your user name, the database and the listener are not able to read the wallet. In order for them to read their wallet, they must be changed to log on as the user who enabled autologin for the database wallet.

   To change the Oracle Services login:
   - Shut down the database by opening the Services control panel and selecting `OracleService <database name>;` choose the `Stop` button; choose `Yes` to confirm.

- Choose the `Startup` button.
- In the `Log On As` region of the Service Window (Figure 17–10):

Figure 17–10 The Oracle Service Window



Choose `This Account` and enter `<domain>\< NT user login>` for the user who enabled autologin for the database wallet; alternatively, you can choose the browse button (...) to select from a list; enter your password in the `Password` and `Confirm Password` fields; choose OK.

- Choose `Start` to start up the Oracle database.
- Repeat these steps (starting with *Choose the Startup Button*) for `OracleOraHome81TNSListener`; do *not* start the listener service.
- Close the Services window.

5. Restart the listener by entering the following at the command line:
   - Microsoft NT: `lsnrctl start`
   - UNIX: `lsnrctl start`

Notes:

- If the Listener starts correctly, it confirms that it is listening on TCPS, on the command line.
- If there are errors when you attempt to start the Listener, you may not have selected `Autologin`, or the wallet may be in the wrong location.
- Under Microsoft NT, you can also run the OracleListener Service under the services control panel to start and stop the listener—but without the command line response that confirms listener activity.

The database wallet is now open, and the database is able to participate in authenticated communications using SSL; the `OracleTNSListener` service is also started.

See Also: Chapter 18, Using Oracle Wallet Manager, for detailed instructions about creating a wallet.

Step 3: Perform Database Logout for Security

Important: If the database will be shut down for an extended period of time, perform a database logout and close the wallet for security purposes.

To log out of the database:

1. Stop the listener by entering the following at the command line
   - Microsoft NT: `lsnrctl stop`
   - UNIX: `lsnrctl stop`
2. Start Oracle Wallet Manager by entering:
   - Microsoft NT: Select `Start–>Programs–>Oracle-OraHome81–>Network Administration–>Wallet Manager`
   - UNIX: Enter `owm` at the command line.

3. Clear the Autologin check box.
4. Save your changes.
5. Restart the listener by entering the following at the command line:
   - Microsoft NT: `lsnrctl start`
   - UNIX: `lsnrctl start`

Step 4: Verify Database Installation

To verify that the database was successfully installed:

1. Verify that there is a `cwallet.sso` file located in the database wallet directory. If not, Autologin was not successfully enabled. If this happens, go back to the Oracle Wallet Manager, open the wallet, select the Autologin check box, and save the wallet.
2. Verify that there is an `ldap.ora` file located in

   `$ORACLE_HOME/network/admin`

   If there is no `ldap.ora` file, the Net8CA failed. Verify that the `ORACLE_HOME` is set and `TNS_ADMIN` is not set. Rerun Net8CA.
3. Use the directory administration tool to verify that a database entry exists under the Oracle Context you specified when you ran the Net8CA. If you do not find the database entry, verify that the directory is running, the Oracle Context is set up, and the `ldap.ora` file exists and is correct. Then register the database again, using DBCA.

Task 6: Create Global Schemas and Roles

*Although this step can be completed using Oracle Enterprise Manager, the following procedures use SQL*Plus directly.*

To create global schemas and roles:

- ❏ Step 1: Create a Shared Schema
- ❏ Step 2: Grant a Create Session Privilege
- ❏ Step 3: Create Global Roles
- ❏ Step 4: Associate Privileges

Step 1: Create a Shared Schema

Using SQL*Plus, create a shared schema (called *Guest*, for example) for enterprise users by entering:

```
CREATE USER guest IDENTIFIED GLOBALLY AS ''
```

Note the two single quotation marks with no space between them at the end of the line. If you enter a specific **distinguished name (DN)** between the quotation marks, only that user is able to connect to that schema.

Step 2: Grant a Create Session Privilege

Users connecting to this schema require a `CREATE SESSION` privilege. You can grant a `CREATE SESSION` privilege either to the guest schema, or to a **global role** which you grant to specific users though an enterprise role.

Step 3: Create Global Roles

Create global roles for the database to hold relevant privileges. These roles are associated with enterprise roles to be created later. Enterprise roles are allocated to users.

For example:

```
CREATE ROLE emprole IDENTIFIED GLOBALLY;

CREATE ROLE custrole IDENTIFIED GLOBALLY;
```

Step 4: Associate Privileges

Associate privileges with the new global roles.

For example:

```
GRANT select ON products TO custrole, emprole;
```

Task 7: Configure Database Clients

Once you have installed Oracle8*i* clients, configure Net8 on the clients by using Net8 Assistant. You may complete this step during or after installation of Oracle8*i* Release 8.1.7.

Because you will be using an LDAP directory service for enterprise security, you may also want to use Net8 directory naming. Net8 directory naming lets the client connect to the database using the database entry registered with the directory by Oracle Database Configuration Assistant. Alternatively, you can use one of the

other Net8 naming methods, such as *local naming* (`tnsnames.ora` file), to configure a net service name for the database.

Use SSL to enable clients to connect and authenticate to a database. Use Net8 Assistant to configure SSL on UNIX; configure an SSL net service name, as described by Step 2 on page 17-39 and Table 17–2. Do not enter a wallet location when configuring a client profile. The lack of a specific wallet location indicates that SSL should find the default wallet for the current OS user. In this way, the `sqlnet.ora` file can be shared by enterprise users, providing easier administration and deployment. If you use a non-default wallet location, you must create separate `sqlnet.ora` files for each user.

Note: If you do not install clients, and `ORACLE_HOME` is set to a database server `ORACLE_HOME`, you must create at least one new `TNS_ADMIN` directory with a `sqlnet.ora` file—with *no* wallet location. This ensures that SSL uses the default location of the wallet for the operating system user.

To Create Wallet Directories for the User Wallets:

- Microsoft NT:

  ```
  c:\winnt\profiles\<user>\ORACLE\WALLETS
  ```

- UNIX:

  ```
  /etc/ORACLE/WALLETS/<OS username>
  ```

Note: Wallets for specific users are set up when you create enterprise users. See Chapter 20, Using Oracle Enterprise Security Manager, for instructions about creating enterprise users.

See Also:

- *Net8 Administrator's Guide*
- Chapter 9, Configuring Secure Socket Layer Authentication, for information about configuring SSL

Task 8: Configure an Enterprise Domain

Oracle Enterprise Security Manager is installed automatically as part of the Oracle8*i* installation, and can be used to configure an enterprise domain. Note that the Oracle default domain is created by default when the Oracle Context is created in the directory, and databases are automatically added as members of that domain when they are registered by DBCA. Table 17–3 lists the steps required to set up an enterprise domain, and cross-references related instructions. If you are using the Oracle default domain, you can skip steps 1 and 4.

Table 17–3 Setting up an Enterprise Domain

Step	Related Instructions
1. Create an enterprise domain.	Creating an Enterprise Domain on page 20-15.
2. Make the enterprise domain trusted / untrusted. Only a trusted domain permits current user database links between member databases.	Creating an Enterprise Domain on page 20-15
3. Create enterprise roles in the domain.	Creating an Enterprise Role within an Enterprise Domain on page 20-17
4. 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
5. Create global roles on the databases. The SQL*Plus command is: `CREATE ROLE` *`rolename`* `IDENTIFIED GLOBALLY`	■ *Oracle8i SQL Reference* ■ See Also: Step 3: Create Global Roles on page 17-48
6. Assign a global role to each enterprise role.	Creating an Enterprise Role within an Enterprise Domain on page 20-17

Task 9: Configure Enterprise Users

To create a new enterprise user:

- ❏ Step 1: Add a New Enterprise User to the Directory
- ❏ Step 2: Create a User Wallet
- ❏ Step 3: Authorize the User

Step 1: Add a New Enterprise User to the Directory

Any directory user can be an enterprise user. You can add users to the directory by using one of the following tools:

- Oracle Enterprise Security Manager
- The administration tool for your directory service
- The standard LDAP command line tools

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 20-28 for instructions about adding new enterprise users to the directory by using Oracle Enterprise Security Manager
- Documentation for your directory service for information about using the directory administration tools

Step 2: Create a User Wallet

To create a user wallet, refer to Chapter 18, Using Oracle Wallet Manager.

Note: Store the user wallet in the default user wallet location:

- Microsoft NT: `x:\winnt\profiles\<os user>\ORACLE\WALLETS`
- UNIX: `/etc/ORACLE/WALLETS/<os user>`

Step 3: Authorize the User

You can do either or both of the following:

- Local Oracle role authorization:

  Use Oracle Enterprise Manager or SQL*Plus to grant local roles and privileges to the database user or schema; this step is optional.

- Enterprise role authorization:

  Use Oracle Enterprise Security Manager to grant enterprise roles to the enterprise user in the directory.

If you are using a shared schema, use Oracle Enterprise Security Manager to map the user to a schema. You can choose either of the following mapping options:

- Database:

  Applies to one database.

- Domain:

  Applies to all databases in the domain.

  For example:

  If you are creating a domain mapping from three users to a shared schema called *guest*, and you have more than one database in the domain, each database must have a shared schema (called guest) for the users to connect to. These three users will not be able to connect to any database in the domain that does not have a shared schema called guest.

  Alternatively, you can create a mapping under a particular database. If you do it this way, the mapping applies only to that database, and not to all databases in the domain. If you have mappings in both places, the database mapping takes precedence.

  See Also:

  - Task 8: Configure an Enterprise Domain, for information about setting up an enterprise domain
  - Administering Enterprise Users on page 20-24

Task 10: Log In as the Enterprise User

To log in as the Enterprise User:

- ❏ Step 1: Open the User Wallet
- ❏ Step 2: Connect to the Database

Step 1: Open the User Wallet

The enterprise user must open the wallet created in Task 9 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. Log in to the operating system as the appropriate user.
2. To open the user wallet, run Oracle Enterprise Login Assistant by entering one of the following commands:
   - Microsoft NT: `Start–>Programs–>Oracle - OraHome81–>Network Administration–>Enterprise Login Assistant`
   - UNIX: Enter `elogin` at the command line

   The Oracle Enterprise Login Assistant window appears (Figure 17–11):

Figure 17–11 The Oracle Enterprise Login Assistant Window

3. A red light indicates that you have not logged into the enterprise; select `AutoLogin–>Login`, and enter the wallet password.
4. The green light should now be on; save your changes. Your wallet is open, and you have authenticated access using the SSL adapter.
5. Set `ORACLE_HOME`.

   If the `ORACLE_HOME` is set to a *server* `ORACLE_HOME`, you must set the `TNS_ADMIN` environment variable to address the directory where you placed the `sqlnet.ora` file—that you created in Task 7: Configure Database Clients on page 17-48.

   If you have a separate client `ORACLE_HOME`, you do not need to set the `TNS_ADMIN` environment variable.

Step 2: Connect to the Database

Launch SQL*Plus and enter:

```
sqlplus/@connect_identifier
```

where `connect_identifier` is the net service name you set up in Task 7: Configure Database Clients on page 17-48. If you are using Net8 directory naming, the connect identifier is the simple database name.

The system should respond `Connected to:`...; this is the principal confirmation of a successful connection and setup. If an error message is displayed, see: Troubleshooting Enterprise User Login on page 17-55.

If you do connect successfully, check that the appropriate global roles were retrieved from the directory by entering:

```
select * from session_roles
```

In the Oracle Enterprise Login Assistant, select `Autologin > Logout` to disable authentication with the SSL adapter.

> **See Also:** Chapter 19, Using Oracle Enterprise Login Assistant, for instructions about using Oracle Enterprise Login Assistant.

Troubleshooting Enterprise User Login

This section describes potential problems and associated corrective actions.

No Global Roles

The following tips help you verify that the user has been allocated the correct global roles upon database login and, if necessary, help determine the cause of failure:

1. Check for the existence of global roles. Enter the following, including the semi-colon (;):

   ```
   SELECT * FROM session_roles;
   ```

2. If there are no roles, one of the following applies:

   - The roles were not allocated to an enterprise role in Oracle Enterprise Security Manager.
   - The enterprise role was not assigned to the user in Oracle Enterprise Security Manager.
   - The database and Oracle Enterprise Security Manager have different values for the database domain; shut down and restart the database to update the database internal value.
   - Your database does not have proper permissions in the directory to see the roles. These permissions are created automatically, so it is possible that the **distinguished name (DN)** in the database certificate does not match the distinguished name registered for the database. In this case, the directory does not recognize the database as the proper entity, and denies access.

     Do an LDAP search to display the appropriate roles by entering the following:

     ```
     ldapsearch -h <directory hostname> P <SSL directory port
     number> -U 3

     -W "file:<walletpath>" -P [database wallet password]

     -b "[n=oracleDBSecurity, cn=Products, cn=OracleContext,
     [admin context]"

     "objectclass=orclDBenterprisedomain"
     ```

If you do not see the roles, the database is not in the correct domain—or there is an incorrect distinguished name (DN) in the database wallet certificate. If the database appears to be receiving information from the wrong domain, try restarting the database to update its internal domain membership information.

TNS Lost Connection

This error may indicate that you attempted to configure a domestic cipher suite. Run the Net8 Assistant again, and be sure that you choose the Show Domestic Cipher Suites button.

ORA-1004: Default username feature not supported

This error indicates that the connection was not over SSL. Look at the `tnsnames.ora` file to verify the protocol value of the net service name that you are using. The value must be `TCPS` and not `TCP`.

ORA-1017: Invalid username/password

The distinguished name that the wallet uses to connect does not match the DN in the `CREATE USER` statement for any schema in the database, and it does not match the DN in any relevant mapping.

1. Check the DN of the user in the mapping created using Oracle Enterprise Security Manager.
2. Also check that your directory is actually listening properly for incoming SSL connections. From a command prompt, enter:

   ```
   set NLS_LANG=AMERICAN_AMERICA.UTF8
   ```

   ```
   ldapbind -h <directory hostname> -p <directory SSL port
   number> -U 3 -W "file:[database wallet path]" -P [database
   wallet password]
   ```

   `Bind successful` should be displayed. If the bind fails, try restarting the SSL instance of your directory.

   Then try the bind again.

   If it still doesn't work, carefully check the wallet location in the configuration set via Oracle Directory Manager. Make sure that it is set to the proper path name.

Microsoft NT Example: Set the wallet location path name to:

```
file:c:\winnt\profiles\oid
```

where `oid` represents your directory mnemonic.

Important: You *must* get this `ldapbind` to work. If it does not work, *do not continue.*

3. If the prior steps do not work, circumvent the user-schema mapping step by altering the user `guest` to be a non-shared schema. In sqlplus as
4. `system/manager@database_name`, enter:

   ```
   alter user guest identified globally as 'cn=oe20emp,c=us';
   ```

   and then try the `connect /@connect_identifier` again. If this succeeds, the problem is associated with the mapping of the user to the schema; use Oracle Enterprise Security Manager to check that mapping in the directory.

   Alter this user back to a shared schema by entering:

   ```
   alter user guest identified globally as '';
   ```

ORA-12560: Protocol adapter error

This error usually means that something is wrong with the wallet. Look in the `sqlnet.log` file in the current operating system directory for more information. Also, on Microsoft NT, this can mean that the Oracle service has stopped; check the Services control panel.

Decryption of Encrypted Private Key Fails

This error occurs when you attempt to open a wallet that you are not allowed to open.

For Example:

- You are logged into the system as `user-x`, but you do not have a local `sqlnet.ora` file that identifies `c:\winnt\profiles\user-x\oracle\wallets` as your wallet location.

- SSL uses the `sqlnet.ora` file in the default location to find the wallet location, and then tries to open the database wallet to get your login credentials.
- This attempt *fails*, because `user-x` does not have permission to open the database wallet.

ORA-28030

This is a catch-all Oracle8*i* error that indicates something unanticipated went wrong with the RDBMS to LDAP directory query.

Tracing

You can use tracing to help debug. This is appropriate if the *ldapbind* (See: **ORA-1017: Invalid username/password**) fails, indicating that the directory's SSL instance is not working properly.

Oracle Internet Directory

If you are using Oracle Internet Directory as your ldap directory, use the following tracing procedure:

1. Turn on tracing in Oracle Internet Directory.
2. At the command line, stop the SSL Oracle Internet Directory instance, and restart it with debug flags ON:

   ```
   oidctl conn=oiddb1 server=oidldapd instance=2 stop

   oidctl conn=oiddb1 server=oidldapd instance=2 conf=1

   flags="debug=65535" start
   ```

   This starts up the SSL Oracle Internet Directory instance in full debug mode. Log files will be written to `$ORACLE_HOME\ldap\log`. Look at the file with an `02` and an `s` in its filename, because it is for the instance 2 server. The log files without the `s` are for the monitor process (`oidmon`) and the dispatcher. Look at the end of the log file immediately after you have tried your `connect /@connect_identifier`. One thing to look for is the string `Distinguished Name` to ensure that it matches the DN of your user.

   To turn off Oracle Internet Directory tracing, restart via `oidctl` without the `flags` parameter.

18

Using Oracle Wallet Manager

Security administrators use Oracle Wallet Manager to manage public-key security credentials on Oracle clients and servers. The wallets it creates are opened by using either the Oracle Enterprise Login Assistant or the Oracle Wallet Manager.

This chapter describes the Oracle Wallet Manager, in the following sections:

- Overview
- Managing Wallets
- Managing Certificates

> **See Also:** Chapter 19, Using Oracle Enterprise Login Assistant, for information about how to open and close wallets for secure SSL communications using Oracle Enterprise Login Assistant

Overview

Traditional private-key or symmetric-key cryptography requires that entities desiring to establish secure communications possess a single secret key known only to them. *Harriet* and *Dick*, for example, could agree to shift each letter in their private messages by two character positions (A becomes C, B becomes E, and so on) to encrypt the message text. Using this method, a *HELLO* message from Harriet to Dick would read *JGNNP*. The actual encryption methods in current use are much more complex and significantly more secure, but an underlying problem remains—sending messages encrypted with a single key requires prior, *secure* distribution of the key to each participating party. Otherwise, a malicious third party might obtain the key, intercept communications, and compromise security. Public-key cryptography addresses this problem, by providing a secure method for key distribution.

Public-key cryptography requires a party to possess a **public/private key pair**. The **private key** is kept secret and is known only to that party. The **public key**, as the name implies, is freely available. To send a secret message to this party requires that a third party sender encrypt the message with the public key. Such a message can only be decrypted by a party holding the associated private key.

For example, when Dick wants to send a secure message to Harriet, he first asks Harriet for her public key (or obtains it from another, public source). Harriet gives Dick the public key, but Tom, a malicious eavesdropper, also obtains the public key. Nevertheless, when Dick sends Harriet a message encrypted with her public key, Tom cannot decrypt it; the message can only be decrypted with Harriet's private key.

Public-key algorithms thus guarantee the secrecy of a message, but they don't guarantee *secure communications* because they don't verify the identities of the communicating parties. In order to establish secure communications, it is important to verify that the public key used to encrypt a message does in fact belong to the target recipient. Otherwise, a third party can potentially eavesdrop on the communication and intercept public key requests, substituting its public key for a legitimate key.

If Tom, for example, is able to substitute his public key for Harriet's public key and send it to Dick, Dick might then send a message to Harriet encrypted with Tom's public key—believing he was using Harriet's public key. Tom could then decrypt a subsequent intercepted message from Dick using his private key, re-encrypt it with Harriet's public key and re-transmit it to Harriet. Harriet could then decrypt the incoming message using her private key, and never know that it had been intercepted by Tom—the **man-in-the-middle**.

In order to avoid such a man-in-the-middle attack, it is necessary to verify the owner of the public key, a process called **authentication**. This authentication can be accomplished through a **certificate authority** (CA).

A CA is a third party that is trusted by both of the parties attempting secure communication. The CA issues public key certificates that contain an entity's name, public key, and certain other security credentials. Such credentials typically include the CA name, the CA signature, and the certificate effective dates (From Date, To Date).

The CA uses its private key to encrypt a message, while the public key is used to decrypt it, thus verifying that the message was encrypted by the CA. The CA public key is well known, and does not have to be authenticated each time it is accessed. Such CA public keys are stored in an Oracle **wallet**.

Oracle Wallet Manager is a stand-alone Java application that wallet owners use to manage and edit the security credentials in their Oracle wallets. These tasks include the following:

- ❏ Generating a public/private key pair and creating a certificate request for submission to a CA.
- ❏ Installing a certificate for the entity.
- ❏ Configuring **trusted certificate**s for the entity.
- ❏ Opening a wallet to enable access to PKI-based services.
- ❏ Creating a wallet that can be accessed by using either Oracle Enterprise Login Assistant or Oracle Wallet Manager.

Managing Wallets

This section describes how to create a new wallet and perform associated wallet management tasks, such as generating certificate requests, exporting certificate requests, and importing certificates into wallets, in the following subsections:

- Starting Oracle Wallet Manager
- Creating a New Wallet
- Opening an Existing Wallet
- Closing a Wallet
- Saving Changes
- Saving the Open Wallet to a New Location
- Saving in System Default
- Deleting the Wallet
- Changing the Password
- Using Auto Login
- Using Oracle Wallet Manager with Oracle Application Server

Starting Oracle Wallet Manager

To start Oracle Wallet Manager:

- Microsoft NT: Select `Start–>Programs–>Oracle-OraHome81–>Network Administration–>Wallet Manager`
- UNIX: Enter `owm` at the command line.

Creating a New Wallet

Create a new wallet as follows:

1. Choose `Wallet > New` from the menu bar; the New Wallet dialog box appears.
2. Read the recommended guidelines for creating a password and enter a password in the Wallet Password field.

   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. A malicious user who guesses the password to a user's wallet can access all the databases that the user can access.

Oracle Corporation recommends that you choose a password that is not too short, not easily guessed, and is reasonably complex. A reasonably complex password has at least six characters, and contains at least one symbol or number—so that it will not be found in a dictionary.

Example: `gol8fer`

It is also a prudent security practice for users to change their passwords periodically, such as once a month, or once a quarter.

3. Re-enter that password in the Confirm Password field.
4. Choose `OK` to continue.
5. An Alert is displayed, and informs you that a new empty wallet has been created. It prompts you to decide whether you want to create a certificate request. See: **Creating a Certificate Request** on page 18-9.

   If you choose Cancel, you are returned to the Oracle Wallet Manager main window. The new wallet you just created appears in the left window pane. The certificate has a status of `Empty`, and the wallet displays its default trusted certificates.
6. Select `Wallet > Save In System Default` to save the new wallet.

   If you do not have permission to save the wallet in the system default, you can save it to another location.

   A message at the bottom of the window informs you that the wallet was successfully saved.

Opening an Existing Wallet

Open a wallet that already exists in the file system directory as follows:

1. Choose `Wallet > Open` from the menu bar; the Select Directory dialog box appears.
2. Navigate to the directory location in which the wallet is located, and select the directory.
3. Choose OK; the Open Wallet dialog box appears.
4. Enter the wallet password in the Wallet Password field.

5. Choose OK.
6. The message `Wallet opened successfully` appears at the bottom of the window, and you are returned to the Oracle Wallet Manager main window. The wallet's certificate and its trusted certificates are displayed in the left window pane.

Closing a Wallet

To close an open wallet in the currently selected directory:

- Choose `Wallet > Close`.
- The message `Wallet closed successfully` appears at the bottom of the window, to confirm that the wallet is closed.

Saving Changes

To save your changes to the current open wallet:

- Choose `Wallet > Save`.
- A message at the bottom of the window confirms that the wallet changes were successfully saved to the wallet in the selected directory location.

Saving the Open Wallet to a New Location

Use the `Save As` option to save the current open wallet to a new directory location:

1. Choose `Wallet > Save As;` the select directory dialog box appears.
2. Select a directory location to save the wallet.
3. Choose OK.

   The following message appears if a wallet already exists in the selected directory:

   ```
   A wallet already exists in the selected path. Do you
   want to overwrite it?.
   ```

   Choose Yes to overwrite the existing wallet, or No to save the wallet to another directory.

   A message at the bottom of the window confirms that the wallet was successfully saved to the selected directory location.

Saving in System Default

Use the `Save in System Default` menu option to save the current open wallet to the system default directory location. This makes the current open wallet the wallet that is used by SSL:

- Choose `Wallet > Save in System Default`.
- A message at the bottom of the window confirms that the wallet was successfully saved in the system default wallet location.

Deleting the Wallet

To delete the current open wallet:

1. Choose `Wallet > Delete`; the `Delete Wallet` dialog box appears.
2. Review the displayed wallet location to verify you are deleting the correct wallet.
3. Enter the wallet password.
4. Choose `OK`; a dialog panel appears to inform you that the wallet was successfully deleted.

Note: Any open wallet in application memory will remain in memory until the application exits. Therefore, deleting a wallet that is currently in use does not immediately affect system operation.

Changing the Password

A password change is effective immediately. The wallet is saved to the currently selected directory, with the new encrypted password.To change the password for the current open wallet:

1. Choose `Wallet > Change Password;` the `Change Wallet Password` dialog box appears.
2. Enter the existing wallet password.
3. Enter the new password.
4. Re-enter the new password.
5. Choose `OK`.

A message at the bottom of the window confirms that the password was successfully changed.

Using Auto Login

The Oracle Wallet Manager Auto Login feature opens a copy of the wallet and enables PKI-based access to secure services—as long as the wallet in the specified directory remains open in memory.

You must enable Auto Login if you want single sign-on access to multiple Oracle databases.

Enabling Auto Login

To enable Auto Login:

1. Choose `Wallet` from the menu bar.
2. Choose the check box next to the Auto Login menu item; a message at the bottom of the window displays `Autologin enabled`.

Disabling Auto Login

To disable Auto Login:

1. Choose `Wallet` from the menu bar.
2. Choose the check box next to the Auto Login menu item; a message at the bottom of the window displays `Autologin disabled`.

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.

Managing Certificates

Oracle Wallet Manager uses two kinds of certificates: user certificates and trusted certificates. This section describes how to manage both certificate types, in the following subsections:

- Managing User Certificates
- Managing Trusted Certificates

Note: You must first install a trusted certificate from the certificate authority before you can install a user certificate issued by that authority. Several trusted certificates are installed by default when you create a new wallet.

Managing User Certificates

Managing user certificates involves the following tasks:

- 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 cannot edit an existing certificate request; store only a correctly filled out certificate request in a wallet.

To create a PKCS #10 certificate request:

1. Choose `Operations > Create Certificate Request`; the `Create Certificate Request` dialog box appears.
2. Enter the following information (Table 18–1):

Table 18–1 Certificate Request: Fields and Descriptions

Field Name	Description
Common Name	Mandatory. Enter the name of the user's or service's identity. Enter a user's name in first name /last name format.

Table 18–1 Certificate Request: Fields and Descriptions

Field Name	Description
Organizational Unit	Optional. Enter the name of the identity's organizational unit. Example: Finance.
Organization	Optional.Enter the name of the identity's organization. Example: XYZ Corp.
Locality / City	Optional. Enter the name of the locality or city in which the identity resides.
State / Province	Optional. Enter the full name of the state or province in which the identity resides. Enter the full state name, because some certificate authorities do not accept two–letter abbreviations.
Country	Mandatory. Choose the drop-down list to view a list of country abbreviations. Select the country in which the organization is located.
Key Size	Mandatory. Choose the drop-down box to view a list of key sizes to use when creating the public / private key pair.
Advanced	Optional. Choose Advanced to view the Advanced Certificate Request dialog panel. Use this field to edit or customize the identity's distinguished name (DN). For example, you can edit the full state name and locality.

3. Choose OK. An Oracle Wallet Manager dialog box informs you that a certificate request was successfully created. You can either copy the certificate request text from the body of this dialog panel and paste it into an e-mail message to send to a certificate authority, or you can export the certificate request to a file.
4. Choose 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

Save the certificate request in a file system directory when you elect to export a certificate request:

1. Choose `Operations > Export Certificate Request` from the menu bar; the Export Certificate Request dialog box appears.
2. Enter the file system directory in which you want o save your certificate request, or navigate to the directory structure under Folders.
3. Enter a file name to save your certificate request, in the Enter File Name field.
4. Choose `OK`. A message at the bottom of the window confirms that the certificate request was successfully exported to the file. You are returned to the Oracle Wallet Manager main window.

Importing the User Certificate into the Wallet

You will receive an e-mail notification from the certificate authority informing you that your certificate request has been fulfilled. Import the certificate into a wallet in either of two ways: copy and paste the certificate from the e-mail you receive from the certificate authority, or import the user certificate from a file.

Pasting the Certificate

To paste the certificate:

1. Copy the certificate text from the e-mail message or file you receive from the certificate authority. Include the lines `Begin Certificate` and `End Certificate.`
2. Choose `Operations > Import User Certificate` from the menu bar; the Import Certificate dialog box appears.
3. Choose the `Paste the Certificate` button, and choose `OK`; an Import Certificate dialog box appears with the following message:

   ```
   Please provide a base64 format certificate and paste it below.
   ```

4. Paste the certificate into the dialog box, and choose `OK`. A message at the bottom of the window confirms that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the wallet status changes to `Ready`.

Selecting a File that Contains the Certificate

To select the file:

1. Choose `Operations > Import User Certificate` from the menu bar.
2. Choose the `Select a file...` certificate button, and choose `OK`; the Import Certificate dialog box appears.
3. Enter the path or folder name of the certificate location.
4. Select the name of the certificate file (for example, `cert.txt`).
5. Choose `OK`. A message at the bottom of the window appears, to inform you that the certificate was successfully installed. You are returned to the Oracle Wallet Manager main panel, and the wallet status is changes to `Ready`.

Removing a User Certificate from a Wallet

1. Choose `Operations > Remove User Certificate`; a dialog panel appears and prompts you to verify that you want to remove the user certificate from the wallet.
2. Choose `Yes`; you are returned to the Oracle Wallet Manager main panel, and the certificate displays a status of `Requested`.

Managing Trusted Certificates

Managing trusted certificates includes the following tasks:

- ❏ Importing a Trusted Certificate
- ❏ Removing a Trusted Certificate
- ❏ Exporting a Trusted Certificate
- ❏ Exporting All Trusted Certificates
- ❏ Exporting a Wallet

Importing a Trusted Certificate

You can import a trusted certificate into a wallet in either of two ways: paste the trusted certificate from an e-mail that you receive from the certificate authority, or import the trusted certificate from a file.

Oracle Wallet Manager automatically installs trusted certificates from VeriSign, RSA, and GTE CyberTrust when you create a new wallet.

Pasting the Trusted Certificate To paste the trusted certificate:

1. Choose `Operations > Import Trusted Certificate` from the menu bar; the Import Trusted Certificate dialog panel appears.
2. Choose the `Paste the Certificate` button, and choose `OK`. An Import Trusted Certificate dialog panel appears with the following message:

   ```
   Please provide a base64 format certificate and paste it
   below.
   ```

3. Copy the trusted certificate from the body of the e-mail message you received that contained the user certificate. Include the lines `Begin Certificate` and `End Certificate.`
4. Paste the certificate into the window, and Choose `OK`. A message at the bottom of the window informs you that the trusted certificate was successfully installed.
5. Choose `OK`; you are returned to the Oracle Wallet Manager main panel, and the trusted certificate appears at the bottom of the Trusted Certificates tree.

Selecting a File that Contains the Trusted Certificate

To select the file:

1. Choose `Operations > Import Trusted Certificate` from the menu bar. The Import Trusted Certificate dialog panel appears.
2. Enter the path or folder name of the trusted certificate location.
3. Select the name of the trusted certificate file (for example, `cert.txt`).
4. Choose `OK`. A message at the bottom of the window informs you that the trusted certificate was successfully imported into the wallet.
5. Choose `OK` to exit the dialog panel; you are returned to the Oracle Wallet Manager main panel, and the trusted certificate appears at the bottom of the Trusted Certificates tree.

Removing a Trusted Certificate

To remove a trusted certificate from a wallet:

1. Select the trusted certificate listed in the Trusted Certificates tree.
2. Choose `Operations > Remove Trusted Certificate` from the menu bar.

A dialog panel warns you that your user certificate will no longer be verifiable by its recipients if you remove the trusted certificate that was used to sign it.

3. Choose `Yes`; the selected trusted certificate is removed from the Trusted Certificates tree.

Note: A certificate that is signed by a trusted certificate is no longer verifiable when you remove it from your wallet.

Also, you cannot remove a trusted certificate if it has been used to sign a user certificate that is still present in the wallet. To remove such a trusted certificate, you must first remove the certificates that it has signed.

Exporting a Trusted Certificate

To export a trusted certificate to another file system location:

1. Select `Operations > Export Trusted Certificate`; the Export Trusted Certificate dialog box appears.
2. Select a file system directory to save your trusted certificate, or choose `Browse` to display the directory structure.
3. Enter a file name to save your trusted certificate.
4. Choose `OK`; you are returned to the Oracle Wallet Manager main window.

Exporting All Trusted Certificates

To export all of your trusted certificates to another file system location:

1. Choose `Operations > Export All Trusted Certificates`. The Export Trusted Certificate dialog box appears.
2. Select the file system directory to save your trusted certificates, or choose `Browse` to display the directory structure.
3. Enter a file name to save your trusted certificates.
4. Choose `OK`; you are returned to the Oracle Wallet Manager main window.

Exporting a Wallet

You can export a wallet to text-based PKI formats. Individual components are formatted according to the following standards (Table 18–2):

Table 18–2 PKI Wallet Encoding Standards

Component	Encoding Standard
Certificate chains	X509v3
Trusted certificates	X509v3
Private keys	PKCS5

19

Using Oracle Enterprise Login Assistant

Use the Oracle Enterprise Login Assistant to open and close existing wallets, and to enable or disable secure SSL-based communications.

This chapter describes Oracle Enterprise Login Assistant, in the following sections:

- About Oracle Enterprise Login Assistant
- Starting Oracle Enterprise Login Assistant
- Enabling Automatic Login
- Disabling Automatic Login
- Changing a Wallet Password

> **See Also:** Chapter 18, Using Oracle Wallet Manager, for information about managing wallets with Oracle Wallet Manager.

About Oracle Enterprise Login Assistant

Oracle Enterprise Login Assistant (ELA) simplifies the use of wallets and certificates created by Oracle Wallet Manager. This login tool provides easy access to existing wallets and certificates, while masking their underlying complexity. Once users securely open their wallets using ELA, they can connect to multiple databases using SSL, without providing additional passwords. This provides strong authentication as well as single sign-on (SSO) capability.

Starting Oracle Enterprise Login Assistant

Refer to your platform-specific documentation for instructions about how to start Oracle Enterprise Login Assistant.

Enabling Automatic Login

The ELA Automatic Login feature enables applications running on a server or a client to revalidate themselves to the other end of the connection without user intervention. Users can thus obtain SSO capability, using the credentials contained in their wallets to authenticate multiple applications using SSL.

To enable secure SSL-based communications using the default wallet:

1. Choose `AutoLogin > Login` from the menu bar; the login dialog box appears.
2. Enter the wallet password.
3. Choose `OK`.
4. ELA creates an obfuscated copy of the wallet on the file system, and you are returned to the ELA window. An `Autologin enabled` message confirms successful connection.

Disabling Automatic Login

Use the ELA to disable single sign-on communications from server side applications to the client. To log out:

1. Choose `AutoLogin > Logout` from the menu bar; a dialog box displays the following warning:

   ```
   If you log out, your applications will no longer use the
   security credentials of your wallet.
   ```

2. Choose `Yes` to continue; you are returned to the ELA window. The message `Autologin not enabled` appears at the bottom of the window.

Changing a Wallet Password

Change a wallet password according to company policy, or whenever a password has been compromised. 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.

To change a wallet password:

1. Choose `AutoLogin > Change Password` from the menu bar; the Change Password dialog box appears.
2. Enter the *existing* password in the Old Password field.
3. Read the text that describes how to create more secure passwords, and enter the *new* password in the New password field.
4. Enter the *new* password again in the Confirm password field.
5. Choose `OK` to continue; a dialog box displays the message `Password changed successfully.`
6. Choose `OK` to exit this dialog box.

20

Using Oracle Enterprise Security Manager

This chapter describes how to use Oracle Enterprise Security Manager to administer database security in an **enterprise domain** of Oracle8*i* databases. It contains the following sections:

- Introduction
- Installing and Configuring Oracle Enterprise Security Manager
- Navigating Oracle Enterprise Security Manager
- Administering Enterprise Databases, Domains, and Users
- Managing Security Administrators

Introduction

Oracle Enterprise Security Manager is an administration tool that provides a graphical user interface to manage **enterprise user**s, enterprise domains, databases, and **enterprise role**s that are held in a directory server.

Installing and Configuring Oracle Enterprise Security Manager

The following tasks describe how to use Oracle Enterprise Security Manager to install Oracle Management Server and Oracle Enterprise Manager:

- ❏ Task 1: 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

Oracle Enterprise Security manager is automatically installed 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. For each database there should be a TNS alias that matches 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 to be managed. 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*.

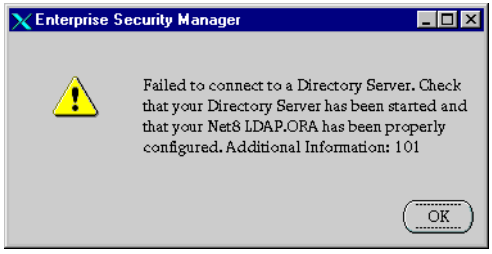
Use the Net8 Configuration Assistant to set up directory access. This creates an `ldap.ora` file on `ORACLE_HOME/network/admin`.

Task 3: Start Oracle Enterprise Security Manager

To start Oracle Enterprise Security Manager, enter the following at the command line:

```
oemapp esm
```

If the `ldap.ora` file is not configured, you receive the following alert:



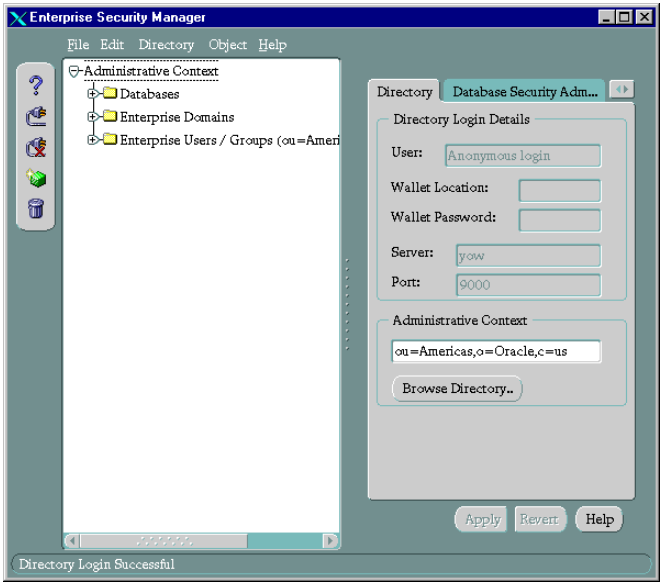
If this happens, exit Oracle Enterprise Security Manager and run Net8 Configuration Assistant to set up directory access, then restart Oracle Enterprise Security Manager. Alternatively, you can:

- Manually connect to the directory server as described in Task 4 on page 20-5.
- 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 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, it attempts to connect by using anonymous authentication.

On startup, Oracle Enterprise Security Manager displays the following window:



If the result of automatic login is not acceptable, log out and log back in again with a specific user name:

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 for instructions about filling in the fields in this dialog box.

Task 4: Log Into the Directory

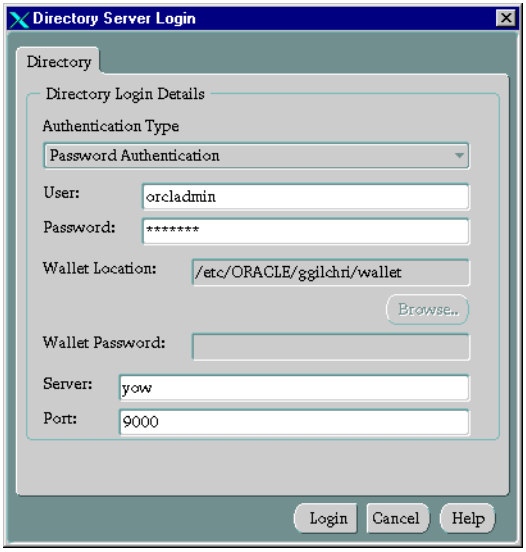
To log into the directory:

1. From the Oracle Enterprise Security Manager menu bar, choose `Directory > Login`.
2. Select the authentication type from Table 20–1:

Table 20–1 Authentication Types

Authentication Type	Description
Password Authentication	Uses simple authentication requiring a user **distinguished name (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.

The Directory Server Login window appears:



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, you must enter the directory's SSL port number.
5. Choose OK.

Navigating Oracle Enterprise Security Manager

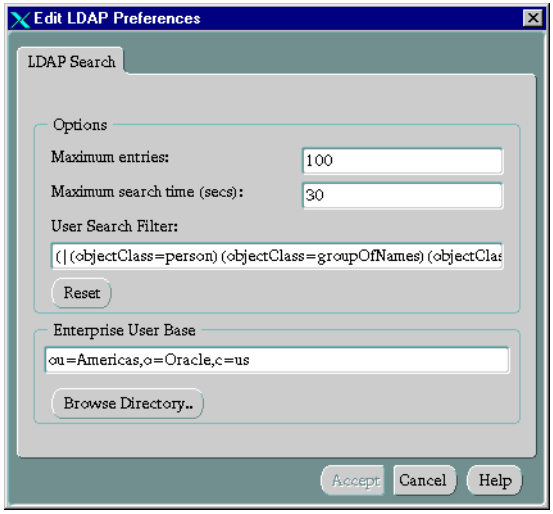
This section describes some basic features of Oracle Enterprise Security Manager, in the following sections:

- ❏ 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. To use a search base other than the configured administrative context, do the following:

1. On the menu bar, choose `Edit > Preferences`; the Edit LDAP Preferences window appears:



2. In the Enterprise Users Base field, enter a **distinguished name (DN)** as the base of the search.

   You can also choose `Browse Directory` to navigate to a directory object to use as the base of the search.

3. Choose `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:

To change the administrative context to `c=acme,c=us`:

1. Navigate to the Oracle Enterprise Security Manager initial screen (Task 3), and choose `Browse Directory`; the corresponding dialog box appears.
2. In the Naming Context field, enter `c=us`.
3. In the Directory Search Attributes field, in the Searchable Attribute Value column, in the object class row, enter `organization`. The entries for organizations in the U.S. appear in the Directory Search Results: Directory Entry field.
4. In the Directory Search Results: Directory Entry field, select a directory entry to use as the new administrative context, and choose `OK`. This returns you to the Oracle Enterprise Security Manager initial screen. The administrative context you specified appears in the Administrative Context field.

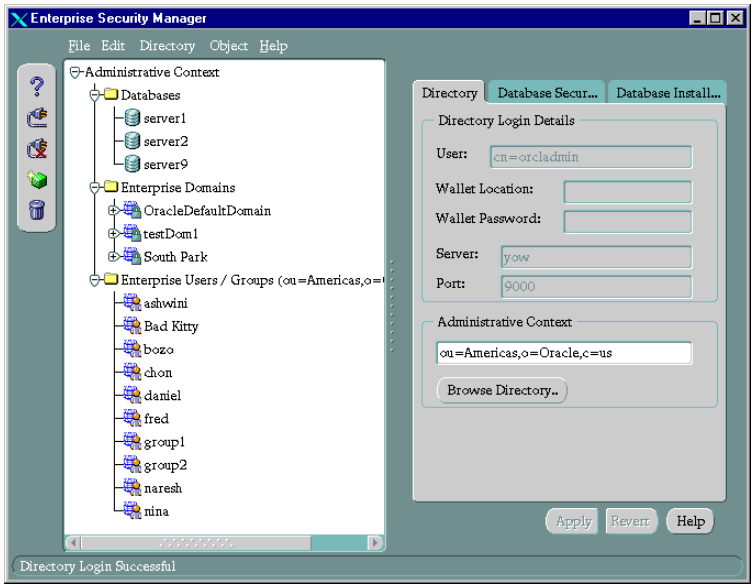
Use the same steps when browsing for directory objects in other contexts (for example, when using the Edit LDAP Preferences dialog box to change the base of a search).

Administering Enterprise Databases, Domains, and Users

The following instructions assume you are running Oracle Enterprise Manager and have invoked the Oracle Enterprise Security Manager.

Managing enterprise users involves working in the three top level nodes in the Oracle Enterprise Security Manager navigator window. These three nodes are discussed in the following sections:

- Administering Databases
- Administering Enterprise Domains
- Administering Enterprise Users



Administering Databases

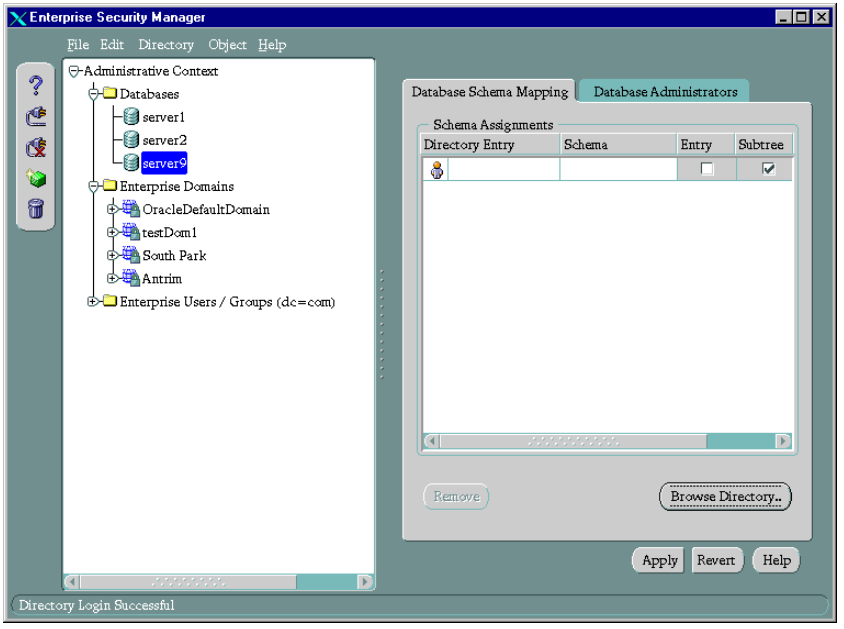
This section describes how to manage user/schema separation for a database.

See Also:

- Shared Schemas on page 17-17, for a conceptual overview of this feature
- Managing Database Administrators on page 20-31

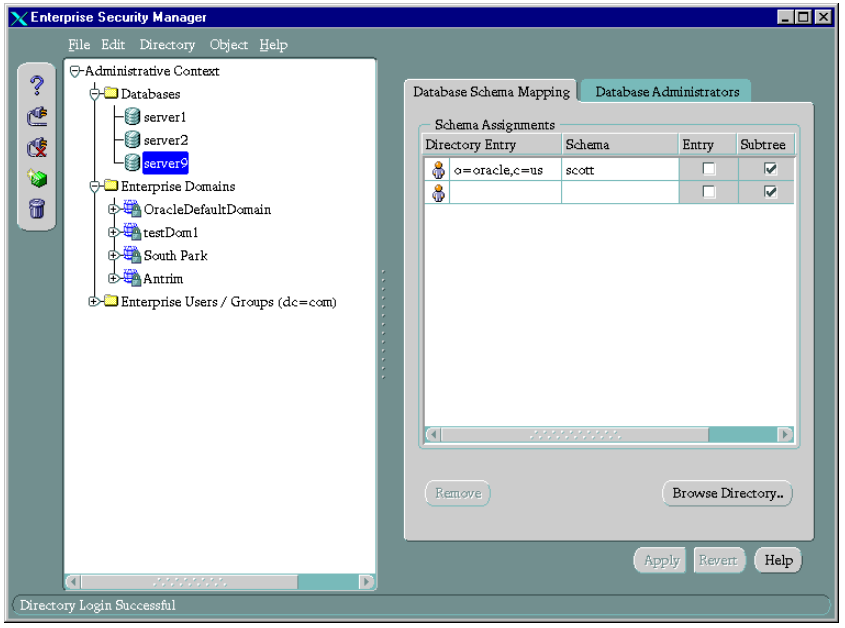
To map an enterprise user to a database schema:

1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Context > Database.`
2. Select a server to support user/schema separation; the corresponding tab pages appear in the right pane of the window.



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 an entry to map to a shared schema. You can also choose 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 for that database.
5. If this is a full DN, choose the check box in the Entry column; if this is a partial DN, choose the check box in the Subtree column.



6. Choose Apply; the database object is updated in the directory, and an empty row is added in the Schema Assignments window. This lets you add future additional 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. The following sections describe how to administer enterprise domains:

- 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

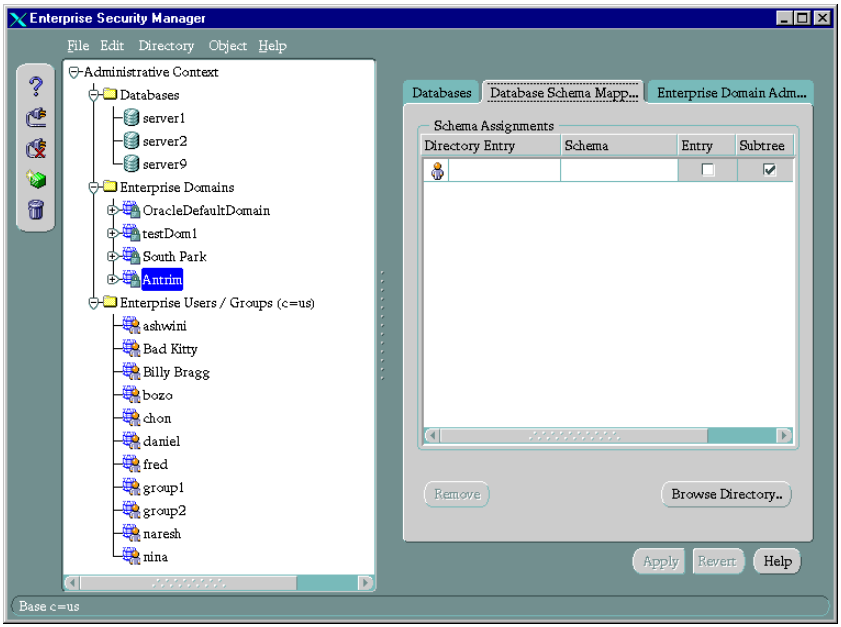
Managing User/Schema Separation

Administering Databases on page 20-10 discussed how to manage user/schema separation for *an individual database*. This section describes how to manage user/schema separation for *all the databases in a given domain*.

To map an enterprise user to a database schema:

1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Context > Enterprise Domains`.

2. Select an enterprise domain to support user/schema separation; the corresponding tab pages appear in the right pane of the window.

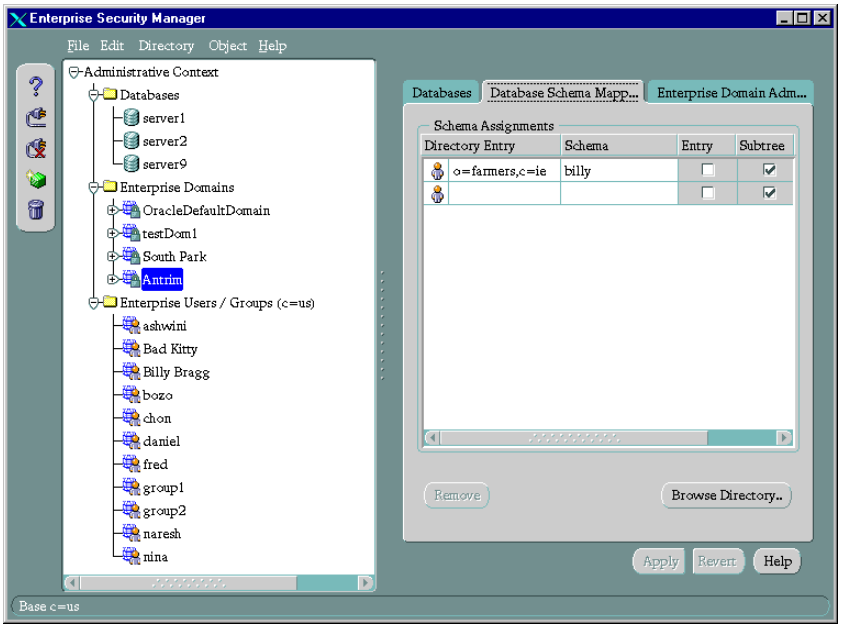


3. Choose the Database Schema Mapping tab.

4. In the Schema Assignments window, in the Directory Entry column, enter either a full or partial DN to map to a shared schema. You can also choose 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, choose the check box in the Entry column; if this is a partial DN, choose the check box in the Subtree column.



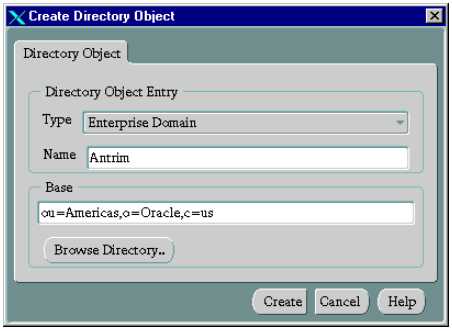
7. Choose `Apply`; the database object is updated in the directory, and an empty row is added in the Schema Assignments window. This lets you add future additional mappings.

Creating an Enterprise Domain

An enterprise domain contains databases and **enterprise role**s. You can create a new enterprise domain by naming it, and defining where it is to be located in the directory.

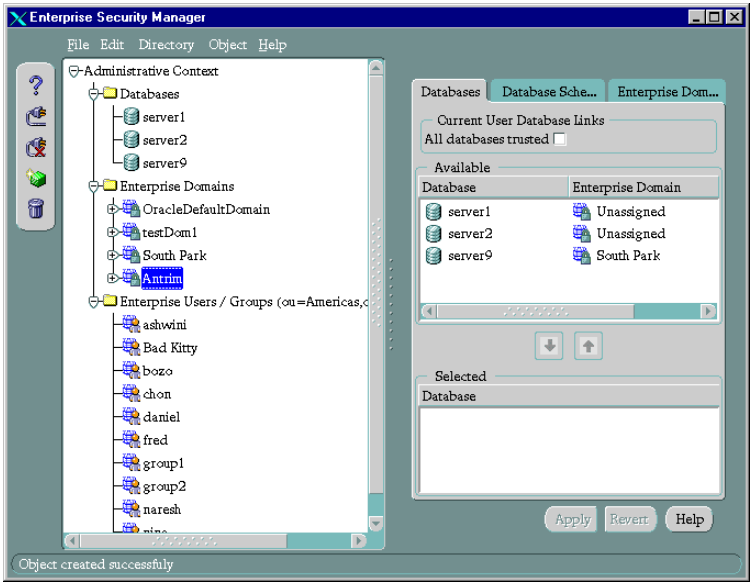
To create an enterprise domain:

1. Choose `Object > Create` on the menu bar; the Create Directory Object window appears:



2. In the Type menu, choose Enterprise Domain.
3. In the Name field, enter the name of the new enterprise domain.
4. In the Base field, Oracle Enterprise Security Manager displays the name of the administrative context. To use a different administrative context, you can change the values in this field. However, be careful to enter the name of a valid administrative context—one that contains and Oracle Context.
5. Choose `Create`.The new enterprise domain appears at the bottom of the Enterprise Domains node.

6. In the navigator pane of the Enterprise Security Manager window, select the name of the new enterprise domain you created; the corresponding group of tab pages appear in the right pane of the window.



7. You can optionally choose the All Databases trusted check box; this lets databases within the enterprise domain have current user database links between them.

Note: Individual **Database Administrator**s 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 it.

Adding a Database to an Enterprise Domain

Upon 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 an enterprise domain:

1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Context > Enterprise Domains`.
2. In the navigator pane of the window, select an enterprise domain to add a database to.
3. In the right pane of the window, in the Available region, select a database name.
4. Choose the down arrow to move the selected database to the Selected list.
5. Choose `Apply`.

See Also: Step 4: Use Oracle Database Configuration Assistant to Register the Database in the Directory on page 17-35.

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 it.

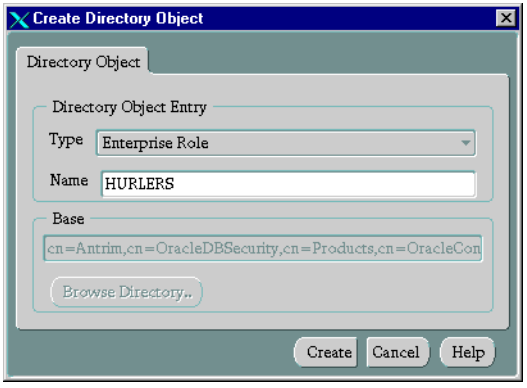
An enterprise role is a set of **global role**s that operate on multiple databases within an enterprise domain. An enterprise role is assigned to one or more enterprise users. The Enterprise Database Administrator uses these enterprise roles to assign sets of global roles on multiple databases to a selected user.

You cannot create two enterprise roles with the same name within a single enterprise domain. However, you can 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 relationship.

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:

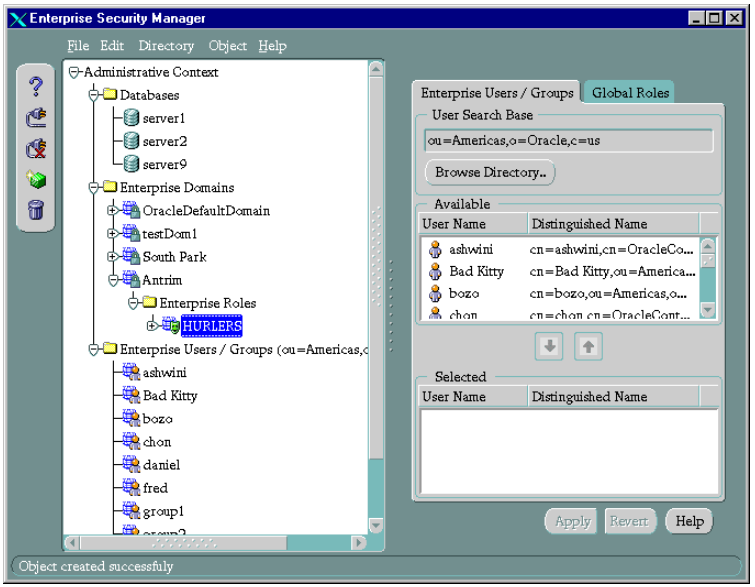
1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Context > Enterprise Domains` and select the enterprise domain name; the corresponding group of tab pages appear in the right pane of the window.
2. On the menu bar, choose `Object > Create`; the Create Directory Objects dialog window appears:



3. From the Type menu, select Enterprise Role.
4. In the Name field, enter the name of the new enterprise role.

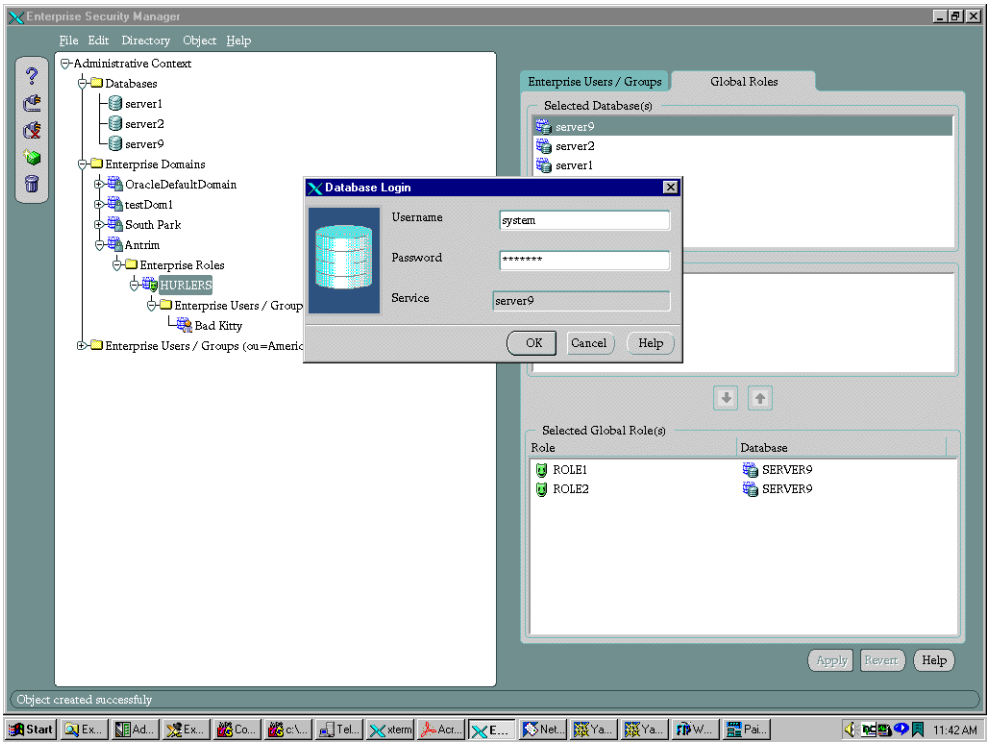
   Note that the directory base chosen for the new enterprise role derives from the currently selected enterprise domain; you cannot edit this value.
5. Choose `Create`.
6. In the navigator pane of the window, expand `Enterprise Domains > enterprise_domain_name > Enterprise Roles.`

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 of the window.

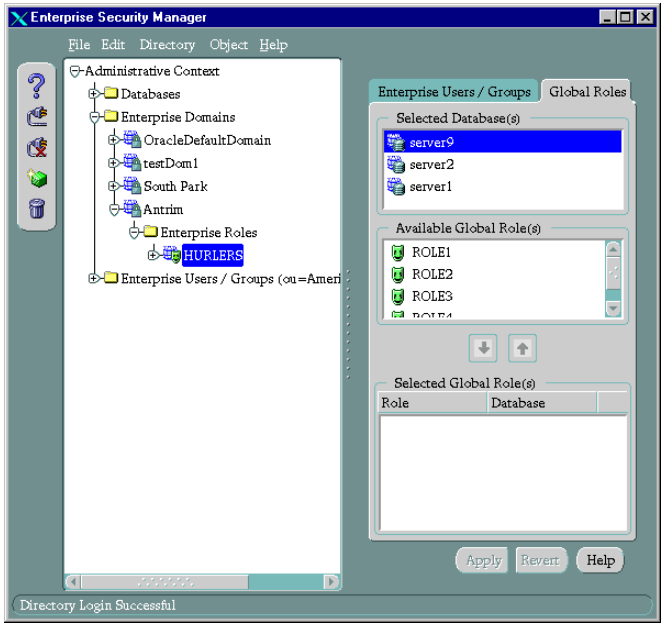


8. Choose the Global Roles tab.

9. Select a database; the Database Login window appears:



10. Supply the correct information about the selected database; choose OK. The selected database roles appear in the Available Global Role(s) region of the window:



If no database service has been configured:

- In the Global Roles tab page, right-click the database name in the Selected Databases field.
- Choose Reconnect.
- 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 Global Role(s) field, select an available role.

12. Click the down arrow to move the role into the Selected Global Role(s) region of the window.
13. Repeat steps 9 through 12 for each database to select roles from.
14. Choose Apply.

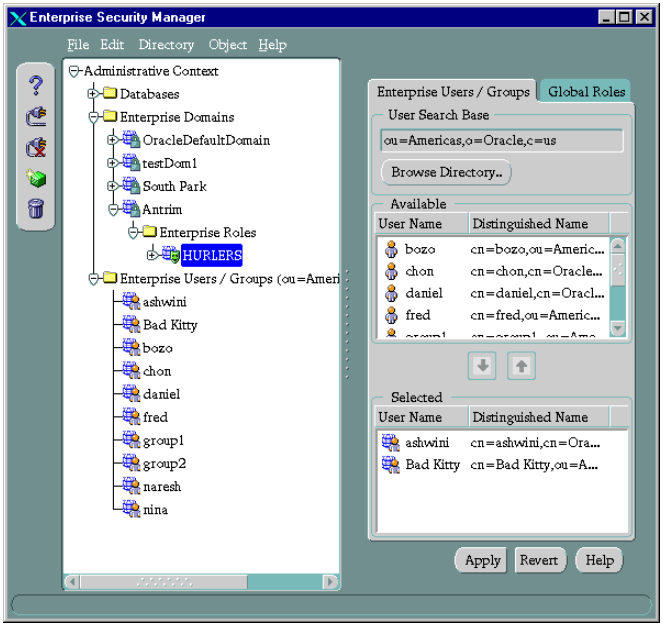
You have created an enterprise role within an enterprise domain of databases, and can assign this enterprise role to any enterprise user.

Assigning Enterprise Users to an Enterprise Role

To assign an enterprise user to an enterprise role:

1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Context > Enterprise Domains > *enterprise_domain_name* > Enterprise Roles`.
2. In the navigator pane of the window, select the enterprise role; the corresponding group of tab pages appears in the right pane.
3. In the right pane of the window, select the Enterprise / Users Groups tab.
4. In the Available region, select enterprise users to assign to the role.

5. Choose the down arrow; the enterprise users appear in the Selected window.



6. Choose Apply; the enterprise users appear under the Enterprise Role node in the navigator pane of the window.

Removing a Database from an Enterprise Domain

1. In the navigator pane of the Enterprise Security Manager window, expand Administrative Context > Enterprise Domains.
2. In the navigator pane of the window, select the name of an enterprise domain to remove a database from.
3. In the Databases tab page, in the Selected window, select a database to remove from the enterprise domain.
4. Choose the up button to move the database from the Selected window to the Available window.
5. Choose Apply.

Deleting an Enterprise Domain

To delete an enterprise domain, you must first delete all of its enterprise roles. Otherwise, an error message appears.

To delete an Enterprise Domain:

1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Contexts > Enterprise Domains.`
2. In the navigator pane of the window, select the name of an enterprise domain to delete.
3. Choose the `Delete Object` button to the left of the navigator pane; a window asks you to confirm the deletion.
4. Choose `Yes`; the selected enterprise domain is removed from the enterprise domains tree.

Administering Enterprise Users

This section describes:

- 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 lets you create new enterprise users if the users do not already exist in the directory server:

1. From the menu bar, choose `Object > Create`; the Create Directory Object window appears.
2. From the Type menu, choose Enterprise User.
3. In the Name field, enter the name of the new enterprise user.
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. Choose `Create`. The enterprise user you created appears in the navigator pane of the window, under the Enterprise Users / Groups node. When you select the new enterprise user, the corresponding tab page appears in the right pane of the window.

Note: In the preceding procedure, the directory user entry that Oracle Enterprise Security Manager creates is associated with only the `top` and `person` object classes. To associate that user entry with other object classes, you must do so in a 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.

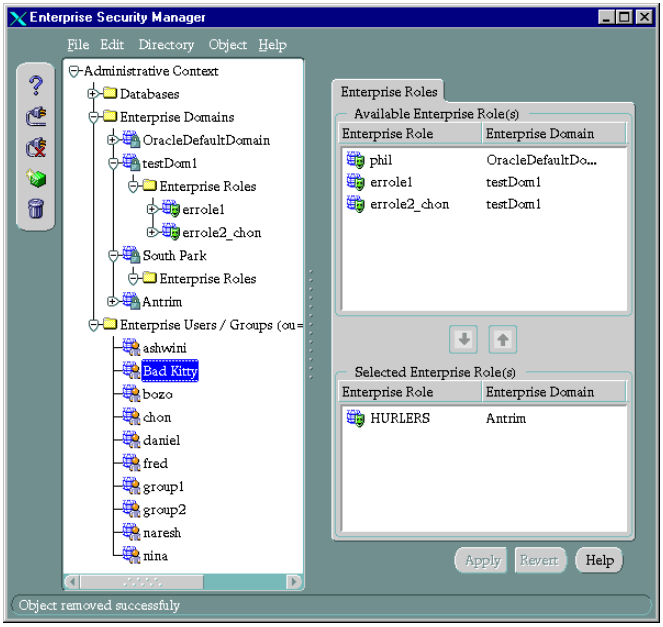
You can grant multiple enterprise roles to enterprise users, and these roles can exist in different enterprise domains. You can grant these roles in two ways:

- 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 access to a global user, it searches the directory for the 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 of the Enterprise Security Manager window, expand `Administrative Context > Enterprise Users.`

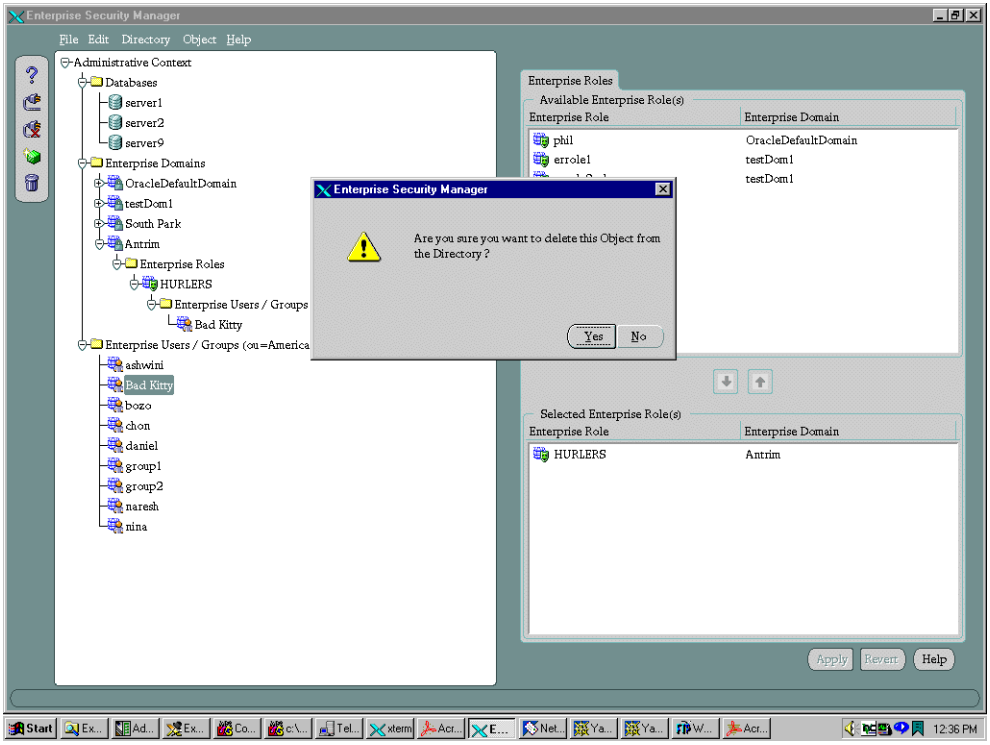


2. Select the name of an enterprise user; the corresponding group of tab pages appears in the right pane of the window.
3. In the Available Enterprise Role(s) window, select an enterprise role to grant to the enterprise user.
4. Choose the down arrow; the selected role is moved from the Available Role(s) list to the Selected Role(s) list.
5. Choose `Apply`.

Deleting an Enterprise User

You can delete an enterprise user only if that user has no enterprise roles. To delete an enterprise user:

1. Expand `Administrative Context > Enterprise Users/Groups`.



2. Select an enterprise user to delete.
3. On the menu bar, choose `Object > Delete`; an alert asks you to confirm the deletion.
4. Choose `Yes`; the enterprise user is deleted from the tree in the navigator pane of the window.

Managing Security Administrators

Use Oracle Enterprise Security Manager to define administrators, as described in the following sections:

- Managing Database Security Administrators
- Managing Database Installation Administrators
- Managing Database Administrators
- Managing Enterprise Domain Administrators

See Also: Oracle Context on page 17-9.

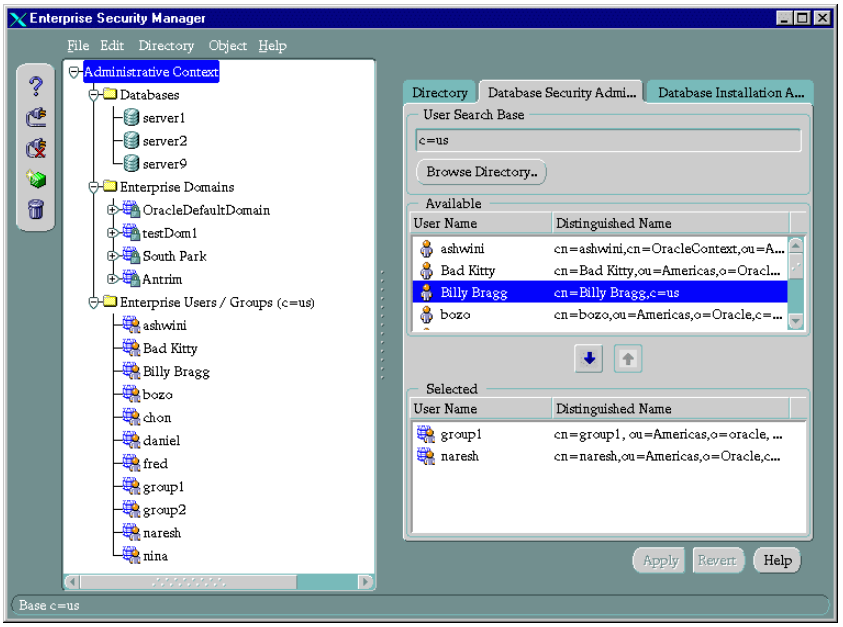
Managing Database Security Administrators

To manage **Database Security Administrator**s, you must be a member of the Database Security Administrators group.

To define a user as a Database Security Administrator:

1. In the navigator pane of the Enterprise Security Manager window, select Administrative Contexts.

2. In the right pane of the window, select the Database Security Administrators tab; enterprise user names appear in the Available field.



3. Select an enterprise user to define as an administrator.
4. Choose the down arrow to move the user to the Selected window.
5. Choose Apply.

Managing Database Installation Administrators

To manage **Database Installation Administrator**s, you must be a member of the Database Security Administrators group.

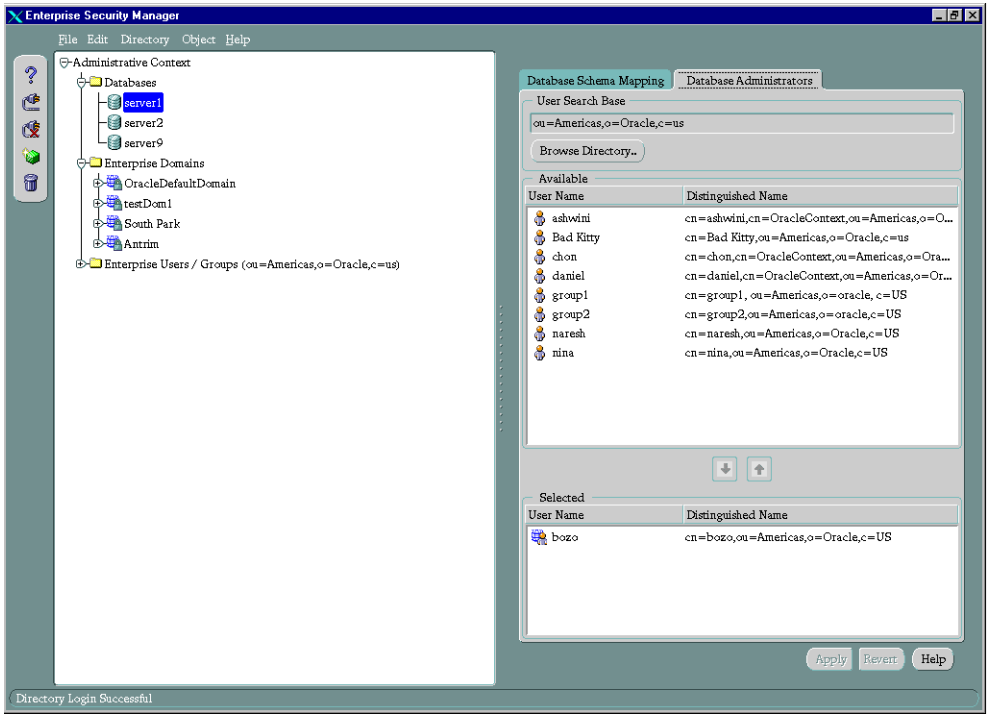
To define a user as a Database Installation Administrator:

1. In the navigator pane of the Enterprise Security Manager window, select Administrative Contexts.
2. In the right pane of the window, select the Database Installation Administrators tab; enterprise user names appear in the Available field.
3. Select an enterprise user to define as an administrator.
4. Choose the down arrow to move the user to the Selected window.
5. Choose `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 of the Enterprise Security Manager window, expand `Administrative Context > Database`.

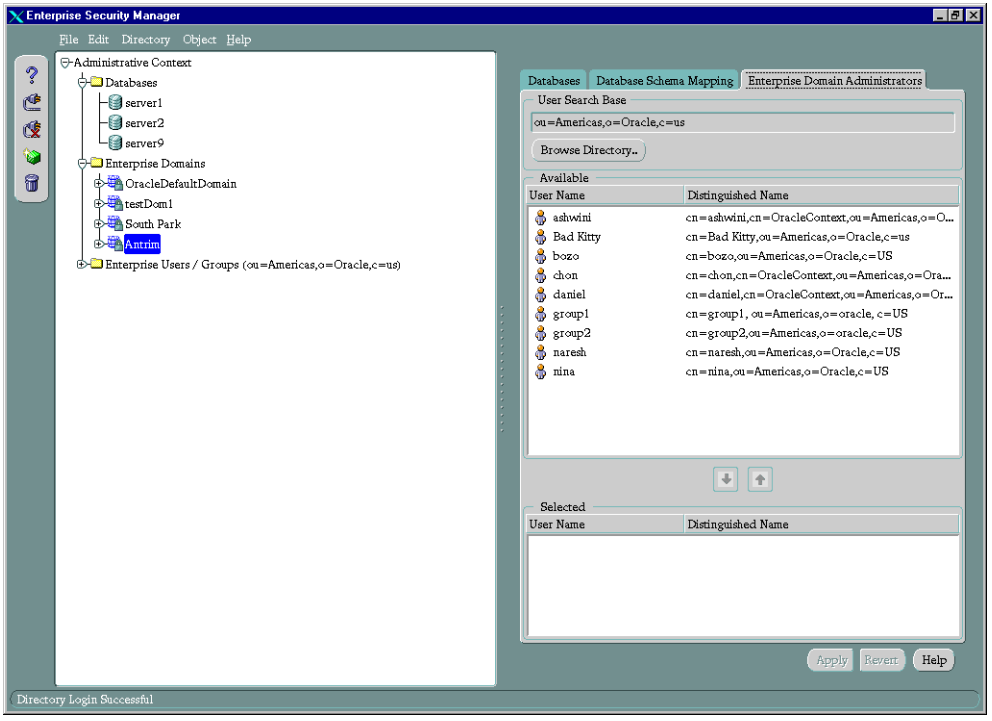


2. Select a database to assign administrators to; the corresponding group of tab pages appears in the right pane of the window.
3. Select the Database Administrators tab; the Available window displays user names of enterprise users available in the current user search base.
4. Select an enterprise user to define as an administrator.
5. Choose the down arrow to move the user to the Selected window.
6. Choose `Apply`.

Managing Enterprise Domain Administrators

To manage **Enterprise Domain Administrator**s, you must be either a member of the Database Security Administrators group or a domain administrator for this particular domain.

1. In the navigator pane of the Enterprise Security Manager window, expand `Administrative Context > Enterprise Domains`.



2. Select an enterprise domain to assign administrators to; the corresponding group of tab pages appears in the right pane of the window.
3. Select the Enterprise Domain Administrators tab; the Available window displays user names of enterprise users available in the current user search base.
4. Select an enterprise user to specify as an administrator.
5. Choose the down arrow to move the user to the Selected window.
6. Choose `Apply`.

Part VI

Appendices

This part contains the following appendices for your reference:

- Appendix A, Data Encryption and Integrity Parameters
- Appendix B, Authentication Parameters
- Appendix C, Integrating Authentication Devices Using RADIUS
- Appendix D, Oracle Advanced Security FIPS 140-1 Settings
- Appendix E, LDAP Directory Schema for Oracle Database Security
- Appendix F, Oracle Implementation of Java SSL

A

Data Encryption and Integrity Parameters

This appendix describes **encryption** and data **integrity** parameters supported by Oracle Advanced Security. It also includes an example of a `sqlnet.ora` file generated by performing the network configuration described in Chapter 2, Configuring Data Encryption and Integrity, and Chapter 9, Configuring Secure Socket Layer Authentication.

This appendix contains the following sections:

- Sample sqlnet.ora File
- Data Encryption and Integrity Parameters

Sample sqlnet.ora File

This section contains a sample `sqlnet.ora` configuration file for a set of clients with similar characteristics and a set of servers with similar characteristics. The file includes examples of Oracle Advanced Security encryption and data integrity parameters.

Trace File Setup

```
#Trace file setup
trace_level_server=16
trace_level_client=16
trace_directory_server=/orant/network/trace
trace_directory_client=/orant/network/trace
trace_file_client=cli
trace_file_server=srv
trace_unique_client=true
```

Oracle Advanced Security Encryption

```
#ASO Encryption
sqlnet.encryption_server=accepted
sqlnet.encryption_client=requested
sqlnet.encryption_types_server=(RC4_40)
sqlnet.encryption_types_client=(RC4_40)
sqlnet.crypto_seed = "-kdje83kkep39487dvmlqEPTbxxe70273"
```

Oracle Advanced Security Integrity

```
#ASO Checksum
sqlnet.crypto_checksum_server=requested
sqlnet.crypto_checksum_client=requested
sqlnet.crypto_checksum_types_server = (MD5)
sqlnet.crypto_checksum_types_client = (MD5)
```

SSL

```
#SSL
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 )
```

Data Encryption and Integrity Parameters

If you do not specify any values for Server Encryption, Client Encryption, Server Checksum, or Client Checksum, the corresponding configuration parameters do not appear in the `sqlnet.ora` file. However, Oracle Advanced Security defaults to ACCEPTED.

For both data encryption and integrity algorithms, the server selects the first algorithm listed in its `sqlnet.ora` file that matches an algorithm listed in the client `sqlnet.ora` file or in the client installed list. If there are no entries in the server sqlnet.ora file, the server sequentially searches its installed list, to match an item on the client side—either in the client `sqlnet.ora file` or in the client installed list. *If no match can be made, the connection fails.*

Data encryption and integrity algorithms are selected independently of each other; encryption can be activated without integrity, and integrity can be activated without encryption, as shown by Table A–1:

Table A–1 Algorithm Selection

Encryption Selected?	Integrity Selected?
Yes	No
Yes	Yes
No	Yes
No	No

There are three classes of parameters required to enable data encryption and integrity:

- Encryption and Integrity Level Settings:
- Encryption and Integrity Selected Lists
- Seeding the Random Key Generator

See Also:

- Chapter 2, Configuring Data Encryption and Integrity
- Activating Encryption and Integrity on page 2-6

Encryption and Integrity Level Settings:

Table A–2 summarizes data encryption and integrity level settings:

Table A–2 Encryption and Integrity Level Settings

Algorithm Type	Platform	Item	Description
Encryption	Server	Purpose	This parameter specifies the desired 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*
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	ACCEPTED
	Client	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*
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	ACCEPTED

Table A–2 Encryption and Integrity Level Settings

Algorithm Type	Platform	Item	Description
Integrity	Server	Purpose	This parameter specifies the desired data integrity behavior when a client (or another server acting as a client) 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
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	ACCEPTED
	Client	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*
		Values	ACCEPTED, REJECTED, REQUESTED, REQUIRED
		Default	ACCEPTED

Encryption and Integrity Selected Lists

Table A–3 Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description
Encryption	Server	Purpose	This parameter specifies a list of encryption algorithms used by the server, in the order of intended use. Enter the most desired algorithm first. This list is used to negotiate a mutually acceptable algorithm with the client end of the connection. Each algorithm is checked against the list of 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*])
		Values	■ RC4_40: RSA RC4 (40-bit key size). ■ RC4_56: RSA RC4 (56-bit key size). ■ RC4_128: RSA RC4 (128-bit key size). ■ RC4_256: RSA RC4 (256-bit key size). ■ 3DES112: Triple-DES (112-bit key size). ■ 3DES168: Triple-DES (168-bit key size). ■ DES40: DES40 (40-bit key size). ■ DES: Standard DES (56-bit key size).
		Default	All installed algorithms are used in a negotiation if no algorithms are defined in the local `sqlnet.ora` file.

Table A–3 Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description
Encryption	Server	Usage Notes	All algorithms are installed: RC4_40, RC4_56, RC4_128, RC4_256, 3DES112, 3DES168, DES40, and DES. If no algorithms are specified (in the local `sqlnet.ora file`), the installed algorithms are used in that order to negotiate a mutually acceptable algorithm with the other end of the connection. You can specify multiple encryption algorithms—either a single value or a list of algorithm names. For example, either of the following encryption parameters is acceptable: SQLNET.ENCRYPTION_TYPES_SERVER=(RC4_40) SQLNET.ENCRYPTION_TYPES_SERVER=(DES,RC4_56,RC4_128,DES40)
	Client	Purpose	This parameter specifies a list of encryption algorithms the client (or a server acting as a client) uses 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]`*`)`
		Values	■ RC4_40: RSA RC4 (40-bit key size). ■ RC4_56: RSA RC4 (56-bit key size). ■ RC4_128: RSA RC4 (128-bit key size). ■ RC4_256: RSA RC4 (256-bit key size). ■ 3DES112: Triple-DES (112-bit key size). ■ 3DES168: Triple-DES (168-bit key size). ■ DES40: DES40 (40-bit key size). ■ DES: Standard DES (56-bit key size).
		Default	All installed algorithms are used in a negotiation if no algorithms are defined in the `sqlnet.ora` file.

Table A–3 Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description

Encryption	Client	Usage Notes	**A**ll algorithms are installed: RC4_40, RC4_56, RC4_128, RC4_256, 3DES112, 3DES168, DES40, and DES. 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.
			
			You can specify multiple encryption algorithms—either a single value or a list of algorithm names. For example, either of the following encryption parameters is acceptable:
			
			`SQLNET.ENCRYPTION_TYPES_CLIENT=(`*`DES,DES40,RC4_56,RC4_40`*`)`
			`SQLNET.ENCRYPTION_TYPES_CLIENT=(`*`RC4_40`*`)`

Integrity	Server	Purpose	This parameter specifies a list of the data integrity algorithms the server is allowed to use, in order of intended use, 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 of the connection. Each algorithm is checked against the list of client algorithm types available until a match is found. The first 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`*`)`
		Values	Message Digest 5 (MD5); Secure Hash Algorithm (SHA-1).
		Default	All installed algorithms are used in a negotiation if no algorithms are defined in the local `sqlnet.ora` file.

Table A–3 Data Encryption and Integrity Selected Lists

Algorithm Type	Platform	Item	Description
Integrity	Client	Purpose	This parameter specifies a list of data integrity algorithms the client (or a 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 of the connection. 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*)
		Values	Message Digest 5 (MD5); Secure Hash Algorithm (SHA-1).
		Default	All installed algorithms are used in a negotiation if no algorithms are defined in the local `sqlnet.ora` file.

Seeding the Random Key Generator

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

The characters that form the value for this parameter are used when generating cryptographic keys. The more random the characters entered into this field are, the stronger the keys are. You set this parameter by entering from 10 to 70 random characters into the above statement.

> **Note:** Oracle Corporation recommends that you enter as many characters as possible, up to 70, to make the resulting key more random and therefore stronger.

This parameter must be present in the `sqlnet.ora` file whenever data encryption or integrity is enabled.

B

Authentication Parameters

This appendix demonstrates some sample configuration files with the necessary profile file (`sqlnet.ora`) and database initialization file (`init.ora`) authentication parameters, when using the CyberSafe, Identix, Kerberos, SecurID, RADIUS, or SSL authentication. It contains 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 (`init.ora`)	`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 Biometric Parameters
- Initialization File 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 parameter 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 Biometric Parameters

Following are two sets of parameters: the **Oracle database method** and the **file system method**. The minimum sets of parameters required for each method are listed below:

Oracle Database Method

```
sqlnet.authentication_services = (beq, identix)
sqlnet.identix_fingerprint_method = oracle
sqlnet.identix_database_directory = <identix_scanner>
sqlnet.identix_fingerprint_database_user = <usrname>
sqlnet.identix_fingerprint_database_password = <pwd>
```

File System Method

```
sqlnet.authentication_services = (beq, identix)
sqlnet.identix_fingerprint_method = file
sqlnet.identix_database_directory = /etc/ofm_storage
```

Initialization File Parameters

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

Parameters for Clients and Servers using Kerberos Authentication

Following is a list of parameters to insert into the configuration files for clients and servers using Kerberos.

Table B–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_CONF_MIT=(FALSE) SQLNET.KERBEROS5_REALMS=/krb5/krb.realms SQLNET.KERBEROS5_KEYTAB=/krb5/v5srvtab
initialization parameter file (init.ora)	REMOTE_OS_AUTHENT=FALSE OS_AUTHENT_PREFIX=""

Parameters for Clients and Servers using 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 (init.ora)	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 (init.ora) 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.

Table B–15 SQLNET.RADIUS_AUTHENTICATION_TIMEOUT

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 (oracle / net / radius / 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, and to set the path to the JDK Libjava.
Default	$ORACLE_HOME / jlib / netradius.jar:$ORACLE_ HOME / JRE / lib / sparc / native_threads

Recommended Minimum Sets of RADIUS Parameters

Following are two set of sample `sqlnet.ora` file RADIUS authentication parameters:

- Static User Name and Password
- 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
```

Note: If you are using the default value, confirm that the following file exists:

```
$ORACLE_HOME/network/security/radius.key
```

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
```

Initialization File (init.ora) Parameters

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

Parameters for Clients and Servers using SSL

There are two ways to configure a parameter:

- Static: The name of the parameter that exists in the `sqlnet.ora` file.
- Dynamic: The name of the parameter used in the security subsection of the 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

Table B–30 SSL Client Authentication Parameters

Description:	To control whether a client, in addition to the server, is authenticated using SSL.
Existing / New Parameter	New
Syntax (static):	SSL_CLIENT_AUTHENTICATION={TRUE | FALSE}
Example (static):	SSL_CLIENT_AUTHENTICATION=FALSE
Syntax (dynamic):	SSL_CLIENT_AUTHENTICATION={TRUE | FALSE}
Example (dynamic):	SSL_CLIENT_AUTHENTICATION=FALSE

Wallet Location

For any application that must access a wallet for loading the security credentials into the process space, you must specify the wallet location parameters defined by Table B–31 in each of the following configuration files:

- sqlnet.ora
- listener.ora

Table B–31 Wallet Location Parameters

Static Configuration	Dynamic Configuration
oss.source.my_wallet = (SOURCE= (METHOD=File) (METHOD_DATA= (DIRECTORY=your wallet location)))	MY_WALLET_DIRECTORY = your_wallet_dir

The default wallet location is the $ORACLE_HOME directory.

C

Integrating Authentication Devices Using RADIUS

This appendix describes how third party authentication vendors customize the RADIUS challenge-response user interface to fit their particular device.

This appendix contains the following sections:

- About the RADIUS Challenge-Response User Interface
- Customizing the RADIUS Challenge-Response User Interface

See Also: Chapter 4, Configuring RADIUS Authentication

About the RADIUS Challenge-Response User Interface

You can set up any authentication device that supports the RADIUS standard to authenticate Oracle users. When your authentication device uses the challenge-response mode, a graphical interface prompts the user first for a password, then for additional information—for example, a dynamic password that the user obtains from a token card. This interface is Java-based to provide optimal platform independence.

Third party vendors of authentication devices must customize this graphical user interface to fit their particular device. For example, a smart card vendor customizes the Oracle client to issue the challenge to the smart card reader. Then, when the smart card receives a challenge, it responds by prompting the user for more information, such as a PIN.

Oracle has developed a Java class for this graphic user interface. It is a set of Java code methods that implement an Oracle RADIUS interface (described below). These methods are loaded dynamically by a C-code module using the Java Native Interface, specified in release 1.1 of the Java Development Kit from JavaSoft. You can find the file `OracleRadiusInterface` file 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 8.1.7 has been validated under U.S. Federal Information Processing Standard (FIPS) 140-1 at the Level 2 security level. This appendix describes the formal configuration required for Oracle Advanced Security to comply with the FIPS 140-1 standard. Refer to the NIST Cryptographic Modules Validation list at the following web site address:

`http://csrc.nist.gov/cryptval/140-1/1401val.htm`

This appendix contains the following sections:

- 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 contains the following sections:

- 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.

This appendix contains the following sections:

- Oracle Java SSL Features
- SSL Cipher Suite Supported in Oracle Java SSL
- Certificate and Key Management with Oracle Wallet Manager
- Oracle Java SSL Examples
- Security Aware Applications Support
- 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 Chapter 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();
    }


}
```

Security Aware Applications Support

To enable the security aware applications to do their own validation, Oracle Java SSL code allows handshakes to pass even if **trust point**s are not set for RSA SSL ciphers.

A sample security aware application will not set trust points. It can get the peer **certificate chain** after the handshake using the following code:

```
javax.security.cert.x509Certificate[] peerCerts
          = jsslSoc.getSession() .getPeerCertificateChain();
```

where jsslSoc is the SSLSocket object used for the connection. Using the certificate chain the individual certificates can be extracted for application specific validation like matching the certificate's **distinguished name (DN)** against a user database. This is useful when there are large numbers of trust points stored in a database and the application does not want to pass all of them to the SSL layer. The application can extract the relevant trust points and match them against certificates in the peer certificate chain. However, the application must match the certificates in the chain against their trust points to verify whether the chain can be trusted or not.

Security unaware applications that always want the trust point check should ensure that trust points are set in the application itself.

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 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. Creates a server socket which uses all network interfaces on the host, and is bound to the specified port.
OracleSSLServerSocket(int, int)	Creates 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)	Creates 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. Creates 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)`
Creates 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 aspects 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 input stream inStream.
equals(java.lang.Object obj)	Checks for the equality
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 valid
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 parameters
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 equality

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 parameters

Returns:
signature algorithm parameters

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 valid

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

The ability of a system to grant or limit access to specific data for specific clients or groups of clients.

Access Control List (ACL)

The group of access directives that you define. The directives grant levels of access to specific data for specific clients and/or groups of clients.

administrative context

A directory entry under which an **Oracle Context** resides. An administrative context can be a **directory naming context**. During directory access configuration, clients are configured with an administrative context in the directory configuration file (ldap.ora). The administrative context specifies the location of the Oracle Context in the directory whose entries a client expects to access

authentication

The process of verifying the identity of a user, device, or other entity in a computer system, often as a prerequisite to granting access to resources in a system. A recipient of an authenticated message can be certain of the message's origin (its sender). Authentication is presumed to preclude the possibility that another party has impersonated the sender.

authorization

Permission given to a user, program, or process to access an object or set of objects. In Oracle, authorization is done through the role mechanism. A single person or a group of people can be granted a role or a group of roles. A role, in turn, can be granted other roles. The set of priveleges available to an authenticated entity.

CDS

See **Cell Directory Services (CDS)**.

Cell Directory Services (CDS)

An external naming method that enables users to use Oracle tools transparently and applications to access Oracle8*i* databases in a Distributed Computing Environment (DCE) environment.

certificate

An ITU x.509 v3 standard data structure that securely binds an identify to a public key.

A certificate is created when an entity's public key is signed by a trusted identity, a certificate authority. The certificate ensures that the entity's information is correct and that the public key actually belongs to that entity.

A certificate contains the entity's name, identifying information, and public key. It is also likely to contain a serial number, expiration date, and information about the rights, uses, and privileges associated with the certificate. Finally, it contains information about the certificate authority that issued it.

certificate authority

A trusted third party that certifies that other entities—users, databases, administrators, clients, servers—are who they say they are. When it certifies a user, the certificate authority first seeks verification that the user is not on the certificate revocation list (CRL), then verifies the user's identity and grants a certificate, signing it with the certificate authority's private key. The certificate authority has its own certificate and public key which it publishes. Servers and clients use these to verify signatures the certificate authority has made. A certificate authority might be an external company that offers certificate services, or an internal organization such as a corporate MIS department.

certificate chain

An ordered list of certificates containing an end-user or subscriber certificate and its certificate authority certificates.

checksumming

A mechanism that computes a value for a message packet, based on the data it contains, and passes it along with the data to authenticate that the data has not been tampered with. The recipient of the data recomputes the cryptographic checksum and compares it with the cryptographic checksum passed with the data; if they

match, it is "probabilistic" proof the data was not tampered with during transmission.

Cipher Block Chaining (CBC)

An encryption method that protects against block replay attacks by making the encryption of a cipher block dependent on all blocks that precede it; it is designed to make unauthorized decryption incrementally more difficult. Oracle Advanced Security employs *outer* cipher block chaining because it is more secure than *inner* cipher block chaining, with no material performance penalty.

cipher suite

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

ciphertext

Message text that has been encrypted.

client

A client relies on a service. A client can sometimes be a user, sometimes a process acting on behalf of the user during a database link (sometimes called a proxy).

confidentiality

A function of cryptography. Confidentiality guarantees that only the intended recipient(s) of a message can view the message (decrypt the ciphertext).

CORBA

Common Object Request Broker Architecture. An architecture that enables pieces of programs, called objects, to communicate with one another regardless of the programming language in which they are written or the operating system on which they are running. CORBA was developed by an industry consortium known as the Object Management Group (OMG).

cryptography

The practice of encoding and decoding data, resulting in secure messages.

Database Administrator

(1) A person responsible for operating and maintaining an Oracle Server or a database application. (2) An Oracle username that has been given DBA privileges

and can perform database administration functions. Usually the two meanings coincide. Many sites have multiple DBAs.

Database Installation Administrator

Also called a database creator. This administrator is in charge of creating new databases. This includes registering each database in the directory using the Database Configuration Assistant. This administrator has create and modify access to database service objects and attributes. This administrator can also modify the Default Domain.

database link

A network object stored in the local database or in the network definition that identifies a remote database, a communication path to that database, and optionally, a username and password. Once defined, the database link is used to access the remote database.

A public or private database link from one database to another is created on the local database by a DBA or user.

A global database link is created automatically from each database to every other database in a network with Oracle Names. Global database links are stored in the network definition.

database method

See **Oracle database method**.

Database Security Administrator

Has create, modify, and read access for enterprise user security. This administrator has permissions on all of the domains in the enterprise and is responsible for:

- Administering the Oracle DBSecurityAdmins and OracleDBCreators groups.
- Creating new enterprise domains.
- Moving databases from one domain to another within the enterprise.

decryption

The process of converting the contents of an encrypted message (ciphertext) back into its original readable format (plaintext).

DES

The U.S. Data Encryption Standard.

dictionary attack

A common attack on passwords. the attacker creates a dictionary of many possible passwords and their corresponding verifiers. Through some means, the attacker then obtains the verifier corresponding to the target password, and obtains the target password by looking up the verifier in the dictionary.

Diffie-Hellman key negotiation algorithm

This is a method that allows two parties communicating over an insecure channel to agree upon a random number known only to them. Though the parties exchange information over the insecure channel during execution of the Diffie-Hellman key negotiation algorithm, it is computationally infeasible for an attacker to deduce the random number they agree upon by analyzing their network communications. Oracle Advanced Security uses the Diffie-Hellman key negotiation algorithm to generate session keys.

digital signature

A digital signature is created when a public key algorithm is used to sign the sender's message with the sender's private key. The digital signature assures that the document is authentic, has not been forged by another entity, has not been altered, and cannot be repudiated by the sender.

directory information tree (DIT)

A hierarchical tree-like structure consisting of the DNs of the entries.

directory naming context

A subtree which is of significance within a directory server. It is usually the top of some organizational subtree. Some directories only allow one such context which is fixed; others allow none to many to be configured by the directory administrator.

distinguished name (DN)

The unique name of a directory entry. It comprises all of the individual names of the parent entries back to the root.

domain

Any tree or subtree within the **Domain Name System (DNS)** namespace. Domain most commonly refers to a group of computers whose host names share a common suffix, the domain name.

Domain Name System (DNS)

A system for naming computers and network services that is organized into a hierarchy of **domains**. DNS is used in TCP/IP networks to locate computers through user-friendly names. DNS resolves a friendly name into an IP address, which is understood by computers.

In Net8, DNS translates the host name in a TCP/IP address into an IP address.

encryption

The process of disguising a message rendering it unreadable to any but the intended recipient.

enterprise domain

A directory construct that consists of a group of databases and **enterprise roles**. A database should only exist in one enterprise domain at any time.

Enterprise Domain Administrator

User authorized to manage a specific enterprise domain, including the authority to add new enterprise domain administrators.

enterprise role

Access privileges assigned to **enterprise users**. Enterprise roles are stored in the directory and contain one or more **global role**s.

enterprise user

A user defined and managed in a directory. Each enterprise user has a unique identify across an enterprise and uses a wallet to store its login credentials.

entry

The building block of a directory, it contains information about an object of interest to directory users.

external authentication

Verification of a user identity by a third party authentication service, such as Kerberos or RADIUS.

file system method

Storing fingerprint templates in files when configuring Identix Biometric authentication. The alternative is to use the **Oracle database method**.

global role

A role managed in a directory, but its privileges are contained within a single database.

HTTP

Hypertext Transfer Protocol: The set of rules for exchanging files (text, graphic images, sound, video, and other multimedia files) on the World Wide Web. Relative to the TCP/IP suite of protocols (which are the basis for information exchange on the Internet), HTTP is an application protocol.

HTTPS

The use of Secure Sockets Layer (SSL) as a sublayer under the regular HTTP application layer.

identity

The combination of the public key and any other public information for an entity. The public information may include user identification data such as, for example, an e-mail address. A user certified as being the entity it claims to be.

initial ticket

In Kerberos authentication, an initial ticket or ticket granting ticket (TGT) identifies the user as having the right to ask for additional service tickets. No tickets can be obtained without an initial ticket. An initial ticket is retrieved by running the kinit program and providing a password.

integrity

The guarantee that the contents of the message received were not altered from the contents of the original message sent.

IIOP

Internet Inter-ORB Protocol. A protocol developed by the Object Management Group (OMG) to implement CORBA solutions over the World Wide Web. IIOP enables browsers and servers to exchange integers, arrays, and more complex objects, unlike HTTP, which supports only transmission of text.

KDC/TGS

Key Distribution Center/Ticket Granting Service. In Kerberos authentication, the KDC maintains a list of user principals and is contacted through the kinit program for the user's **initial ticket**. The Ticket Granting Service maintains a list of service

principals and is contacted when a user wants to authenticate to a server providing such a service.

The KDC/TGS is a trusted third party that must run on a secure host. It creates ticket-granting tickets and service tickets. The KDC and TGS are usually the same entity.

Kerberos

A network authentication service developed under Massachusetts Institute of Technology's Project Athena that strengthens security in distributed environments. Kerberos is a trusted third-party authentication system that relies on shared secrets and assumes that the third party is secure. It provides single sign-on capabilities and database link authentication (MIT Kerberos only) for users, provides centralized password storage, and enhances PC security.

key

When encrypting data, a key is a value which determines the ciphertext that a given algorithm will produce from given plaintext. When decrypting data, a key is a value required to correctly decrypt a ciphertext. A ciphertext is decrypted correctly only if the correct key is supplied.

With a symmetric encryption algorithm, the same key is used for both encryption and decryption of the same data. With an asymmetric encryption algorithm (also called a public-key encryption algorithm or public-key cryptosystem), different keys are used for encryption and decryption of the same data.

kinstance

An instantiation or location of a service. This is an arbitrary string, but the host machine name for a service is typically specified.

kservice

An arbitrary name of a Kerberos service object.

LDAP

See **Lightweight Directory Access Protocol (LDAP)**.

Lightweight Directory Access Protocol (LDAP)

A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. The framework of design conventions supporting industry-standard directory products, such as the Oracle Internet Directory.

listener

A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.

Every time a client requests a network session with a server, a listener receives the actual request. If the client information matches the listener information, then the listener grants a connection to the server.

listener.ora file

A configuration file for the listener that identifies the:

- Listener name
- Protocol addresses that it is accepting connection requests on
- Services it is listening for

The `listener.ora` file typically resides in `$ORACLE_HOME/network/admin` on UNIX platforms and *`ORACLE_HOME`*`\network\admin` on Windows NT.

man-in-the-middle

A security attack characterized by the third-party, surreptitious interception of a message, wherein the third-party, the *man-in-the-middle*, decrypts the message, re-encrypts it (with or without alteration of the original message), and re-transmits it to the originally-intended recipient—all without the knowledge of the legitimate sender and receiver. This type of security attack works only in the absence of **authentication**.

MD5

An algorithm that assures data integrity by generating a 128-bit cryptographic message digest value from given data. If as little as a single bit value in the data is modified, the MD5 checksum for the data changes. Forgery of data in a way that will cause MD5 to generate the same result as that for the original data is considered computationally infeasible.

message authentication code

Also known as data authentication code (DAC). A **checksumming** with the addition of a secret key. Only someone with the key can verify the cryptographic checksum.

message digest

See **checksumming**.

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.

non-repudiation

Uncontestable proof ot he origin, delivery, submission, or transmission of a message.

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.

Oracle database method

Using an Oracle database to store fingerprint templates when configuring Indentix Biometric authentication. The alternative is to use the **file system method**.

plaintext

Message text that has not been encrypted.

principal

A uniquely-identified client or server. A Kerberos object, consisting of *kservice/kinstance@REALM*. See also *kservice*, *kinstance*, and *realm*.

private key

In public-key cryptography, this key is the secret key. It is primarily used for decryption but is also used for encryption with digital signatures. See **public/private key pair**.

public key

In public-key cryptography this key is made public to all, it is primarily used for encryption but can be used for verifying signatures. See **public/private key pair**.

public-key encryption

The process where the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the message is decrypted by the recipient using its private key.

public-key infrastructure

Information security technology utilizing the principles of public key cryptography. Public key cryptography involves encrypting and decrypting information using a shared (public) and private key pair. Provides for secure, private communications within a public network.

public/private key pair

A 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 one key from the other. Public keys are typically made widely available, while a private key is available only to the owner. Public and private keys are used only with asymmetric encryption algorithms, also called public-key encryption algorithms, or public-key cryptosystems. 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, and data cnrypted with a private key cannot be decrypted with the same private key.

realm

A Kerberos object. A set of clients and servers operating under a single key distribution center / ticket-granting service (KDC / TGS). *kservices* that are in different realms but that have the same name are unique.

root key certificate

See **trusted certificate**.

Secure Hash Algorithm

An algorithm that assures data integrity by generating a 160-bit cryptographic message digest value from given data. If as little as a single bit in the data is modified, the Secure Hash Algorithm checksum for the data changes. Forgery of a given data set in a way that will cause the Secure Hash Algorithm to generate the same result as that for the original data is considered ccomputationally infeasible.

An algorithm that takes a message of less than 264 bits in length and produces a 160-bit message digest. The algorithm is slightly slower than MD5, but the larger message digest makes it more secure against brute-force collision and inversion attacks.

Secure Sockets Layer (SSL)

An industry standard protocol designed by Netscape Communications Corporation for securing network connections. SSL provides authentication, encryption, and data integrity using public key infrastructure (PKI).

server

A provider of a service.

service

A network resource used by clients; for example, an Oracle database server.

service name

For Kerberos-based authentication, the **kservice** portion of a service principal.

service principal

See **principal**.

service table

In Kerberos authentication, a service table is a list of service principals that exist on a *kinstance*. This information must be extracted from Kerberos and copied to the Oracle server machine before Kerberos can be used by Oracle.

service ticket

Trusted information used to authenticate the client. A ticket-granting ticket is also known as the initial ticket, is obtained by directly or indirectly running kinit and providing a password, and is used by the client to ask for service tickets. A *service ticket* is used by a client to authenticate to a service.

session key

A key shared by at least two parties (usually a client and a server) that is used for data encryption for the duration of a single communication session. Session keys are typically used to encrypt network traffic; a client and a server can negotiate a session key at the beginning of a session, and that key is used to encrypt all network traffic between the parties for that session. If the client and server communicate again in a new session, they negotiate a new session key.

session layer

A network layer that provides the services needed by the presentation layer entities that enable them to organize and synchronize their dialogue and manage their data exchange. This layer establishes, manages, and terminates network sessions between the client and server. An example of a session layer is Network Session.

SHA

See **Secure Hash Algorithm**.

shared schema

Database or application schemas that can be used by multiple enterprise users. Oracle Advanced Security supports the mapping of multiple enterprise users to the same shared schema on a database, which lets an administrator avoid creating an account for each user in every database. Instead, the administrator can create a user in one location, the enterprise directory, and map the user to a shared schema that other enterprise users can also map to. Sometimes called **user/schema separation**.

single sign-on

The ability for a user to authenticate once, combined with strong authentication occurring transparently in subsequent connections to other databases or applications. Single sign-on lets a user access multiple accounts and applications with a single password. Oracle Advanced Security supports Kerberos, CyberSafe, DCE, and SSL-based single sign-on.

smart card

A plastic card (like a credit card) with an embedded integrated circuit for storing information, including such information as user names and passwords, and also for performing computations associated with authentication exchanges. A smart card is read by a hardware device at any client or server.

A smartcard can generate random numbers which can be used as one-time use passwords. In this case, smartcards are synchronized with a service on the server so that the server expects the same password generated by the smart card.

sniffer

Device used to surreptitiously listen to or capture private data traffic from a network.

sqlnet.ora file

A configuration file for the client or server that specifies:

- Client domain to append to unqualified service names or net service names
- Order of naming methods the client should use when resolving a name
- Logging and tracing features to use
- Route of connections
- Preferred Oracle Names servers
- External naming parameters
- Oracle Advanced Security parameters

The `sqlnet.ora` file typically resides in `$ORACLE_HOME/network/admin` on UNIX platforms and *`ORACLE_HOME`*`\network\admin` on Windows platforms.

ticket

A piece of information that helps identify who the owner is. See **service ticket**.

token card

A device for providing improved ease-of-use for users through several different mechanisms. Some token cards offer one-time passwords that are synchronized with an authentication service. The server can verify the password provided by the token card at any given time by contacting the authentication service. Other token cards operate on a challenge-response basis. In this case, the server offers a challenge (a number) which the user types into the token card. The token card then provides another number (cryptographically-derived from the challenge), which the user then offers to the server.

transport layer

A networking layer that maintains end-to-end reliability through data flow control and error recovery methods. Net8 uses Oracle protocol supports for the transport layer.

trusted certificate

A trusted certificate, sometimes called a root key certificate, is a third party identity that is qualified with a level of trust. The trusted certificate is used when an identity is being validated as the entity it claims to be. Typically, the certificate authorities you trust are called trusted certificates. If there are several levels of trusted certificates, a trusted certificate at a lower level in the certificate chain does not need to have all its higher level certificates reverified.

trusted certificate authority

See **certificate authority**.

trust point

See **trusted certificate**.

user/schema separation

See **shared schema**.

wallet

A wallet is a data structure used to store and manage security credentials for an individual entity. It implements the storage and retrieval of credentials for use with various cryptographic services. A **Wallet Resource Locator** (WRL) provides all the necessary information to locate the wallet.

Wallet Resource Locator

A wallet resource locator (WRL) provides all necessary information to locate a wallet. It is a path to an operating system directory that contains a wallet.

WRL

See **Wallet Resource Locator**.

X.509

Public keys can be formed in various data formats. The X.509 v3 format is one such popular format.

Index

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

R

S

T

U

V

W

X