Enterprise DBA Part 3: Network Administration

Student Guide

30051GC10

Production 1.0

August 1999

M09125

ORACLE®

Author
Sharaaz Khan

Technical Contributors and Reviewers
Peter Kilpatrick
Bruce Ernst
Hanne Rasmussen
A. J. Vos
Mary Bryksa
Joel Goodman
Andrew Philips
Alexander Hunold

Publisher
Tommy Cheung

Copyright © Oracle Corporation, 1999. All rights reserved.

This documentation contains proprietary information of Oracle Corporation. It is provided under a license agreement containing restrictions on use and disclosure and is also protected by copyright law. Reverse engineering of the software is prohibited. If this documentation is delivered to a U.S. Government Agency of the Department of Defense, then it is delivered with Restricted Rights and the following legend is applicable:

Restricted Rights Legend

Use, duplication or disclosure by the Government is subject to restrictions for commercial computer software and shall be deemed to be Restricted Rights software under Federal law, as set forth in subparagraph (c) (1) (ii) of DFARS 252.227-7013, Rights in Technical Data and Computer Software (October 1988).

This material or any portion of it may not be copied in any form or by any means without the express prior written permission of Oracle Corporation. Any other copying is a violation of copyright law and may result in civil and/or criminal penalties.

If this documentation is delivered to a U.S. Government Agency not within the Department of Defense, then it is delivered with “Restricted Rights,” as defined in FAR 52.227-14, Rights in Data-General, including Alternate III (June 1987).

The information in this document is subject to change without notice. If you find any problems in the documentation, please report them in writing to Education Products, Oracle Corporation, 500 Oracle Parkway, Box SB-6, Redwood Shores, CA 94065. Oracle Corporation does not warrant that this document is error-free.

Oracle is a registered trademark and trademarks or registered trademarks of Oracle Corporation.

All other products or company names are used for identification purposes only and may be trademarks of their respective owners.

Introduction

Course Objectives I-2
Database Administrator Tasks I-4
Course Schedule I-5

Lesson 1: Networking Overview

Objectives 1-2
Network Environment Challenges 1-3
Network Configurations 1-5
Oracle's Solutions 1-8
Summary 1-20

Lesson 2: Basic Net8 Architecture

Objectives 2-2
Overview 2-3
Basic Operations 2-4
Files and Locations 2-5
The Net8 Stack 2-10
Summary 2-20

Lesson 3: Basic Net8 Server-Side Configuration

Objectives 3-2
Overview: The Listener Process 3-3
The Listener Responses 3-4
Bequeath Session 3-5
Redirect Session (Dedicated) 3-6
Redirect Session (Dispatcher) 3-7
The LISTENER.ORA File 3-9
LISTENER.ORA File Parameters 3-12
Listener Configuration: Creation 3-14
Listener Configuration: Services 3-15
Listener Control Utility (LSNRCTL) 3-16
LSNRCTL Commands 3-17
Additional LSNRCTL Commands 3-18

LSNRCTL SET and SHOW Modifiers 3-19
Automatic Instance Registration 3-21
Automatic Instance Registration: Parameters 3-22
Troubleshooting the Listener 3-23
Summary 3-25
Frequently Asked Questions 3-26

Lesson 4: Basic Net8 Client-Side Configuration

Objectives 4-2
Overview 4-3
Host Naming 4-4
Local Naming 4-6
Configuring Local Naming Using Net8 Assistant 4-7
Generated Files 4-16
New TNSNAMES.ORA Parameters 4-18
Connection Load Balancing 4-20
Connection Load Balancing and Failover: Example 4-22
Troubleshooting the Client Side 4-23
Summary 4-25
Frequently Asked Questions 4-26

Lesson 5: Centralized Naming Concepts

Objectives 5-2
What Is a Service Name? 5-3
Service Name Resolution Methods 5-4
Resolution with Local Naming 5-5
Resolution with Centralized Naming 5-6
Centralized Naming Using a Names Server 5-7
When to Use a Names Server 5-8
Oracle Names Directory Objects 5-9
Domain Naming Models: Flat 5-10
Domain Naming Models: Hierarchical 5-11
Names Server Dynamic Discovery 5-12
Client-Side Cache 5-13

Cache Replication 5-14
The Region Database 5-15
Summary 5-16

Lesson 6: Oracle Names Usage and Configuration

Objectives 6-2
Configuring Centralized Naming 6-3
Configuring the Names Server (Cache) 6-5
Configuring the Client Profile 6-11
Testing the Names Server 6-16
Configuring a Region Database 6-17
Names Control Utility (NAMESCTL) 6-20
NAMESCTL Commands 6-21
Other NAMESCTL Commands 6-24
NAMESCTL SET Modifier 6-26
NAMESCTL SHOW Modifier 6-27
Summary 6-29

Lesson 7: Multithreaded Server Usage and Configuration

Objectives 7-2
Server Configurations 7-3
Multithreaded Server Architecture 7-9
Configuring the Multithreaded Server 7-12
Data Dictionary 7-23
Connection Pooling 7-25
Summary 7-29

Lesson 8: Connection Manager Usage and Configuration

Objectives 8-2
Configuring Connection Manager 8-5
Summary 8-21

Lesson 9: Troubleshooting the Network Environment

Objectives 9-2
Overview 9-3

Troubleshooting Checklist 9-4
TNSPING Utility 9-6
Net8 Assistant: Logging and Tracing 9-7
Listener Audit Trail 9-14
Logging and Tracing Components 9-15
Trace Assistant 9-18
Summary 9-21
Frequently Asked Questions 9-22

Lesson 10: Security in the Network Environment

Objectives 10-2
Overview: Network Security Risks 10-3
Data Privacy: Data Theft 10-4
Data Integrity: Data Modification 10-5
Data Integrity: Data Disruption 10-6
Compromised Authentication 10-7
Compromised Authorization 10-8
Network Security Solutions 10-9
Data Encryption 10-10
Cryptographic Checksumming 10-11
Configuring Encryption and Checksumming 10-12
Encryption and Checksumming Modes 10-13
Authentication 10-14
Enhanced User Authentication 10-15
Token Cards 10-16
Biometric Authentication 10-17
Kerberos Authentication 10-18
RADIUS Authentication 10-19
Configuring Authentication 10-20
Single Sign-On 10-21
Secure Sockets Layer 10-22
DCE Integration 10-23
Summary 10-24

Appendix A: Practices

Practice 3 A-2
Practice 4 A-3
Practice 6 A-4
Practice 7 A-6
Practice 8 A-7
Practice 9 A-8

Appendix B: Practice Solutions

Practice 3 Solutions B-2
Practice 4 Solutions B-6
Practice 6 Solutions B-12
Practice 7 Solutions B-18
Practice 8 Solutions B-22
Practice 9 Solutions B-26

Appendix C: Configuring External Procedures

Configuring External Procedures C-2

1

Introduction

Course Objectives

Objectives

At the end of this course, you should be able to do the following:

- **Identify network trends and problems, and provide solutions for them**
- **Define the Net8 architectural layers**
- **Configure a simple client and server, and establish a connection between them**
- **Configure and start a Names server and use it to resolve a service name**

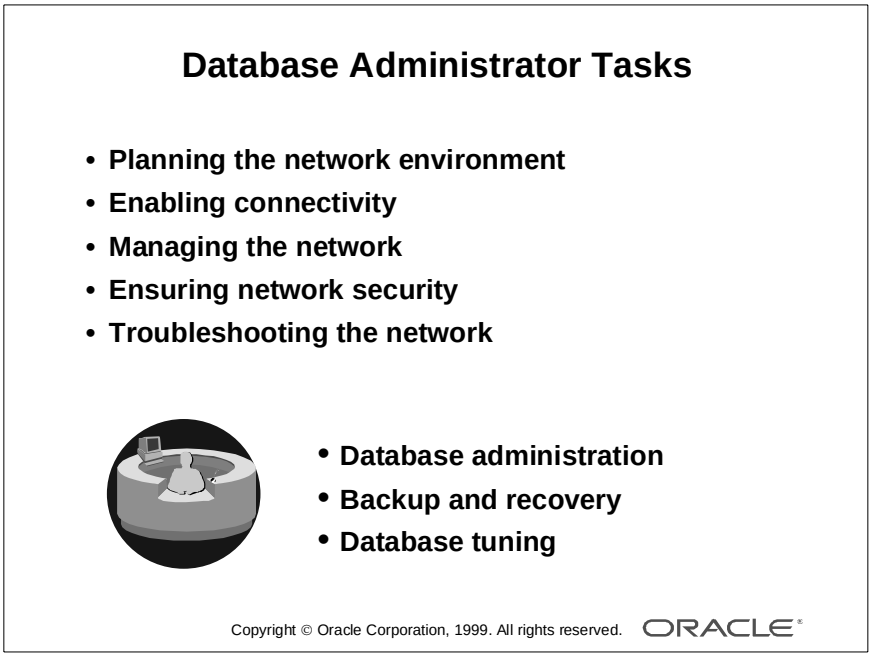
Copyright © Oracle Corporation, 1999. All rights reserved.

Objectives

- **Configure and start up a multithreaded server**
- **Configure the Connection Manager and use it for pooling connections and restricting clients from connecting**
- **Analyze and troubleshoot Net8 problems using log files, trace files, and Trace Assistant**
- **Identify network security risks and their solutions and configure data encryption using the Oracle Advanced Security option**

Copyright © Oracle Corporation, 1999. All rights reserved.

Database Administrator Tasks



Scope of the Course

This course is the fourth in a series of four courses that cover the core database administrator tasks. The tasks covered in this course are:

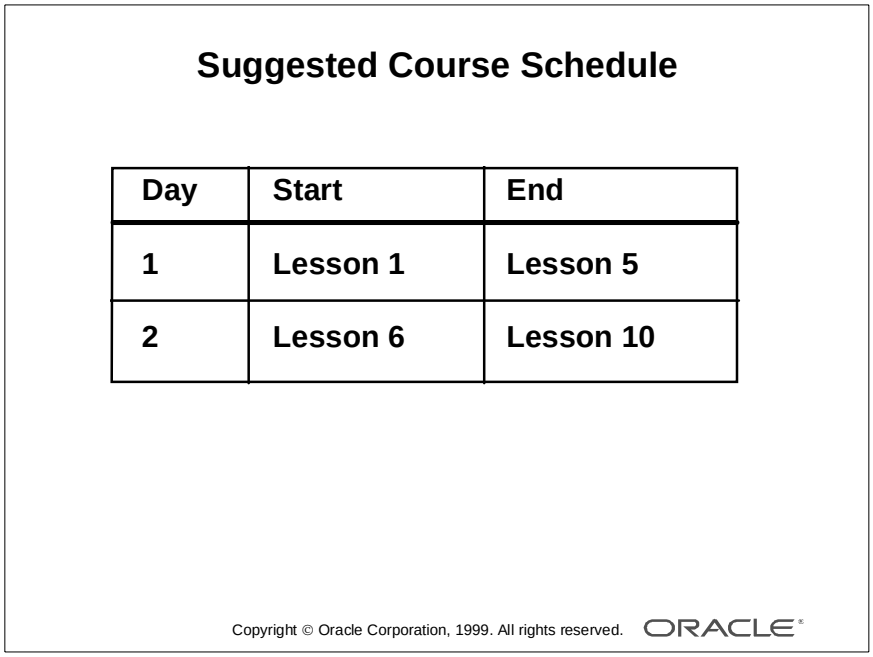
- Planning the network environment to address connectivity, performance, and security issues
- Enabling network client-server, server-server, and middle-tier connectivity
- Managing the network through administration utilities and tools
- Ensuring solid network integrity, privacy, authentication, and authorization
- Invoking the tools needed for troubleshooting the network environment

DBA Tasks Covered in Other Courses

The following tasks are discussed in other courses:

- Database administration in *Enterprise DBA Part 1A: Architecture and Administration*
- Backup and recovery in *Enterprise DBA Part 1B: Backup and Recovery*
- Database tuning in *Enterprise DBA Part 2: Performance and Tuning*

Course Schedule



Agenda

The following is the recommended lesson schedule for this course:

Day 1

- Networking Overview
- Basic Net8 Architecture
- Basic Net8 Server-Side Configuration
- Basic Net8 Client-Side Configuration
- Centralized Naming Concepts
- Oracle Names Usage and Configuration (Theory)

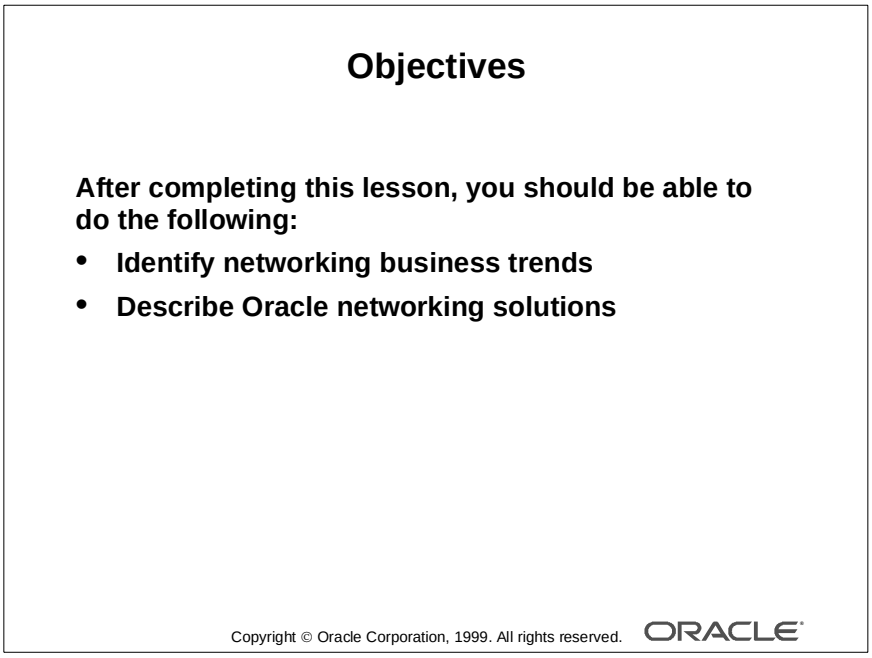
Day 2

- Oracle Names Usage and Configuration (Practice)
- Multithreaded Server Usage and Configuration
- Connection Manager Usage and Configuration
- Troubleshooting the Network Environment
- Security in the Network Environment

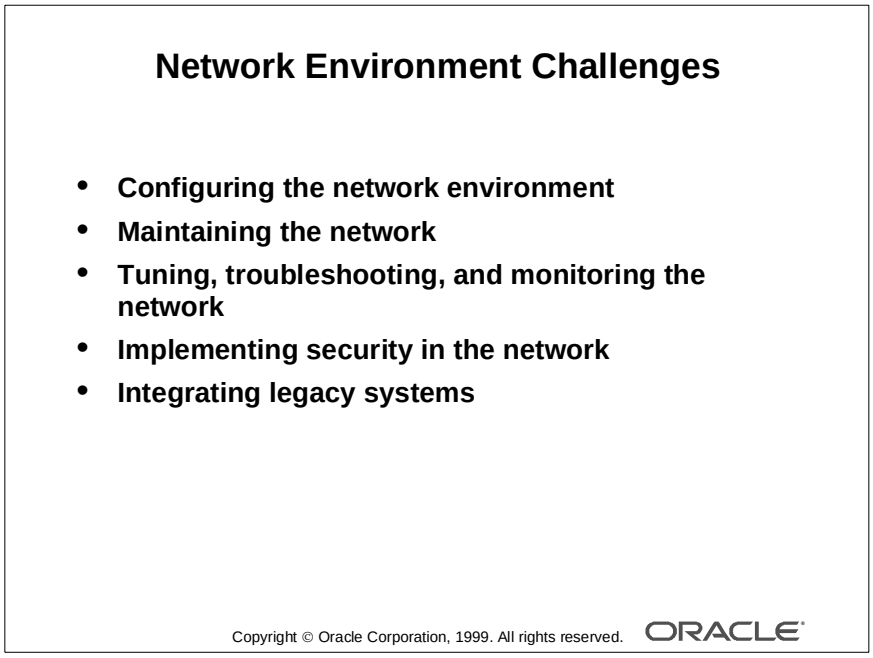
1

Networking Overview

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Identify networking business trends
• Describe Oracle networking solutions
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Network Environment Challenges



Configuring the Network Environment

To implement a successful networking environment consider the following questions:

- What type of network are you configuring? Is it a small network with a few clients, or a more complex network with many clients and many servers?
- Are you using a single protocol or multiple protocols?
- Is the network static or expanding?
- What configuration options do you have?
- Are there user-friendly tools available to configure the network?

Maintaining the Network

- How much network maintenance is required for your enterprise?
- Will you add clients and servers to your network?
- Do you anticipate frequent upgrades?

Tuning, Troubleshooting, and Monitoring the Network

- Does your network include the needed tools?
- How large a workload do you anticipate?
  - Number of users
  - Number of transactions
  - Number of nodes
  - Location of nodes

Implementing Security in the Network

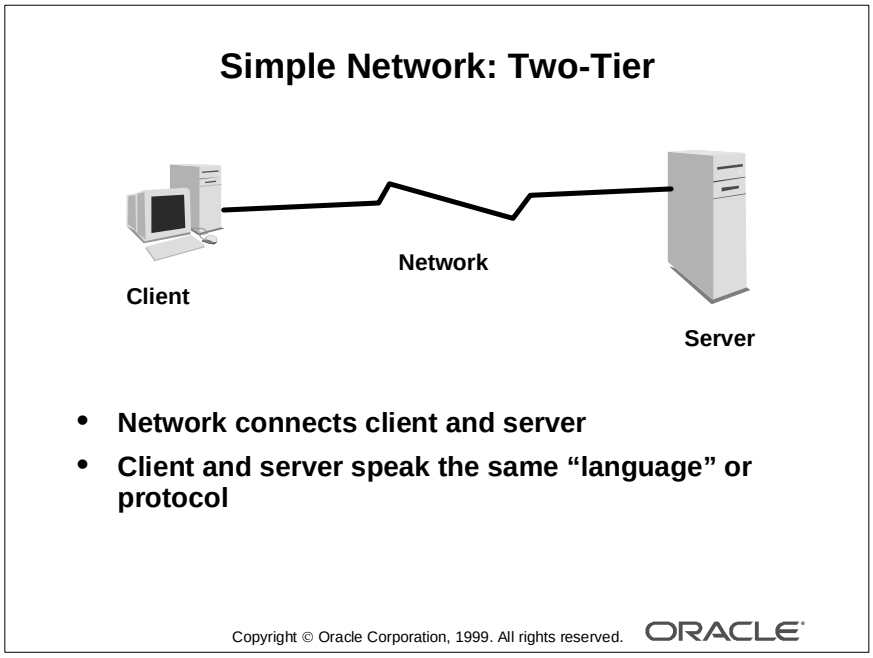
- Do you need to secure your network environment?
- Is secure and sensitive information being transmitted over the network?
- What tools are available for implementing security?

Integrating Legacy Systems

How will your legacy systems interact with your networking environment?

Note: Performing an up-front analysis that answers questions like these helps you choose the appropriate network strategy from the beginning.

Network Configurations



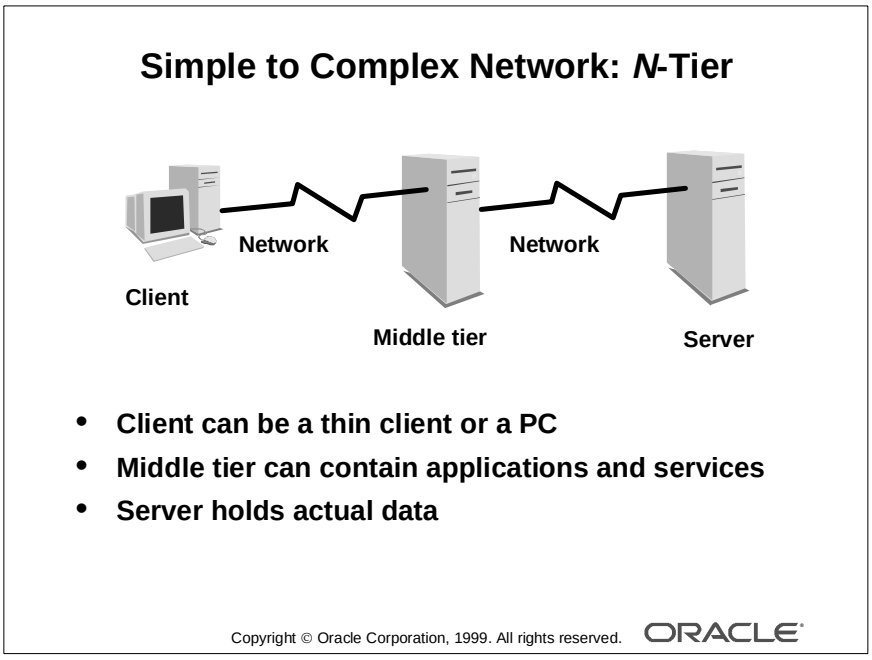
Two-Tier Networks

In a two-tier network, a client communicates directly with a server. This is also known as a *client-server architecture*.

A client-server network is an architecture that involves client processes that request service from server processes.

The client and server communicate over a network using a given protocol, which must be installed on both the client and the server.

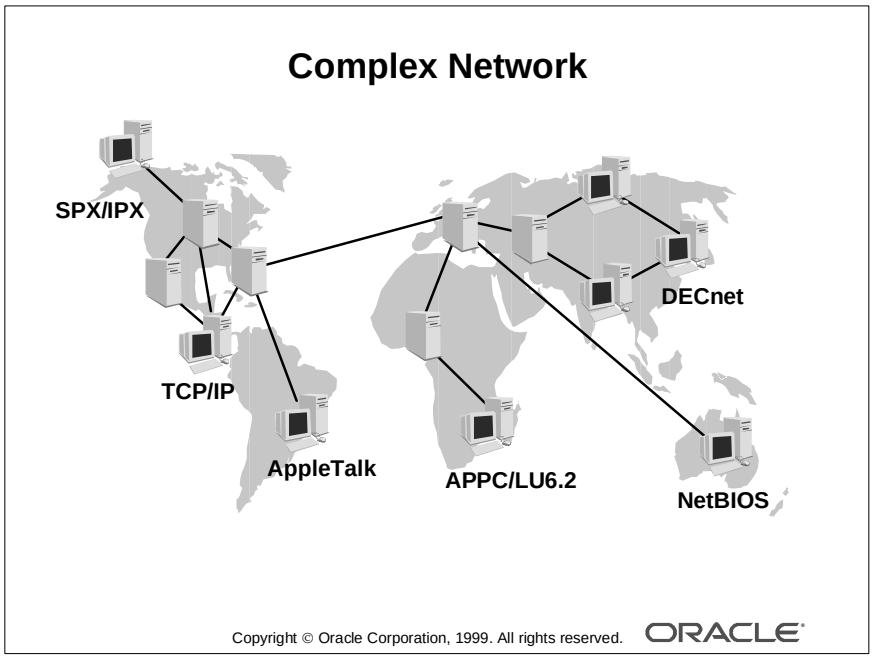
A common error in client-server network development is to prototype an application in a small, two-tier environment and then scale up by simply adding more users to the server. This approach can result in an ineffective system, as the server becomes overburdened. To properly scale to hundreds or thousands of users, it may be necessary to implement an *N-tier architecture*, which introduces one or more servers or agents between the client and server.



N-Tier Networks

In an *N*-tier architecture, the role of the middle-tier agent can be manifold. It can provide:

- Translation services (as in adapting a legacy application on a mainframe to a client-server environment or acting as a bridge between protocols)
- Scalability services (as in acting as a transaction-processing monitor to balance the load of requests between servers)
- Intelligent agent services (as in mapping a request to a number of different servers, collating the results, and returning a single response to the client)



Complex Network Issues

Networks should improve communication rather than impede distributed operations. In a more complex network environment, several issues must be addressed:

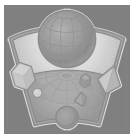
- Different hardware platforms that run different operating systems
- Multiple protocols used on these platforms
- Variable syntax issues between the different but connected applications
- Different geographical locations in which the connected applications reside

A well-designed complex network can support a large-scale distributed system.

Oracle’s Solutions

Oracle’s Solutions

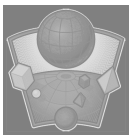
- **Net8**
- **Oracle Names server**
- **Connection Manager**
- **Advanced Security option**
- **Open Gateways**



Copyright © Oracle Corporation, 1999. All rights reserved.

Oracle's Solutions: Net8

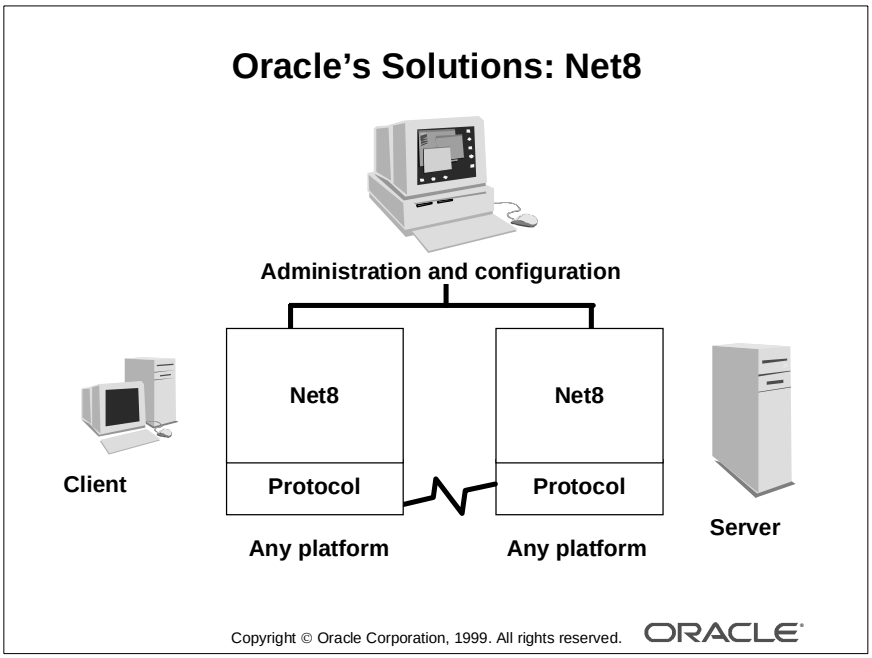
- **Protocol independence**
- **Comprehensive platform support**
- **Integrated GUI administration tools**
- **Multiple configuration options**
- **Tracing and diagnostic toolset**
- **Open API**
- **Basic security**



Copyright © Oracle Corporation, 1999. All rights reserved.

Net8 Key Features

Net8 introduces key new features to address the changes occurring from the growth in distributed environments. These changes include increasing user access to data stores, creating more easily configured and administered environments, and enhancing user authentication to securely identify users.

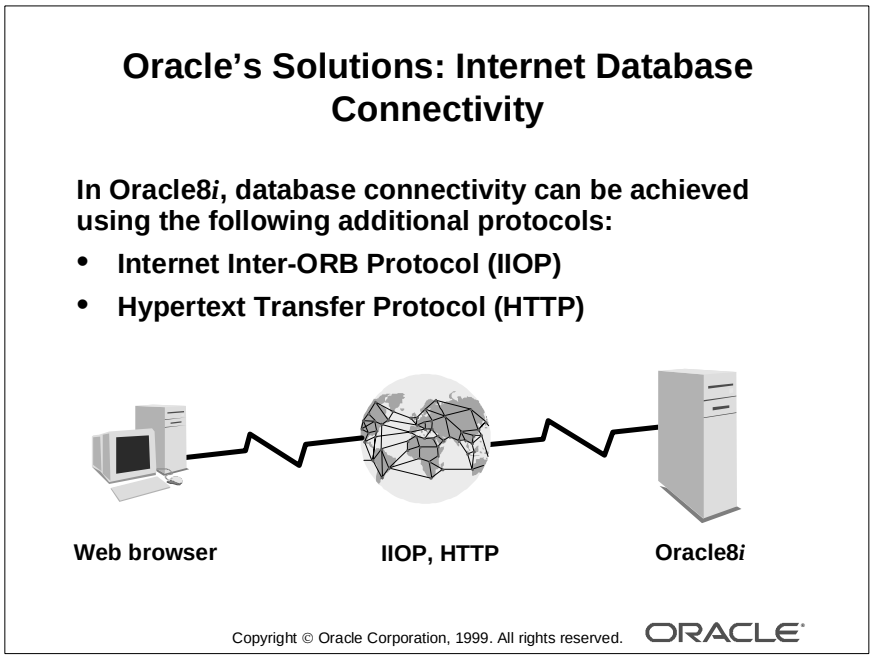


Net8

Net8 provides the industry's broadest support for network transport protocols, including TCP/IP, Novell SPX/IPX, IBM LU6.2, and DECnet. All data conversion using Net8 is invisible to the user and the application. This enables Oracle8*i* to operate across different types of computers, operating systems, and networks to transparently connect any combination of PC, UNIX, legacy, and other systems without expensive changes to the existing infrastructure.

Net8 contains configuration and administration mechanisms and eliminates the need for a centralized configuration utility. For simple environments, Net8's default settings provide a transparent name resolution adapter. This eliminates the need for generating configuration files. For more complicated environments, Oracle Names server stores connection information in a database or in a local data cache.

In Oracle8*i*, Net8 addresses Internet connectivity through integration of standard solutions such as Remote Authentication Dial-In User Service (RADIUS) and Lightweight Directory Access Protocol (LDAP) with legacy systems.



IIOP and HTTP Connectivity

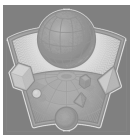
Connections to the database are not limited to Net8 alone; clients can establish connections to the database using Internet protocols such as Internet Inter-ORB Protocol (IIOP) and Hypertext Transfer Protocol (HTTP). Using these Internet protocols, users can run applications from within a Web browser to connect directly to an Oracle8*i* database. Internet technologies such as iFS (Internet File System), Enterprise JavaBeans (EJB), and the Internet standard Secure Sockets Layer (SSL) protocol provide added security to network connections.

Note: Net8 supports a new presentation layer called General Inter-ORB Protocol (GIOP) that is used for clients that connect to the Java option. IIOP is an implementation of GIOP over TCP/IP or TCP/IP with SSL. Oracle provides the GIOP service implementation.

Oracle's Solutions: Oracle Names

Oracle Names offers:

- **Centralized configuration**
- **Simplified network administration**



Copyright © Oracle Corporation, 1999. All rights reserved.

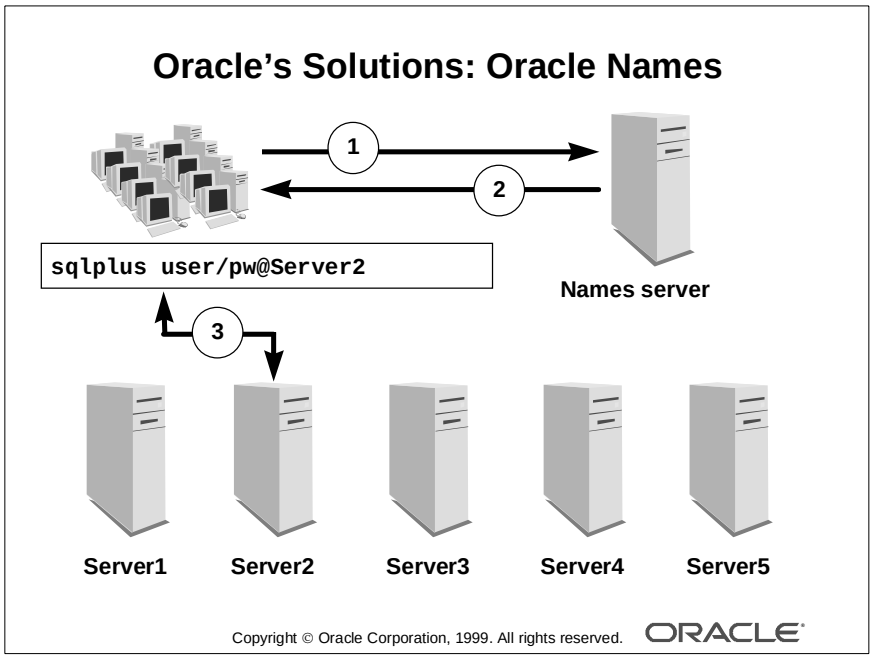
Oracle Names

Oracle Names is a distributed name service, similar to the Yellow Pages, that stores all the database addresses for the network environment. You can install one or more Oracle Names servers in your environment. They can be configured to act as backup servers for one another.

Using Oracle Names, clients do not need to know where the databases are located in the environment. The only configuration required is the address of the Oracle Names server. When a user wants to access the database, Net8 calls Oracle Names, resolves the address, and connects to the address given by the Oracle Names server.

Benefits

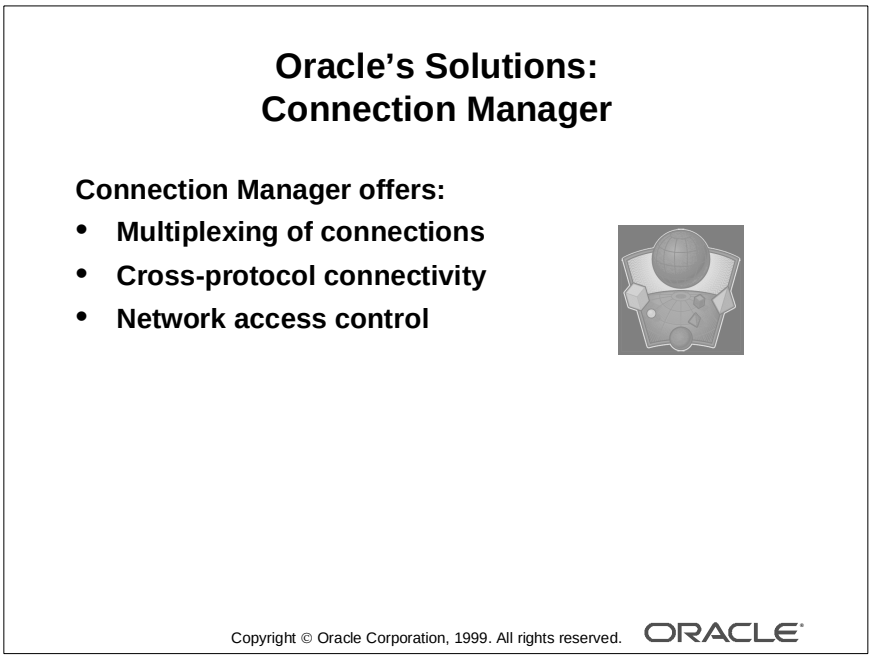
- High availability
- Makes Net8 clients easy to deploy and administer
- System can be scaled to meet the needs of the entire enterprise



Address Resolution Using Oracle Names

This example describes the basic steps taken to resolve a service name when using an Oracle Names configuration.

1. When a user issues a request to connect to the service name Server2, the Names server is contacted to get the address for Server2. The address can be stored locally but that would add a configuration overhead if you support multiple clients.
2. The address of Server2 is resolved and the address is passed back to the client.
3. The client connects using the address it got from the Names server. When the connection is established, there is no further communication between the client and the Names server.



Connection Manager

Connection Manager is a tool normally configured and installed on a middle tier. The Connection Manager can be configured for the following features:

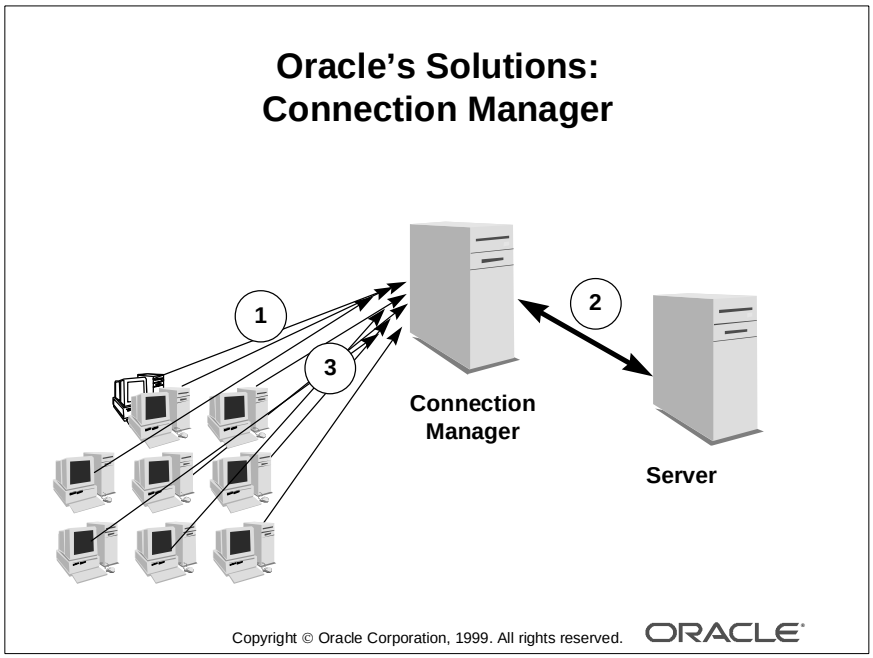
Multiplexing Connection Manager can handle several incoming connections and transmit them simultaneously over a single outgoing connection to the destination. This technique, called *multiplexing*, gives larger numbers of users access to a server. The configuration is offered only in a TCP/IP environment.

Cross-Protocol Connectivity Using this feature, a client and a server can communicate with different network protocols.

Network Access Control Using Connection Manager, designated clients can connect to certain servers in a network based on the TCP/IP protocol.

Benefits of Connection Manager

- Supports more users on the end tier if you use a middle tier to deploy Connection Manager and provides for better use of resources and scalability
- Enables cross-protocol communication
- Can act as an access control mechanism



Connection Multiplexing

This example shows how Connection Manager acts as a multiplexer to funnel data from many clients to one server.

1. The initial connection from a client to a server is established by connecting to Connection Manager.
2. Connection Manager establishes the connection to the server.
3. When additional clients are request connections to the server through Connection Manager, they use the same connection that Connection Manager used for the initial connection.

Oracle's Solutions: Advanced Security Option

- **Network security using encryption**
- **Integration with third-party security servers**
- **DCE Integration**



Copyright © Oracle Corporation, 1999. All rights reserved.

Oracle Advanced Security Option

Oracle Advanced Security option is an optional product that works with Net8 and SQL*Net release 2.3.4 and later SQL*Net versions. It includes the following features:

Security Services With the Oracle Advanced Security option, Net8 and related products can use network data encryption and checksumming so that data cannot be read or altered during transmission.

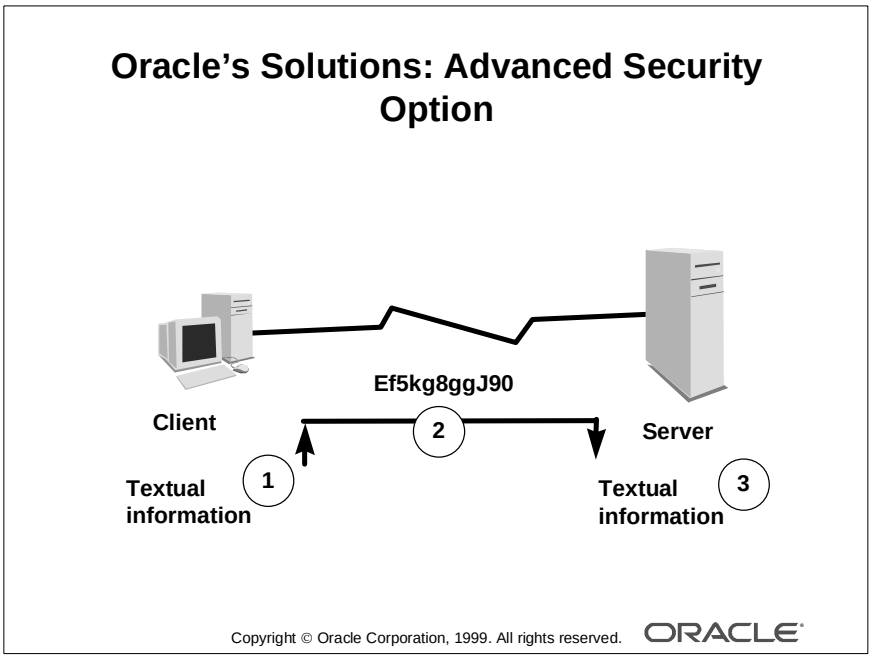
Authentication Services Oracle Advanced Security option includes enhanced user authentication services such as support for single sign-on. No changes to applications are required. Oracle Advanced Security option works over all protocols, operating systems, and name services. It supports token authentication through Security Dynamics ACE Server, Kerberos, DCE Security Server, and supports biometrics authentication through Identix. With Oracle8*i*, authentication is also supported through RADIUS with services by Lucent, ActivCard, Funk, and Secure Computing. X.509 Certificates are also supported with Oracle8*i*.

Oracle Advanced Security Option (continued)

Oracle DCE Integration Oracle Distributed Computing Environment (DCE) Integration is an optional product that works with Net8 and SQL*Net 2.1.6 and later. With it, users can transparently use Oracle tools and applications to access Oracle7 through Oracle8*i* Servers in a DCE environment. It provides authenticated RPC (remote procedure call) as the transport mechanism, which enables multivendor interoperability. Using the DCE Integration security service, a user logged in to DCE can securely access any Oracle application without having to specify a username or password.

Benefits

- Enables encrypted transmission of data
- Enables integration with third-party security servers



Encryption Using Advanced Security Option

This example shows one of the major tasks of a secure transmission through a network.

To ensure such a transmission, Oracle Advanced Security option must be installed and configured on both the client and the server side.

Once Advanced Security option is configured, data in all transmissions over Net8 can be encrypted as follows:

1. Start sending textual information from the client side. One layer of the network on the client side encrypts the information before it is transmitted over the network link.
2. Encrypted data, potentially including checksumming with each package sent, transmitted over the network link.
3. On the server side, the message is decrypted, and checksums can ensure that the data arrive in the correct order without tampering. Only the server that holds the correct key can decrypt the information and verify the checksumming sequence of the data.

Oracle's Solutions: Open Gateways

- **You can access legacy data as if it resides in a single, local relational database.**
- **Open Gateways offers:**
  - **Transparent gateways**
  - **Procedural gateways**



Copyright © Oracle Corporation, 1999. All rights reserved.

Open Gateways

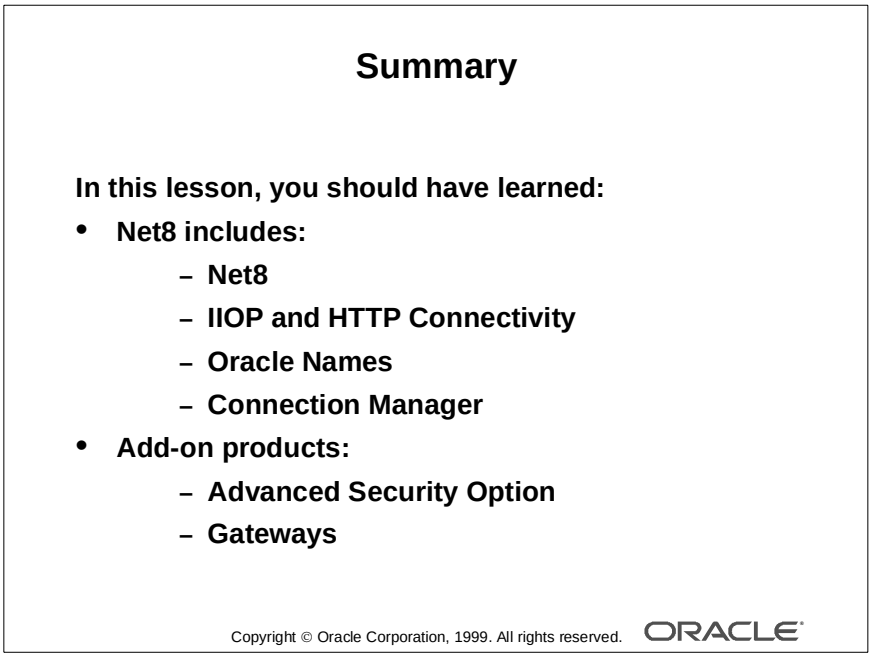
The Oracle Open Gateways provide seamless integration between the Oracle server and environments other than Oracle. Transparent gateways give users SQL access to more than 30 database environments. Procedural gateways extend Oracle-stored procedures to mainframe TP monitors or to message and queuing systems.

Benefit

Oracle Open Gateways enable integration with foreign data sources.

Note: Configuration of Open Gateways not covered in this class.

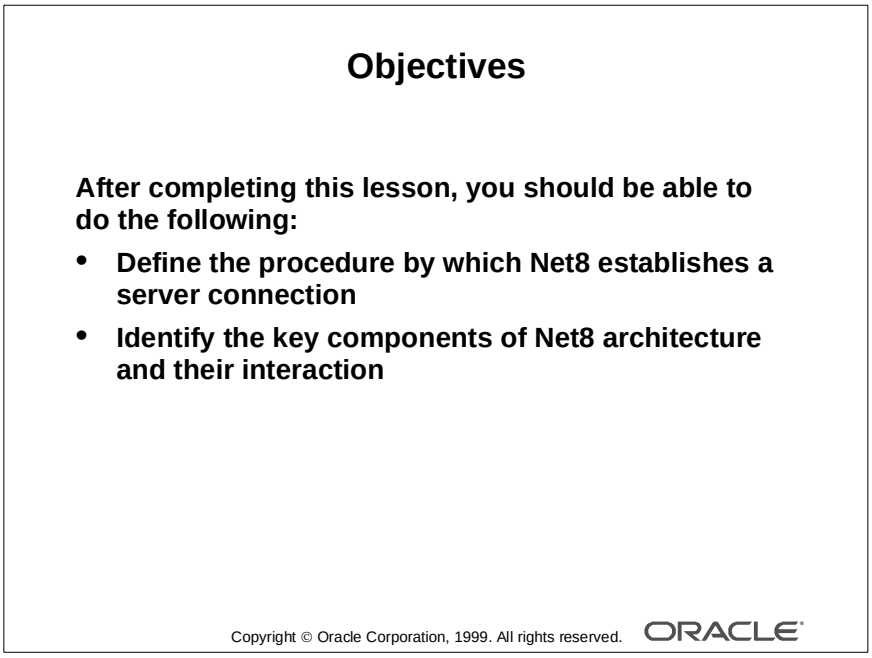
Summary


Summary
In this lesson, you should have learned:
• Net8 includes:
– Net8
– IIOP and HTTP Connectivity
– Oracle Names
– Connection Manager
• Add-on products:
– Advanced Security Option
– Gateways
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

2

Basic Net8 Architecture

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Define the procedure by which Net8 establishes a server connection
• Identify the key components of Net8 architecture and their interaction
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Overview

Overview

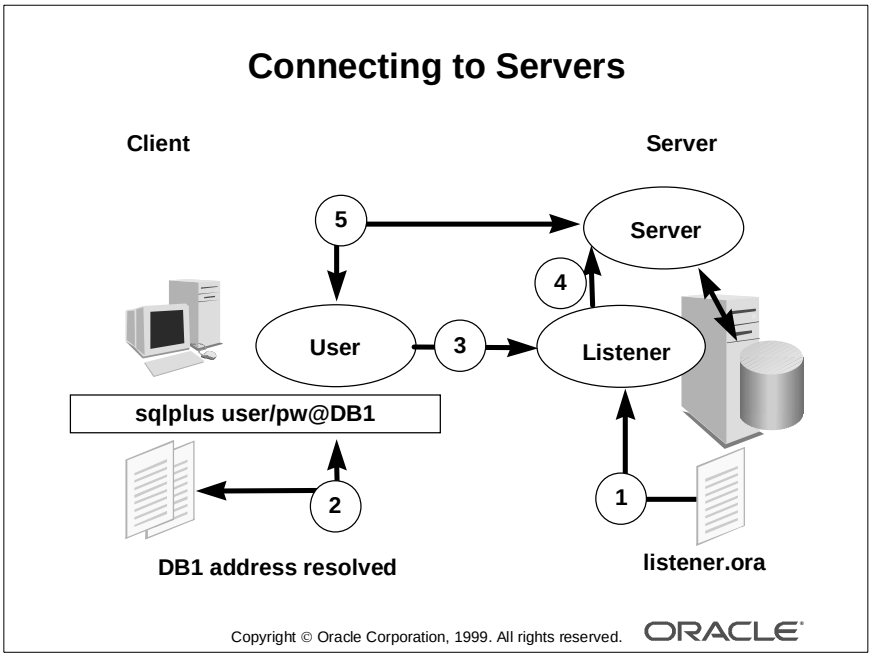
Net8 provides three basic functions:

- **Connect operations**
- **Data operations**
- **Exception operations**

The Net8 architecture comprises several layers, each of which has a unique responsibility in a networking session.

Copyright © Oracle Corporation, 1999. All rights reserved.

Basic Operations



Connecting to a Server (Open)

The following procedure is true for all connections and is illustrated in the slide example.

1. In order for a client to connect to a server, a process needs to be listening for incoming requests on the server side. The listener process is responsible for detecting and routing the incoming requests to the proper destination.

   To start a listener process, a configuration file located on the server side typically must be configured.

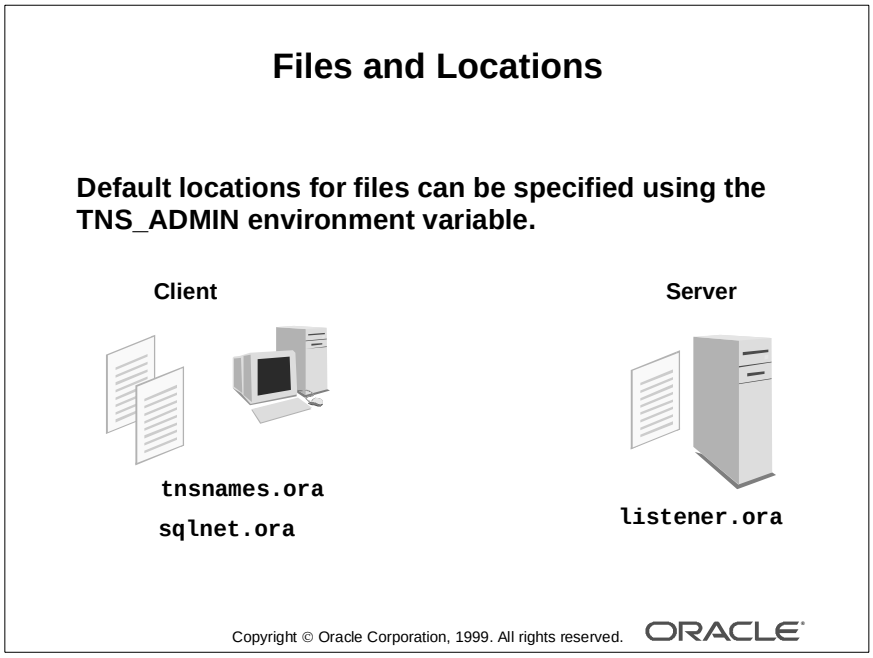
2. A user or program resolves the *service name* to a connection string. A service name is an alias for a destination address.

   The service name directly maps to a connect descriptor, which is resolved through one of the following methods (described fully in later lessons):

   – Host naming (using the `sqlnet.ora` file and a native naming service)
   – Local naming (using the `tnsnames.ora` file)
   – Oracle Names (using the `sqlnet.ora` file and an Oracle Names server)

3. When the service name is resolved, a request is sent to the listener. The listener receives the session request and determines where to direct the request.
4. The listener spawns a new process or redirects the connection to an existing process that handles the communication with the Oracle database.
5. The address of the process is handed off to the client-side process, and the client side directly communicates with the server-side process for the duration of the session without involving the listener.

Files and Locations



Files

`tnsnames.ora` This configuration file contains the names and addresses of services on the network. It is used by both clients and distributed servers to identify destination servers.

`sqlnet.ora` This file is used by all clients and servers on the network. It contains client profile information, including optional diagnostic parameters, client information about Oracle Names (if used), and other optional parameters such as native naming and security.

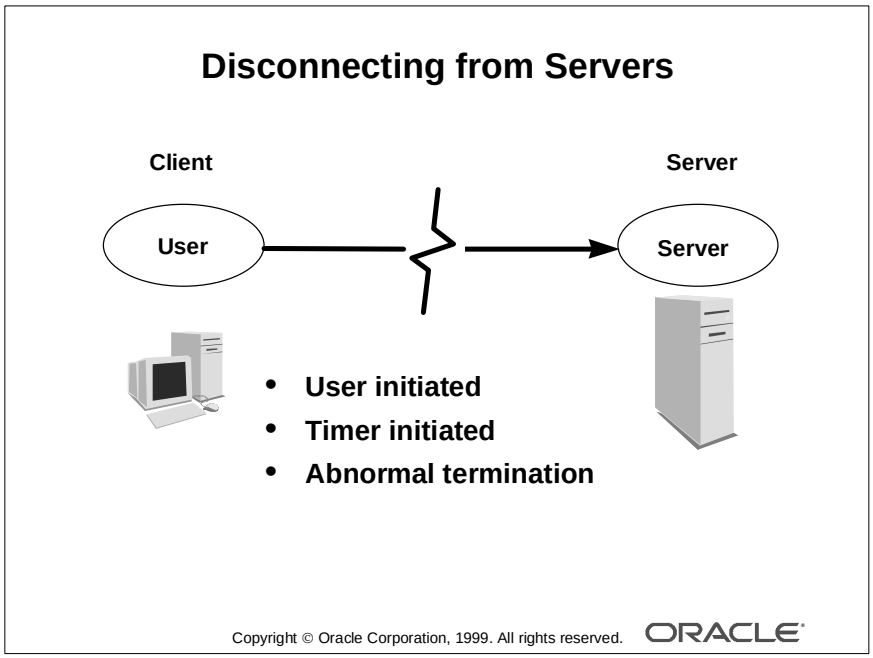
`listener.ora` This file contains configuration information for the listener located on the machine that you want to connect to. The listener is an executable program that enables an Oracle8 Server to accept connections from client machines over Net8.

Locations

UNIX Workstations The default location for these files is the `$ORACLE_HOME/network/admin` directory. You can also place them in this directory or place them in a directory of your choice by setting an environment variable TNS_ADMIN to point to the directory containing the files.

NT Workstations On an NT workstation, the default directory for the configuration files is `ORACLE_HOME\NET80\ADMIN`. You can add the TNS_ADMIN parameter to change the directory name for configuration files from the default location. To do this add the TNS_ADMIN variable to the registry or add it to the environment under System Properties, found by clicking the System icon on the Control Panel.

Note: On Oracle8*i* for NT, the default directory for the configuration files is `ORACLE_HOME\NETWORK\ADMIN`.



Disconnecting from a Server

A request to disconnect from a server can be initiated in the following ways:

User-Initiated Disconnect A user can request a disconnection from the server when a client-server transaction completes. A server can also disconnect from a second server when all server-server data transfers have been completed and there is no need for the link to remain connected.

Timer-Initiated Disconnect or Dead Connection Detection Dead Connection Detection is a feature through which Net8 identifies connections that have been left hanging by the abnormal termination of a client.

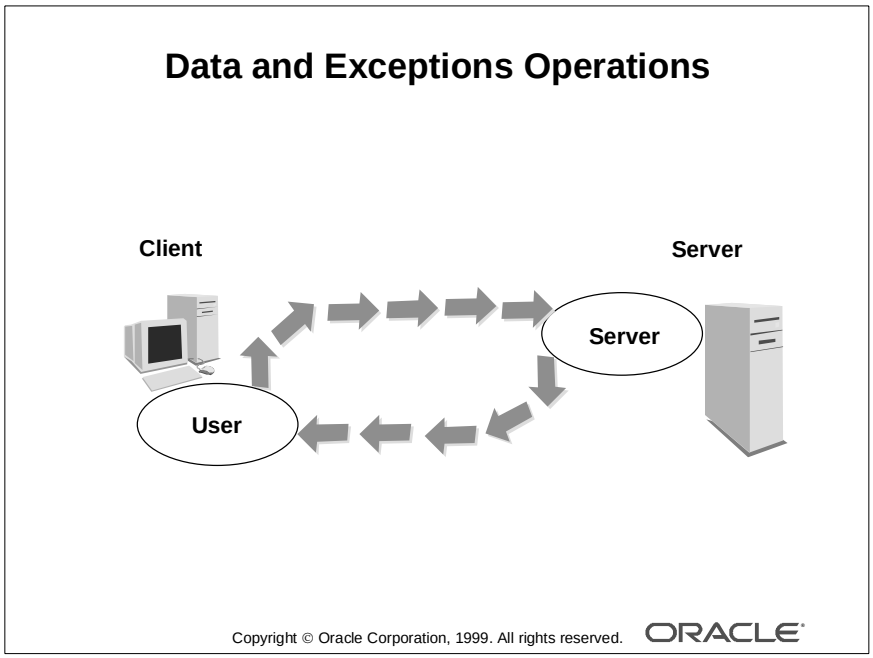
- This situation can occur if a user decides to power down the PC on which the client side application is running without exiting or quitting the application properly.
- On some operating systems, the server side process is left running for a period of time that is operating system–dependent and may vary from a couple of hours to infinity. This can impede the server-side resources.
- On a connection with Dead Connection Detection enabled, a small probe packet is sent from server to client at a user-defined interval (usually several minutes). If the connection is invalid (usually this is due to the client process or machine being unreachable), an error is generated by the send operation, and the server process terminates the connection.

  The SQLNET.EXPIRE_TIME parameter is used to set this feature.

This feature minimizes the waste of resources by invalid connections. It also automatically forces a database rollback of uncommitted transactions and locks held by the user of the broken connection.

Disconnecting from a Server (continued)

Abnormal Connection Termination Other components occasionally disconnect or abort communications without giving notice to Net8. In this event, Net8 recognizes the failure during its next data operation and cleans up client and server operations, effectively disconnecting the current operation. This situation typically results in an `ORA-3113: “end-of-file on communication channel”` error message.



Data Operations

All communication between a client and a server is done synchronously unless a multithreaded server is used.

Exceptions Operations

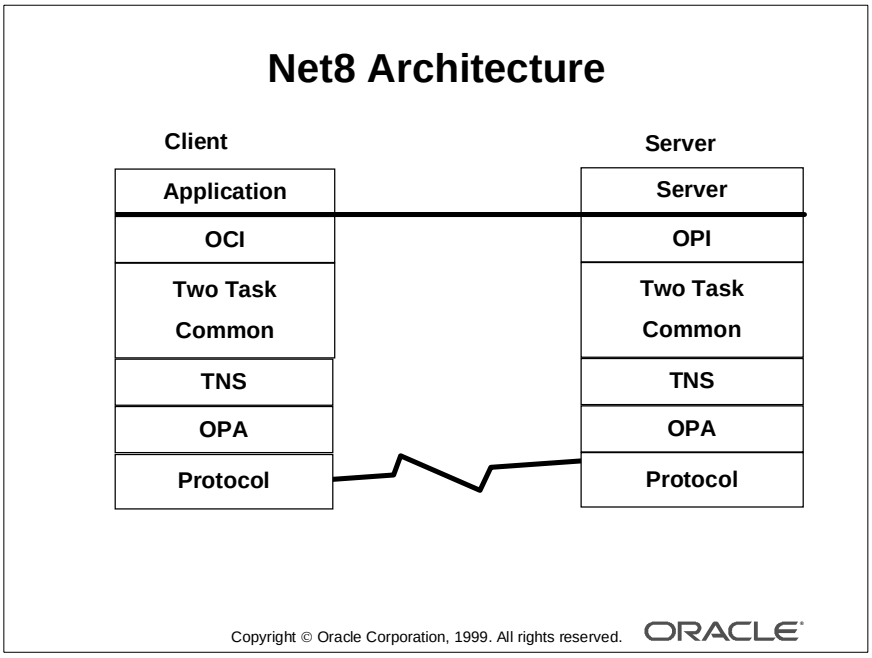
Using Net8, the user can initiate a break over the connection. When the user presses the Interrupt key ([Ctrl] + [C] on some machines), the application causes an exception operation. Additionally, the database can initiate a break to the client if an abnormal operation occurs, such as during an attempt to load a row of invalid data using SQL*Loader.

There are two types of connection breaks: inband breaks and outband breaks:

- Inband breaks are transmitted as part of the regular data traffic between the client and the server using the normal protocol read and write functions. An example of this is an aerospace application in which a satellite must receive all transmitted data in sequence without interruption by a user or application break, unless all the data is received first.
- Outband breaks are faster, because the break message is sent using urgent data messages. An example is being able to hit [Ctrl] + [C] when mistakenly querying a huge table. Rather than having all the data displayed, you can break out of the execution.

The difference between outband breaks and inband breaks, from the server point of view, is that inband breaks use messages that are simply queued, whereas outband breaks use messages that cause signals and therefore take load off the server and can break faster.

The Net8 Stack



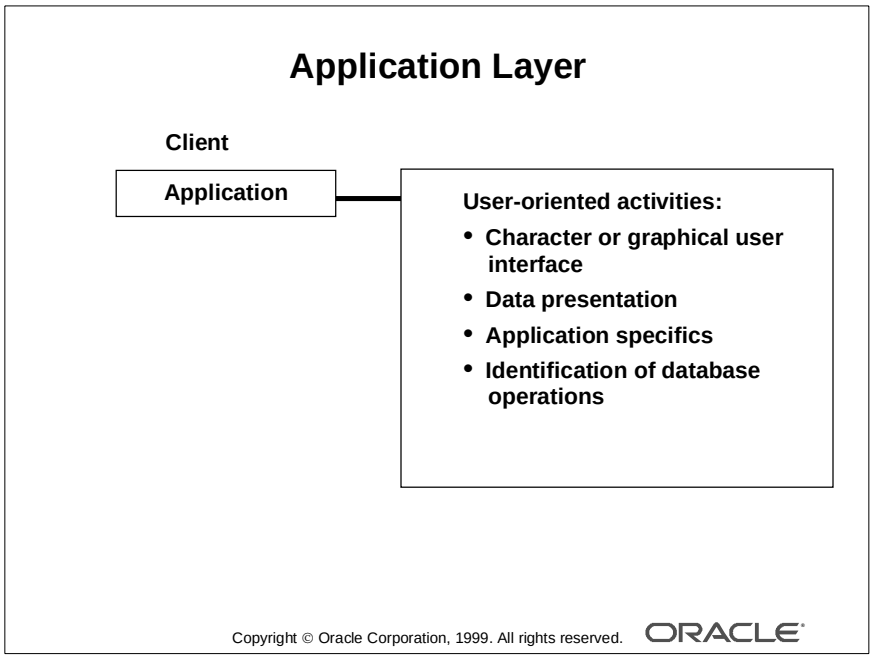
Net8 Layers

Oracle clients and servers use stack communications to share, modify, and manipulate data between themselves.

In an Oracle client-server transaction, information passes through the following layers:

- Application (client)
- Oracle Call Interface (OCI) or Oracle Program Interface (OPI)
- Two Task Common
- Transparent Network Substrate (TNS)
- Oracle protocol adapters (OPA)
- Network-specific protocols

Each layer of the stack is responsible for one or more specific tasks.

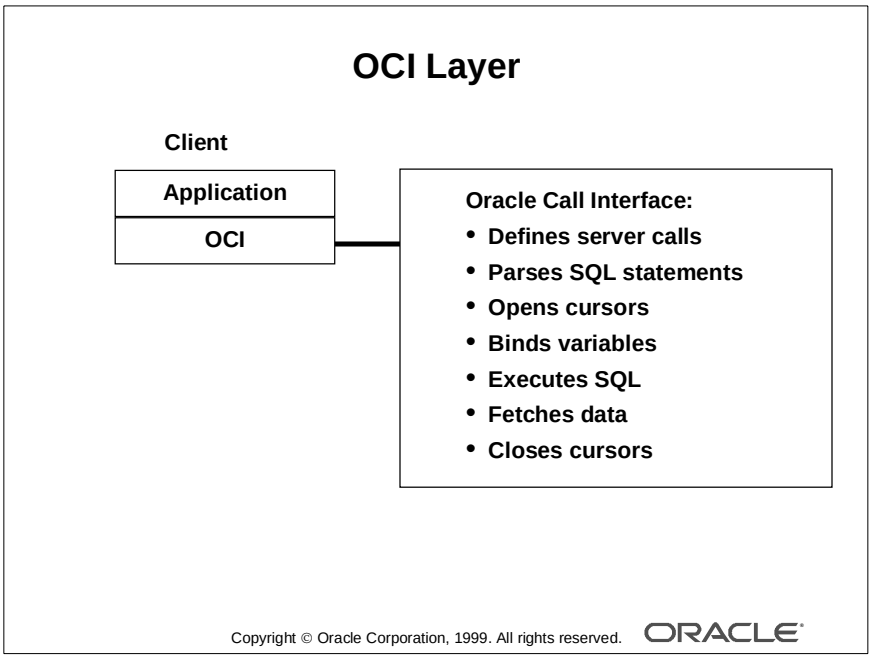


Application Layer

Oracle client applications provide all user-oriented activities, such as character or graphical user interface (GUI) display, screen control, data presentation, application flow, and other application specifics.

An example of a client application could be Oracle Forms or SQL*Plus.

The application identifies database operations to send to the server and passes them through to the Oracle Call Interface (OCI).



Oracle Call Interface Layer

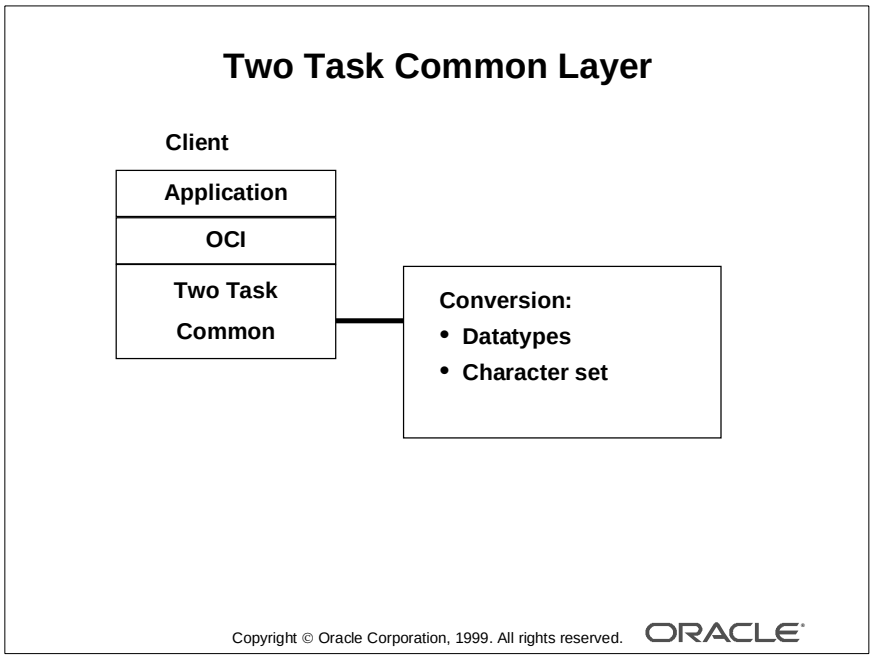
The OCI code contains all the information required to initiate a SQL dialogue between the client and the server. It defines calls to the server to:

- Describe the contents of the fields being returned based on the values in the server's data dictionary
- Parse SQL statements for syntax validation
- Open a cursor for the SQL statement
- Bind client application variables into the server shared memory
- Execute SQL statements within the cursor memory space
- Fetch one or more rows of data into the client application
- Close the cursor

The OCI is also referred to as the OPI, or Oracle Program Interface, on the server. The client application uses a combination of these calls to request activity within the server. OCI calls can be combined into a single message to the server, or they can be processed one at a time through multiple messages to the server, depending on the nature of the client application. Oracle products attempt to minimize the number of messages sent to the server by combining many OCI calls into a single message to the server. When a call is sent, control is passed to Net8 to establish the connection and transmit the request to the server.

Technical Note

The OCI is fully documented, so anyone who wants to write applications using OCI can do so.



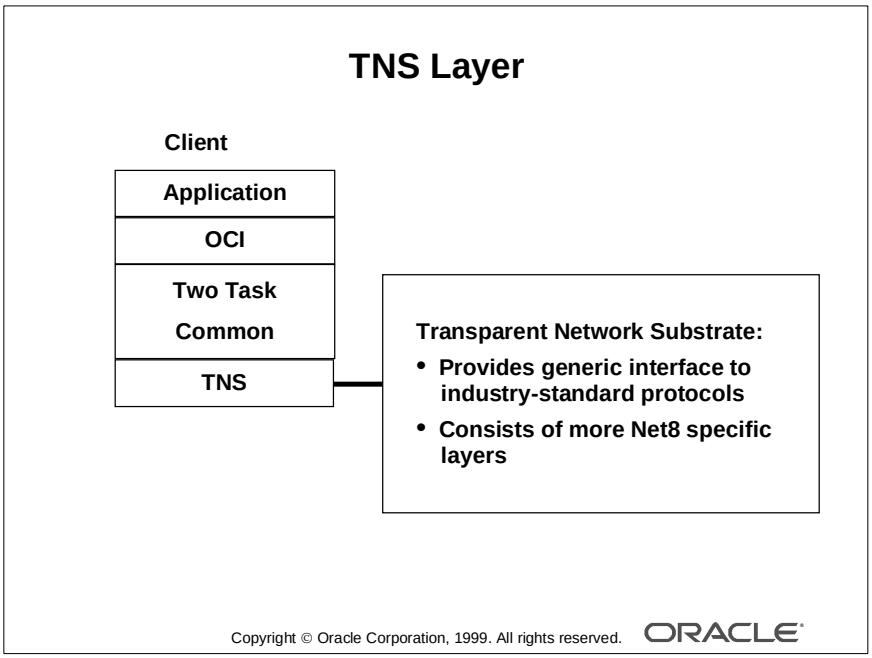
Two Task Common Layer

The Two Task Common layer provides character set and data type conversion between different character sets or formats on the client and server. This layer is optimized to perform conversion only when required on a per-connection basis.

During the initial connection, the Two Task Common layer is responsible for evaluating differences in internal data and character set representations and determining whether conversions are required for the two computers to communicate.

Conversion is required in situations in which the client-side character set differs from that of the server side.

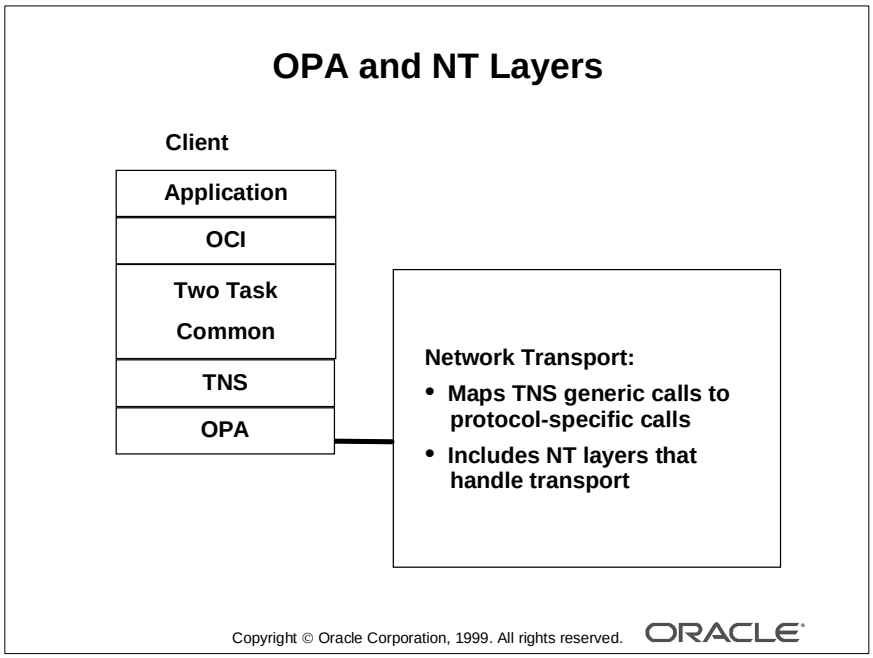
One example of this situation is a client running the WE8ISO8859P1 character set while the server is running the US7ASCII character set. In this case, the Two Task Common layer acts as a translator, helping to convert the values of one character set to the other.



Transparent Network Substrate Layer

The Transparent Network Substrate (TNS) layer is an underlying layer of Net8 providing a common interface to industry-standard protocols. TNS receives requests from Two Task Common and settles all generic machine-level connectivity issues, such as the location of the server or destination (open, close functions), whether one or more protocols are involved in the connection (open, close functions), and how to handle interrupts between client and server based on the capabilities of each (send, receive functions). The generic set of TNS functions (open, close, send, receive) passes control to an Oracle Protocol Adapter to make a protocol-specific call. Additionally, TNS supports encryption and sequenced cryptographic message digests to protect data in transit.

The TNS layer itself consists of more Net8-specific layers, each handling its specific functions of the TNS. These are not covered in further detail.

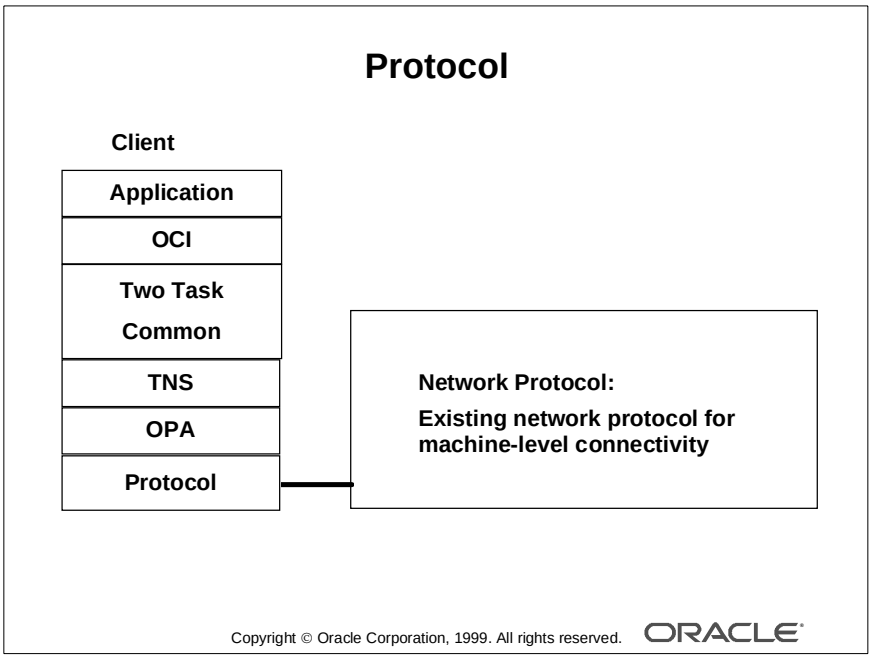


Oracle Protocol Adapter Layer

Oracle protocol adapters (OPA) are responsible for mapping TNS functions to industry-standard protocols used in the client-server connection. Each adapter is responsible for mapping the equivalent functions between TNS and a specific protocol.

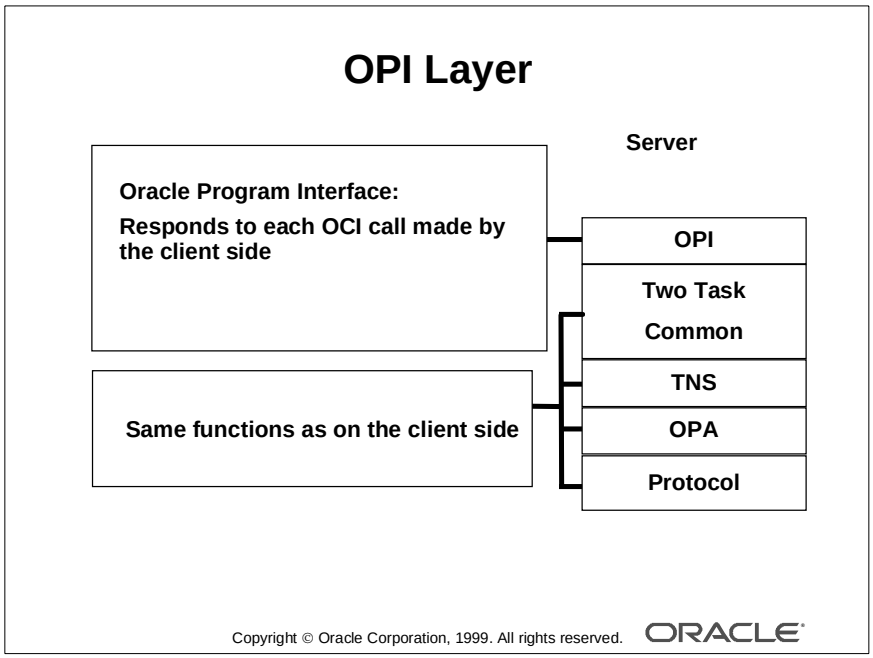
Network Protocols

The Net8 architecture provides support for a broad range of networking protocols, including TCP/IP, Novell SPX/IPX, IBM LU6.2, DECnet, and OSI. By using Net8 protocol adapters, companies can maintain their existing infrastructure without expensive changes. Additionally, Net8 supports multiple protocols on a single machine simply through installation of multiple protocol adapters.



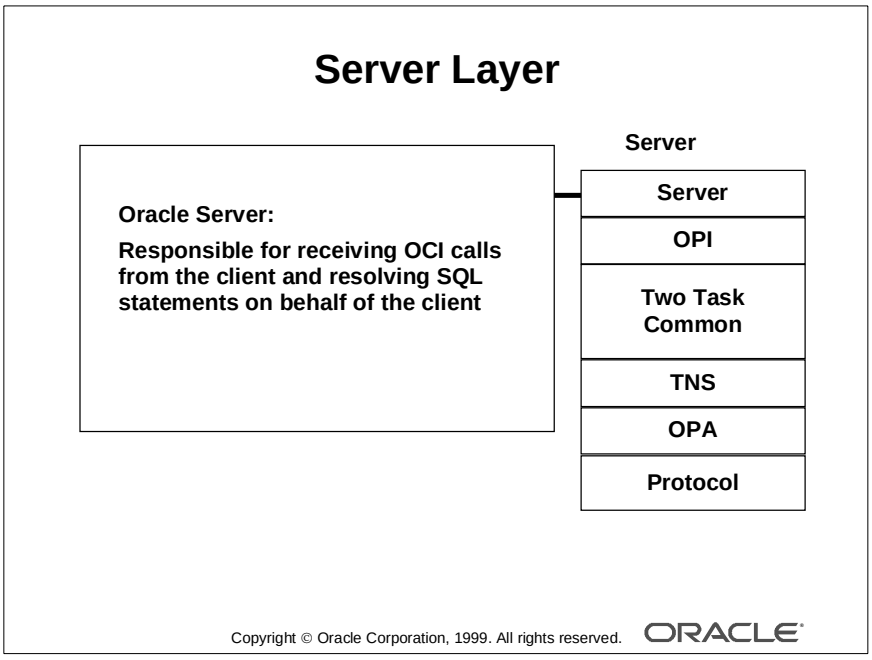
Network-Specific Protocol

All Oracle software in the client-server connection process requires an existing network protocol stack to make the machine-level connection between the two machines. The network protocol is responsible only for transmitting the data from the client machine to the server machine, at which point the data is passed to the server-side protocol adapter.



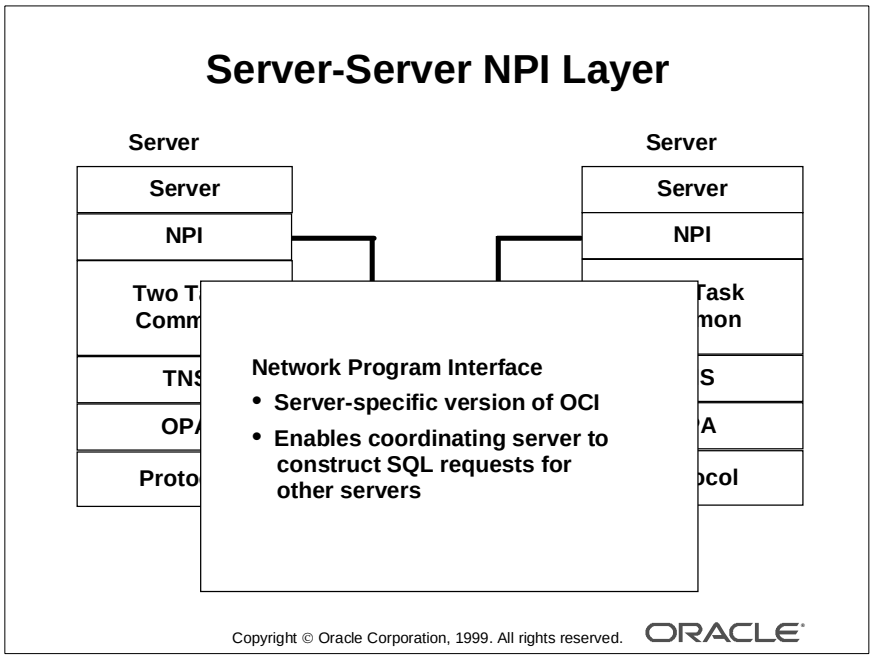
Oracle Program Interface Layer

The OPI performs a complementary function to that of the OCI. It is responsible for responding to each of the possible messages sent by the OCI. For example, an OCI request to fetch 25 rows would have an OPI response to return the 25 rows after they have been fetched.



Server Layer

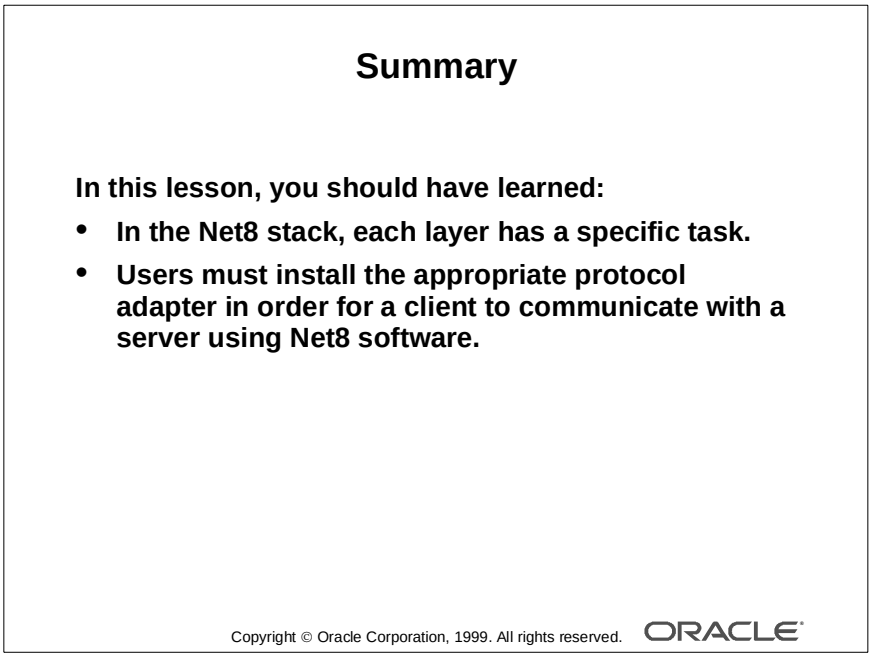
The Oracle server side of the connection is responsible for receiving requests from the client OCI code and resolving SQL statements on behalf of the client application. Once received, a request is processed and the resulting data is passed to the OPI for responses to be formatted and returned to the client application.



Network Program Interface Layer

When two servers communicate to complete a distributed transaction, the process and layers are the same as in the client-server scenario, except that there is no client application. The server has its own version of OCI, called the Network Program Interface (NPI). The NPI performs all of the functions for the server that the OCI does for clients, enabling a coordinating server to construct SQL requests for additional servers.

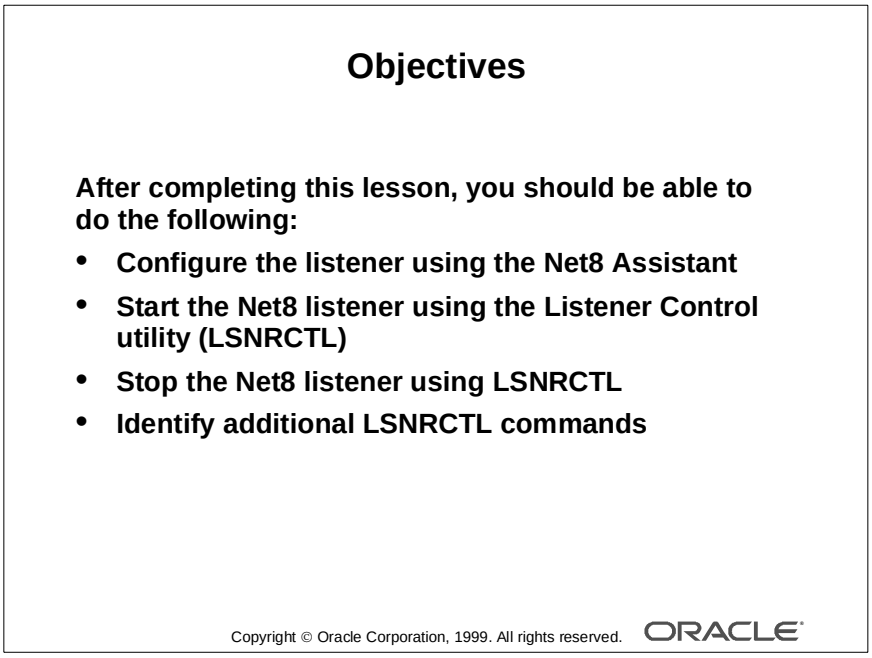
Summary


Summary
In this lesson, you should have learned:
• In the Net8 stack, each layer has a specific task.
• Users must install the appropriate protocol adapter in order for a client to communicate with a server using Net8 software.
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

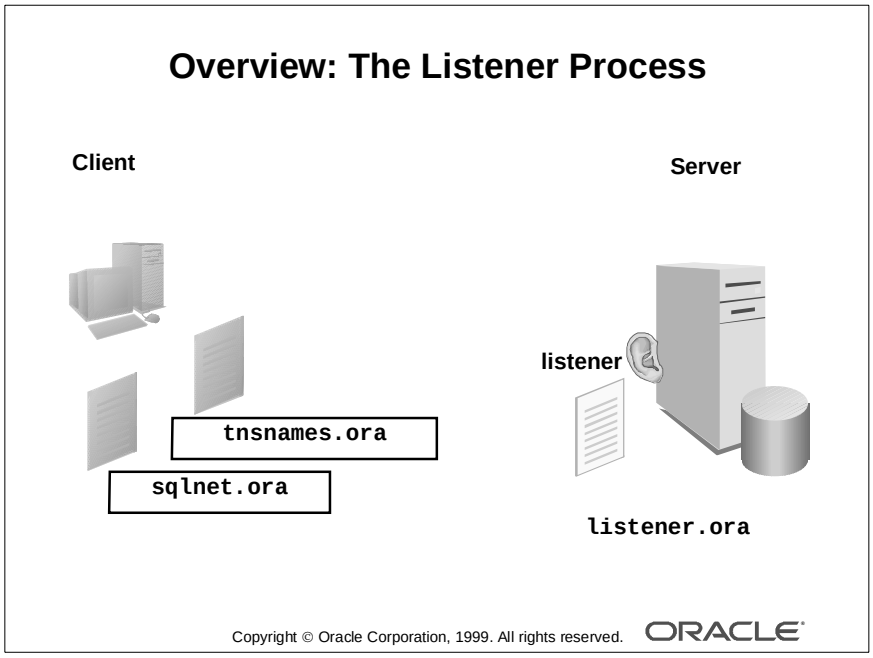
3

Basic Net8 Server-Side Configuration

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Configure the listener using the Net8 Assistant
• Start the Net8 listener using the Listener Control utility (LSNRCTL)
• Stop the Net8 listener using LSNRCTL
• Identify additional LSNRCTL commands
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Overview: The Listener Process

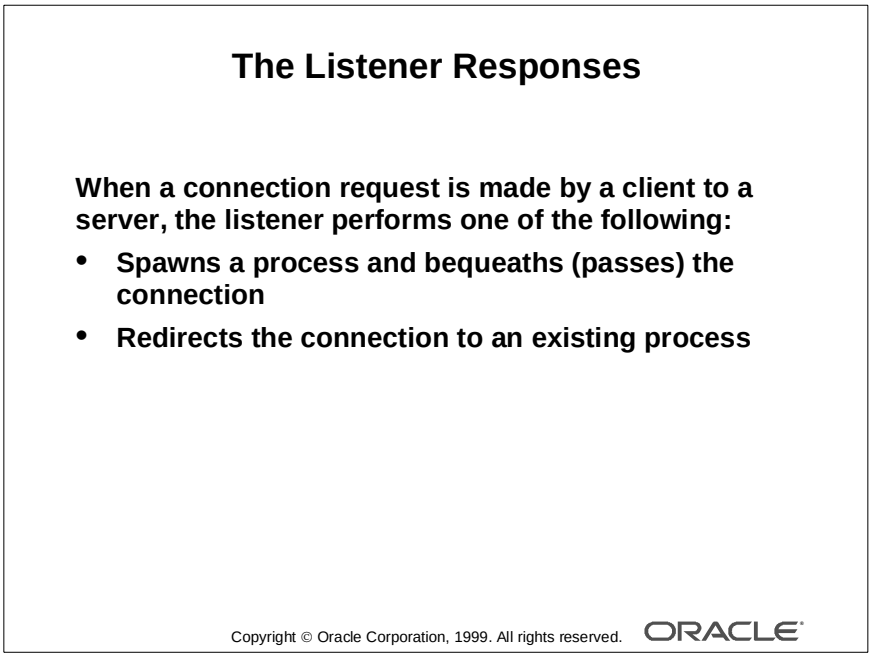


Characteristics of the Listener Process

The listener is a process running on a node that listens for incoming connections on behalf of a database or a number of databases. The following are the characteristics of a listener:

- A listener process can listen for more than one database.
- Multiple listeners can listen on behalf of a single database to perform load balancing.
- The listener can listen on multiple protocols.
- The default name of the listener in Net8 is LISTENER.
- The name of the listener must be unique on the machine on which it resides, and per `listener.ora` file.

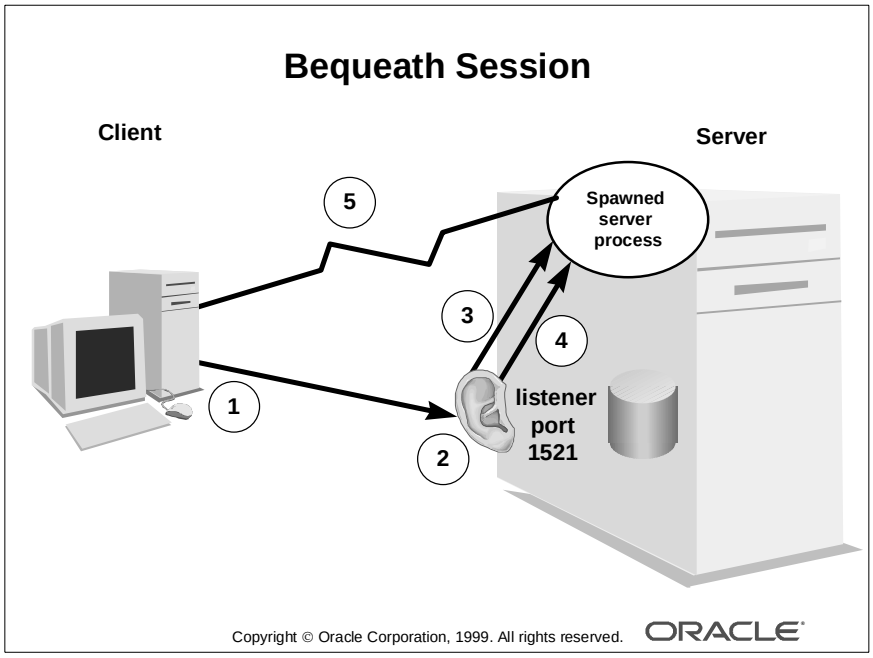
The Listener Responses



Transparency of Bequeath and Redirect

Whether a connection uses a Bequeath session or a Redirect session is transparent to the user. It can be detected only by turning on tracing and analyzing the resulting trace file.

Bequeath Session



Bequeath Session

The listener may create or spawn and bequeath dedicated server processes as connection requests are received.

Steps in a Bequeath Session

When the listener spawns a dedicated server process and bequeaths (passes) the connection to the dedicated server process, the session is called a *Bequeath session*. The following sequence of events occurs:

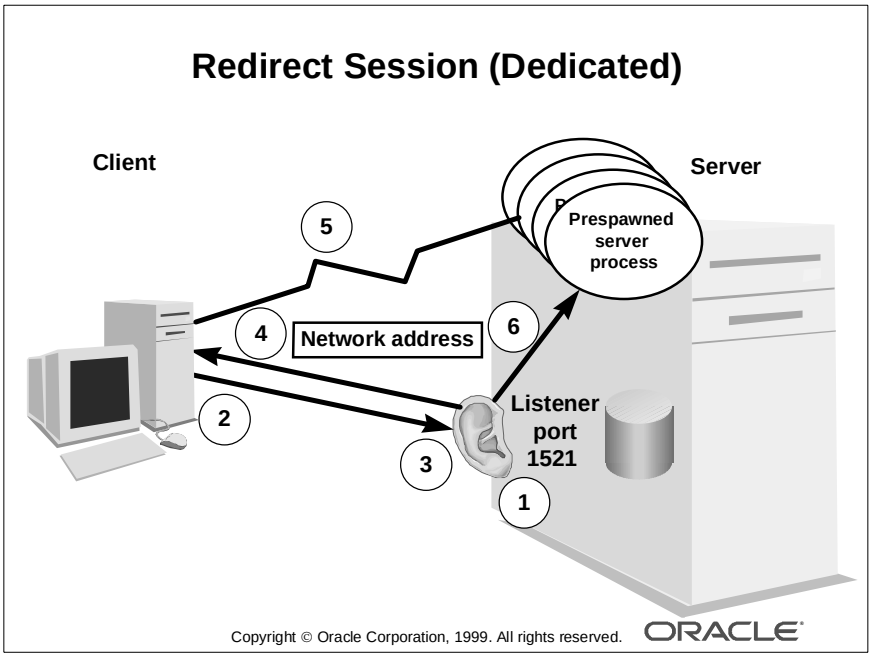
1. A client connects to the listener with the network address.
2. The listener receives the session request and determines if the client's request may be serviced. If not, the listener refuses the session and then continues listening.
3. The listener spawns a new dedicated server process to serve the incoming session.
4. The listener bequeaths the session to that server process.
5. Once the session is established, data flows directly between the client and dedicated server process.
6. The listener continues listening for incoming sessions.

Note: When a client disconnects, the dedicated server process associated with the client closes.

Using the Bequeath Method

- If a dedicated server does not have prespawned server processes, the Bequeath session method is used by default.
- If, because of the operating system or protocol, a connection cannot pass between two different processes on the same machine, this method cannot be used.

Redirect Session (Dedicated)



Redirect Session (Prespawned)

The use of prespawned dedicated server processes requires specification in a listener configuration file (`listener.ora`).

Steps in a Redirect Session (Prespawned Dedicated Server Process)

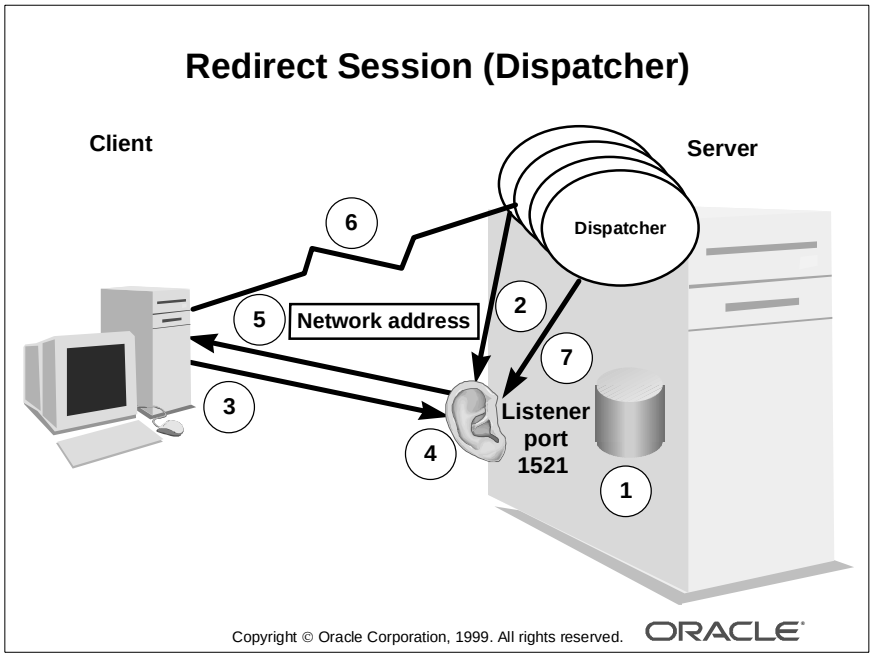
In a prespawned dedicated server, the listener redirects the session to a prespawned server process, rather than spawns the process as in the bequeathed connection.

1. The listener spawns a series of dedicated server processes when it is started.

   Note: The maximum number of server processes are defined by the PRESPAWN_MAX parameter in the listener configuration file (`listener.ora`). The listener stops spawning additional servers once the PRESPAWN_MAX has reached. The listener maintains a log of the number of prespawned servers and the servers that are in use by various clients.
2. A client connects to the listener with the network address.
3. The listener receives the session request and determines if the client's request may be serviced. If not, the listener refuses the session and then continues listening.
4. The listener issues a Redirect message to the client containing the network address of one of the prespawned servers.
5. The client disconnects from the listener and connects to the prespawned address given by the listener.
6. Once the session is established, the listener spawns another server to replace the one used by the client.
7. The listener continues listening for incoming sessions.

Prespawned servers take up system resources, but the connection time from the client to the server process is faster than in the Bequeath method.

Redirect Session (Dispatcher)



Redirect Session (Dispatcher)

When an Oracle server has been configured as a multithreaded server, incoming sessions are always redirected by the listener to the Dispatcher process unless either the session specifically requests a dedicated server or no dispatchers are available.

Steps in a Redirect Session (Dispatcher Server Process)

In a multithreaded server, the listener redirects the session to a dispatcher process, rather than redirects the connection to a prespawned server process.

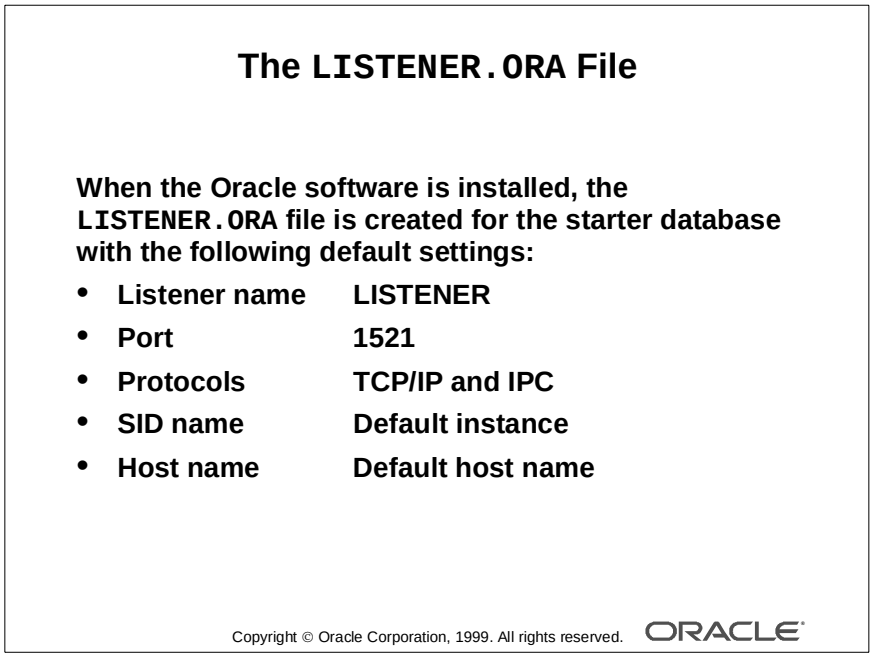
1. When a database instance is started, dispatchers are started according to the parameters defined in the configuration file (`init.ora`).
2. At startup, the address of each dispatcher is then registered with the listener.

   Note: The listener, therefore, has knowledge of the usage of the different dispatchers available.
3. A client connects to the listener with the network address.
4. The listener receives the session request and determines if the client's request can be serviced. If not, the listener refuses the session and then continues listening.
5. If the request is processed, the listener issues a redirect message to the client containing the network address of the least-used dispatcher.
6. The client disconnects from the listener and connects to the dispatcher address given by the listener.
7. The dispatcher updates the listener with the new load value. The listener continues listening for incoming sessions.

Using of a Redirect Method (Dispatcher Server Process)

With this method, many clients can connect to the same dispatcher without the need to spawn a server process or to have dedicated server processes prespawned. When a multithreaded server is configured, incoming sessions are always routed to a dispatcher.

The LISTENER.ORA File

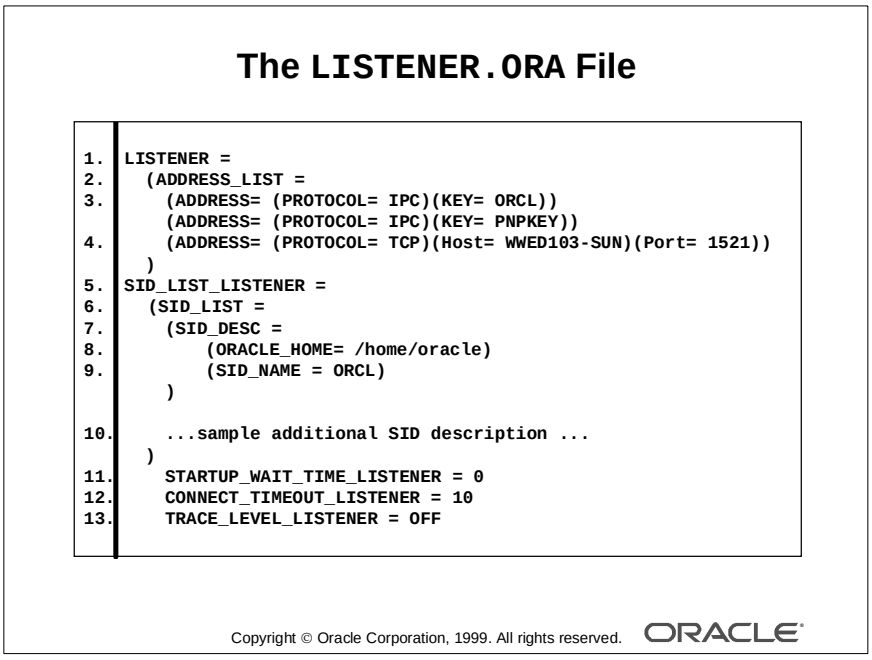


The LISTENER.ORA File

The `listener.ora` file is used to configure the listener. The `listener.ora` file must reside on the machine or node on which the listener is to reside.

The `listener.ora` file contains configuration information for the following:

- The listener name
- The listener address
- Databases that use the listener
- Listener parameters



LISTENER.ORA File Contents

The default `listener.ora` file contains the following parameters:

1. The name of the listener. The default name is LISTENER.
2. The ADDRESS_LIST parameter contains a block of addresses at which the listener listens for incoming connections. Each of the addresses defined in this block represents a different way by which a listener receives a connection.
3. IPC addresses identify both incoming connection requests from applications on the same node as the listener and information sent or registered by a database dispatcher. If the IPS addresses identify connection requests from the same node, the KEY value is equal to the service name of the database. If the addresses identify a database dispatcher, the KEY value is equal to the database system identifier (SID). If the service name is the same as the SID, only one IPC address is needed.
4. The TCP address identifies incoming TCP connections from clients on the network attempting to connect to port 1521. The clients use the port defined in their `tnsnames.ora` file to connect to this listener. Based on the SID_LIST defined for this listener, the listener specifies the database to which to connect.

LISTENER.ORA File Contents (continued)

5 A listener can listen for more than one database on a machine. The SID_LIST_listener_name block or parameter is where these SIDs are defined.

6 The SID_LIST parameter is defined if more than one SID is defined.

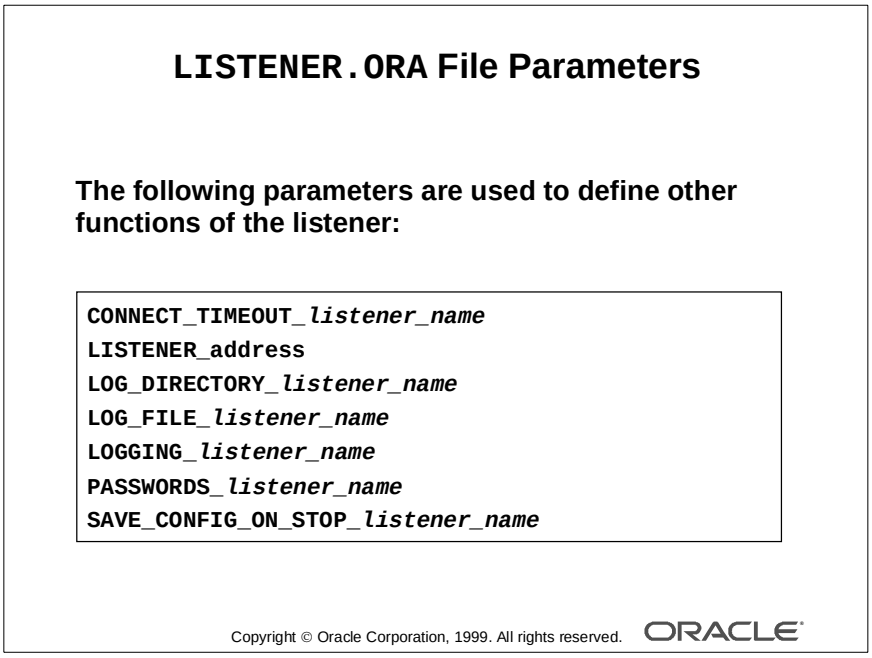
7 The SID_DESC parameter must exist for each defined SID.

8 The ORACLE_HOME is where the home directory of the database is defined. This enables the listener to identify the location of a database executable file.

9 The SID_NAME parameter defines the name of the SID on behalf of which the listener accepts connections.

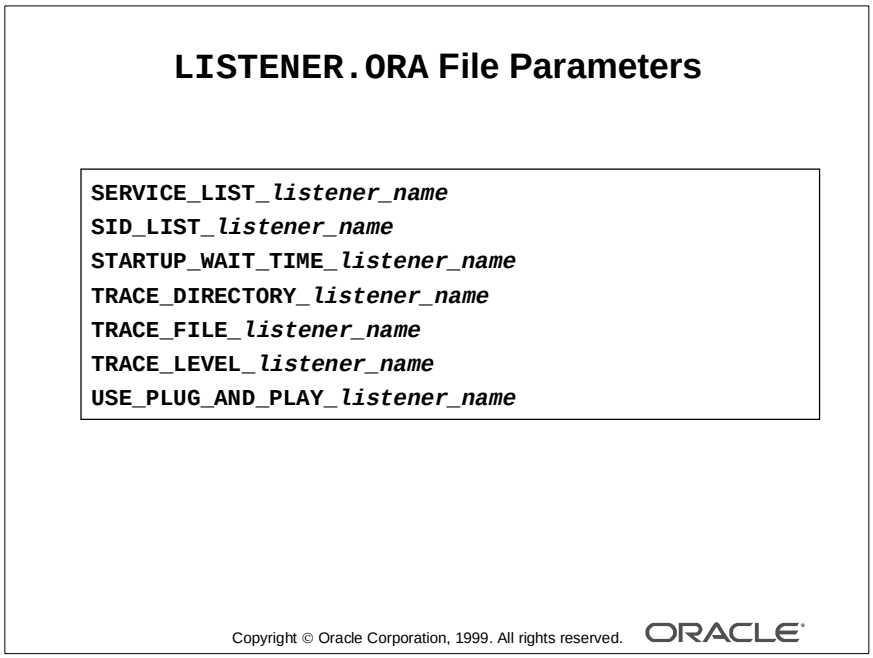
10 By default, an example SID is defined here.

LISTENER.ORA File Parameters



LISTENER.ORA Parameters

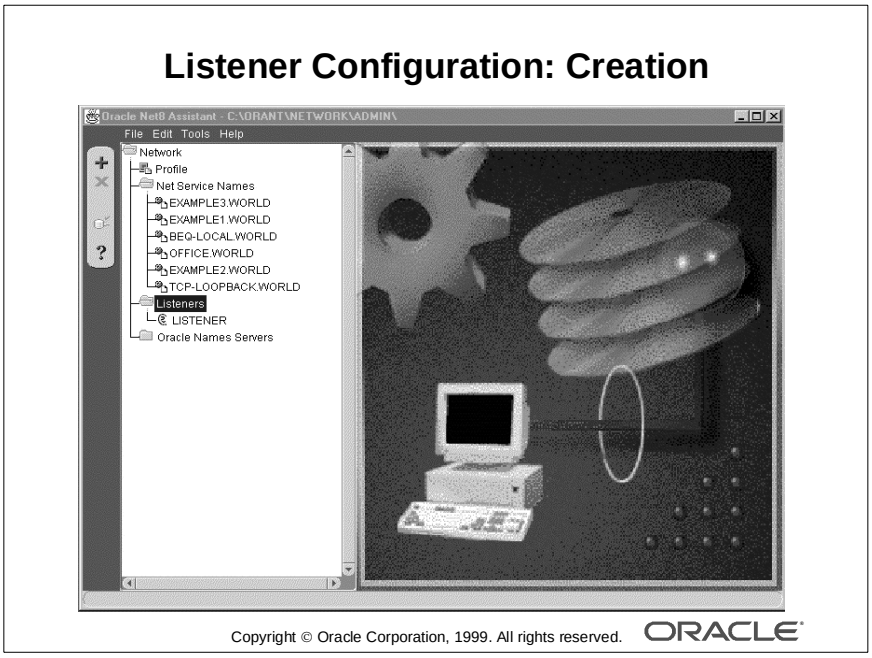
Parameter	Description
CONNECT_TIMEOUT_*listener_name*	Sets the number of seconds that the listener waits for the server process to get a valid database query after the session has started.
LISTENER_address	Defines the listening addresses for the listener.
LOG_DIRECTORY_*listener_name*	Controls the directory in which the log file is written.
LOG_FILE_*listener_name*	Specifies the filename to which the log information is written.
LOGGING_*listener_name*	By default, logging is always on unless you provide this parameter and turn logging off.
PASSWORDS_*listener_name*	Sets a nonencrypted password for authentication to the Listener Control utility (LSNRCTL).
SAVE_CONFIG_ON_STOP_*listener_name*	Any changes made by the LSNRCTL SET command are made permanent if the parameter is set to TRUE.



LISTENER.ORA Parameters

Parameter	Description
SERVICE_LIST_*listener_name*	Defines the service served by the listener. This is the same as the SID_LIST, made more generic for nondatabase servers.
SID_LIST_*listener_name*	Defines the SID of the databases served by the listener.
STARTUP_WAIT_TIME_*listener_name*	Sets the number of seconds that the listener sleeps before responding to the first LSNRCTL STATUS command. This assures that a listener with a slow protocol has time to start up before responding to a status request.
TRACE_DIRECTORY_*listener_name*	Controls the directory in which the trace file is written.
TRACE_FILE_*listener_name*	Sets the name of the trace file.
TRACE_LEVEL_*listener_name*	Turns tracing off or to a specified level.
USE_PLUG_AND_PLAY_*listener_name*	Instructs the listener to register with a well-known Names server. Continues to look for a well-known Names server until one is found.

Listener Configuration: Creation

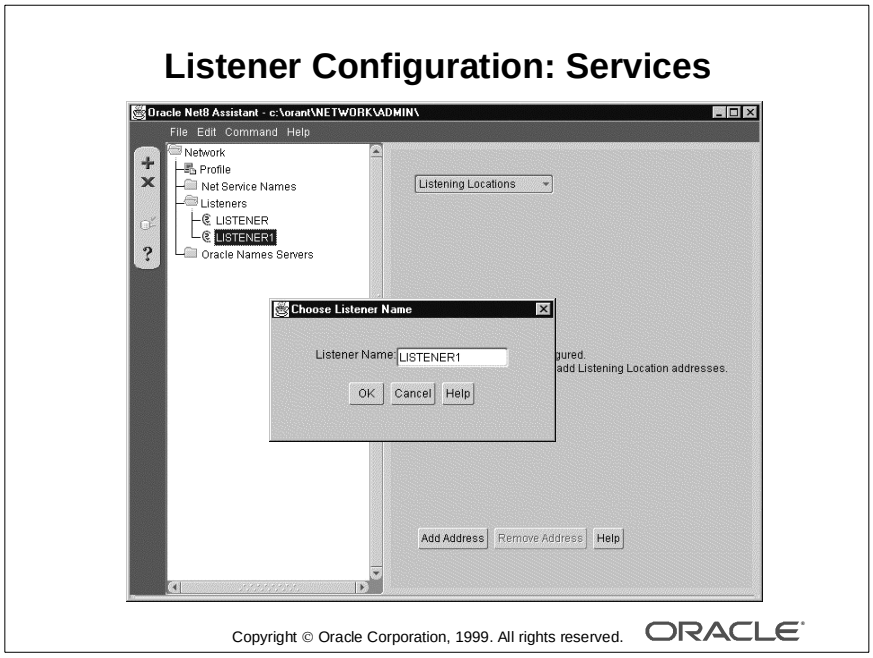


Creating the Listener

1. Start up Net8 Assistant.
2. Click the Listeners icon.
3. Select Create from the Edit menu.
4. Enter a listener name in the Listener Name field on the dialog box that appears.
5. Select Listening Locations from the pull-down menu within Net8 Assistant for your listener.
6. Click the Add Address button.
7. Change or enter information in the Protocol, the Host, and Port fields as necessary.

   Note: The fields may vary according to the protocol used. The preceding step represents a listener listening on a TCP port. Due to a bug in 8.0.4, SID names with uppercase characters in the first character are not accepted. Press [Shift]-*Uppercase Character* a couple of times.
8. Select Net8 Clients in the Protocol Stack Support field.
9. Select Save Network Configuration from the File menu of Net8 Assistant.

Listener Configuration: Services



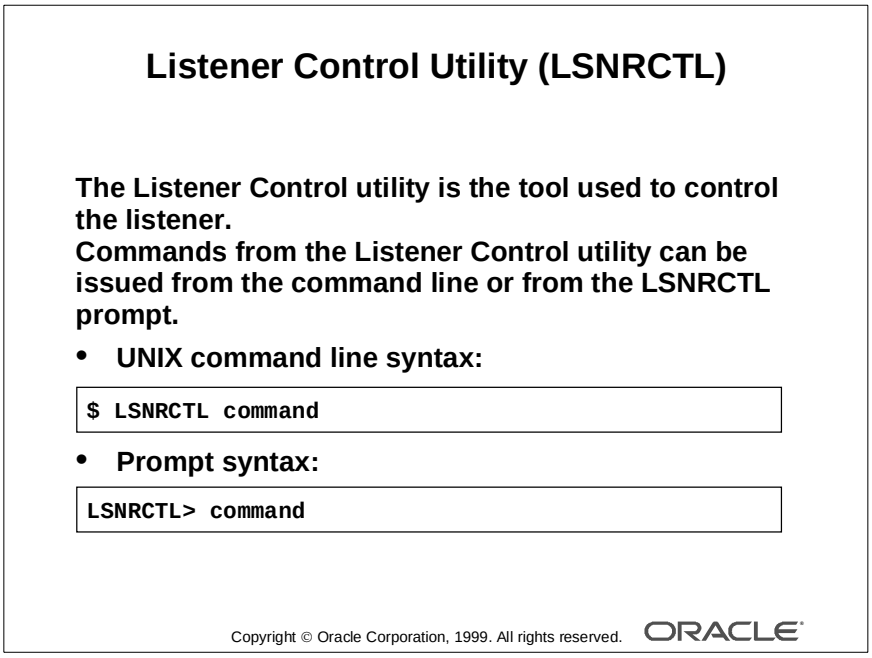
Configuring Database Services

1. Select Database Services from the pull-down menu within Net8 Assistant for your listener.
2. Click the Add Database button.
3. Enter the global database name, the Oracle home directory, and the SID in the appropriate fields.
4. Select whether or not to use prespawned dedicated servers.
5. If prespawned servers are to be used, select the option button for it, and click the Configure Prespawned Servers button.
6. If the prespawned servers are configured, a dialog box appears.
7. Set the number of prespawned servers and their timeout values for the supported protocols appropriately, and click OK.
8. Select Save Network Configuration from the File menu of Net8 Assistant.

Configuring Listener Logging and Tracing

1. Select General Parameters from the pull-down menu within Net8 Assistant for your listener.
2. Click the Logging & Tracing tab.
3. Enable logging by selecting the Logging Enabled option button.
4. Enter the path and filename for a log file.
5. Select Save Network Configuration from the File menu of Net8 Assistant.

Listener Control Utility (LSNRCTL)



Windows NT Platform Command Line Syntax

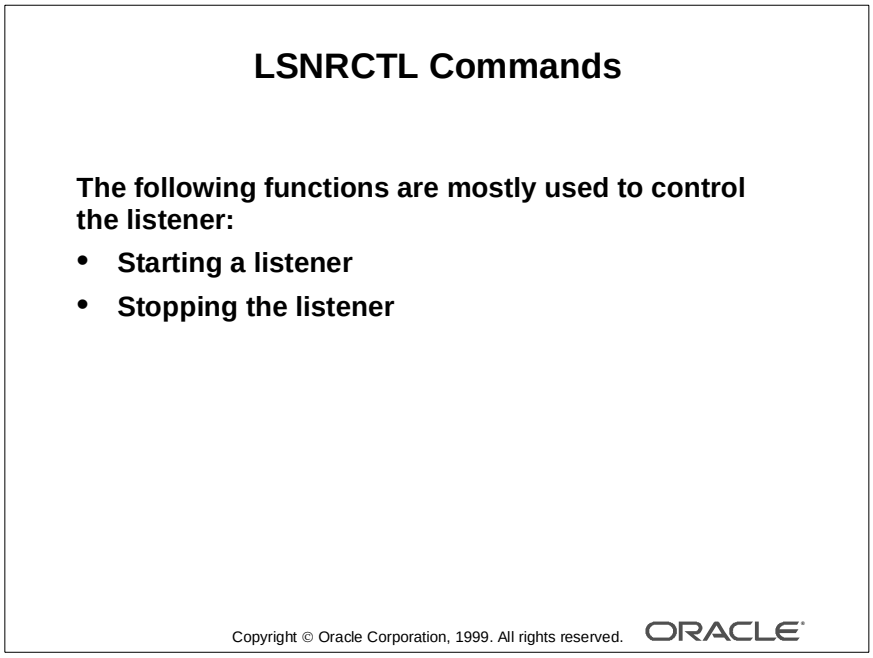
On the Windows NT operating system, use the following command to start the Listener Control utility:

```
C:\> LSNRCTL80 command
```

On Oracle8*i* for Windows NT, the Listener Control utility is started by issuing the following command:

```
C:\> LSNRCTL command
```

LSNRCTL Commands



Starting the Listener

You can use the START command to start the listener from the Listener Control utility. Any manual changes to the `listener.ora` file must be made when the listener is shut down. The argument for the START command is the name of the listener, and if no argument is specified, the current listener is started. If a current listener is not defined, LISTENER is started.

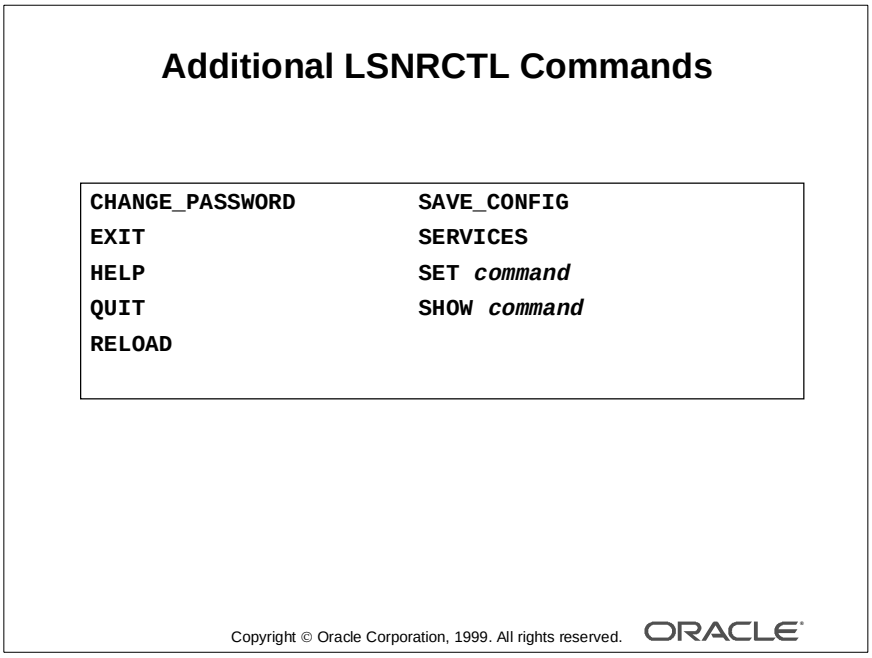
```
LSNRCTL> START [listener_name]
```

Stopping the Listener

The STOP command stops the listener. The listener must be running to stop it properly. If a password is configured, the SET PASSWORD command must be used before the STOP command can be used. The password must be set from within the LSNRCTL prompt; it cannot be set from the operating system command line. It is good practice to send a warning message to all network users before stopping a listener.

```
LSNRCTL> STOP [listener_name]
```

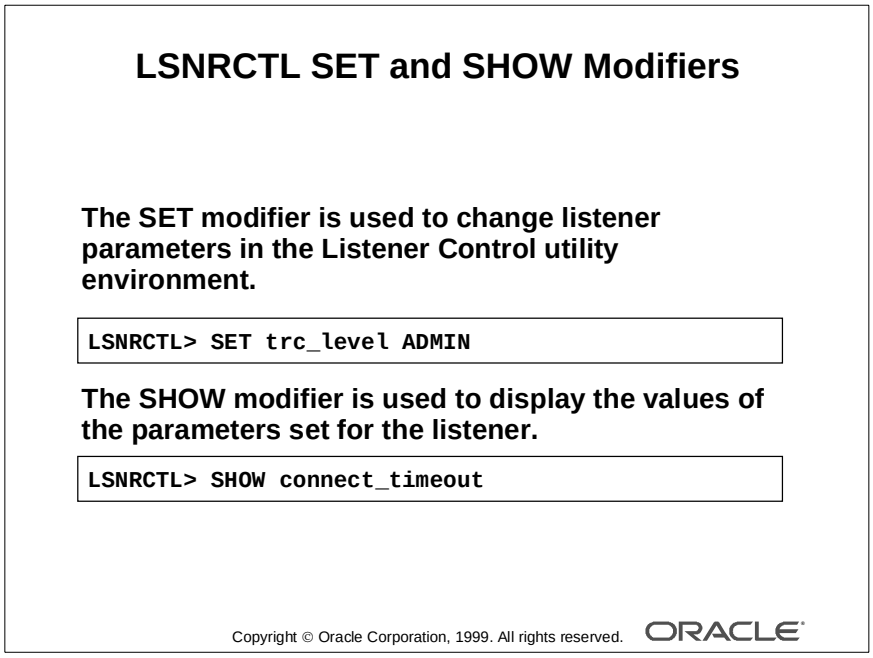
Additional LSNRCTL Commands



Other LSNRCTL Commands

Command	Description
CHANGE_PASSWORD	Dynamically changes the encrypted password of a listener.
EXIT	Quits the LSNRCTL utility.
HELP	Provides the list of all available LSNRCTL commands.
QUIT	Provides the functionality of the EXIT command.
RELOAD	Shuts down everything except listener addresses and rereads the `listener.ora` file. You use this command to add or change services without actually stopping the listener.
SAVE_CONFIG	Creates a backup of your listener configuration file (called `listener.bak`) and updates the `listener.ora` file itself to reflect any changes.
SERVICES	Provides detailed information about the services the listener listens for.
SET *parameter*	This command sets a listener parameter.
SHOW *parameter*	This command lists the value of a listener parameter.

LSNRCTL SET and SHOW Modifiers

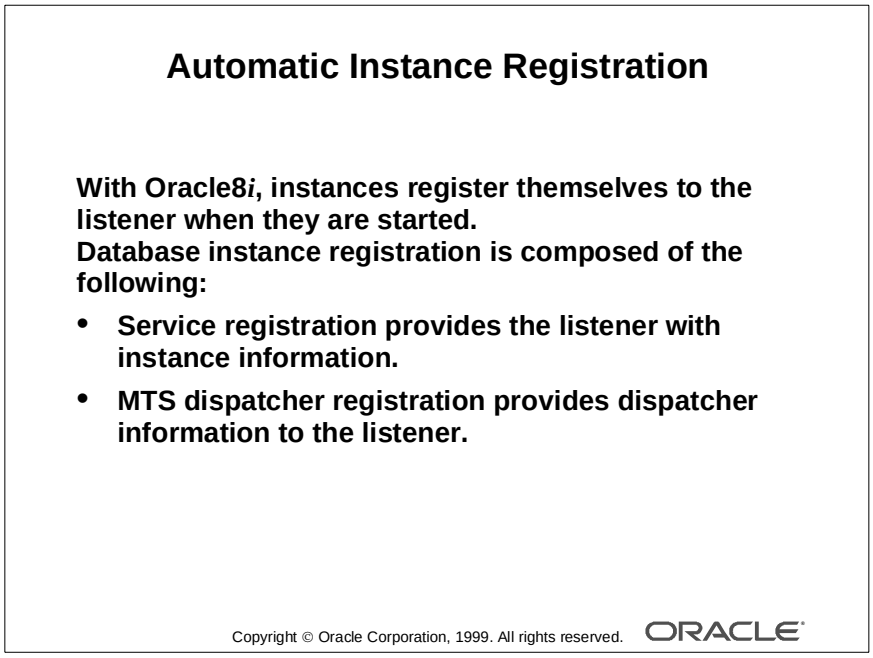


SET and SHOW Modifiers

Command	Description
SET CONNECT_TIMEOUT	Determines the amount of time the listener waits for a valid connection request after a connection has been started.
SET CURRENT_LISTENER	Sets or shows parameters when multiple listeners are used.
SET LOG_DIRECTORY	Sets a nondefault location for the log file or to return the location to the default.
SET LOG_FILE	Sets a nondefault name for the log file.
SET LOG_STATUS	Turns listener logging on or off.
SET PASSWORD	Changes the password sent from the LSNRCTL utility to the listener process for authentication purposes only.
SET SAVE_CONFIG_ON_STOP	Saves any changes made by the LSNRCTL SET command permanently if the parameter is on. All parameters are saved right before the listener exits.
SET STARTUP_WAITTIME	Sets the amount of time the listener sleeps before responding to a START command.
SET TRC_DIRECTORY	Sets a nondefault location for the trace file or to return the location to the default.
SET TRC_FILE	Sets a nondefault name for the trace file.
SET TRC_LEVEL	Turns on tracing for the listener.

Note: The SHOW command has the corresponding parameters of the SET command except SET PASSWORD.

Automatic Instance Registration



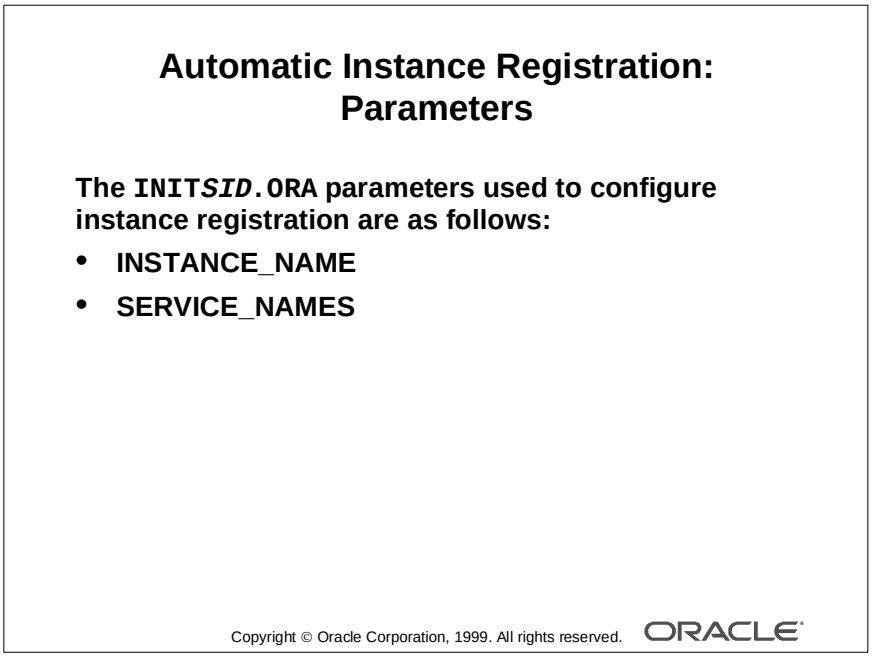
Automatic Instance Registration

Prior to Oracle8*i*, information about the instance had to be manually configured in the `listener.ora` file (SID_LIST_*listener_name* parameter). Now instances register themselves with the listener. By default, the instance registers its information with the listener on the local machine.

Benefits

- The `listener.ora` file does not require the SID_LIST_*listener_name* parameter that specifies information on the databases served by the listener. This parameter is still required if the management tool you are using still requires it.
- Connect-time failover is enabled.
- Connection load balancing is enabled.

Automatic Instance Registration: Parameters



New INIT*SID*.ORA Parameters

With the new initialization parameters, instances can specify their instance name and which services they want to belong to. When an instance is started, initialization parameters are read from the init*SID*.ora file.

The following are examples of the usage of instance registration parameters:

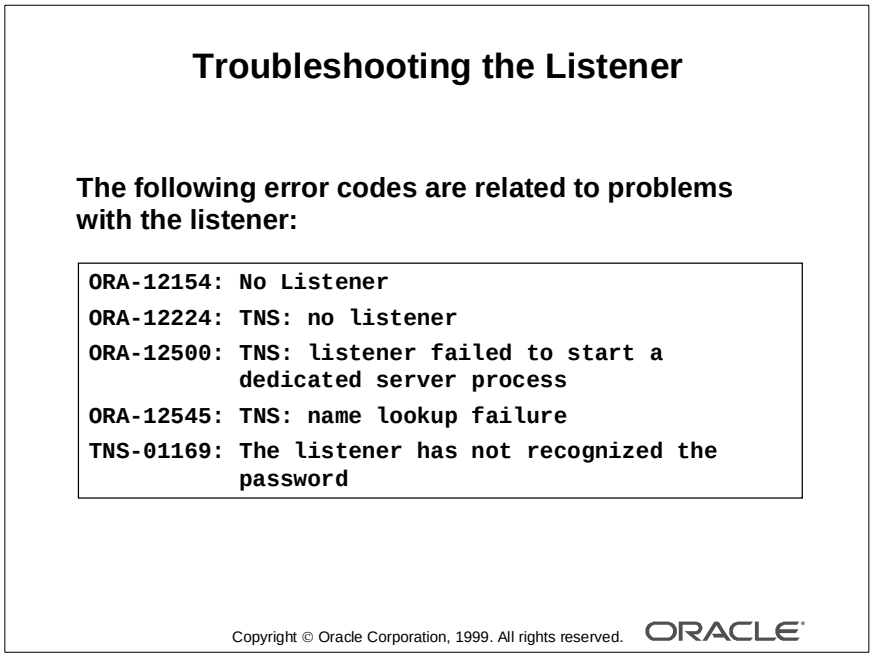
```
instance_name = U01
service_names = U01.us.oracle.com

instance_name = U02
service_names = U02.world
```

Technical Note

The SERVICE_NAMES parameter is comprised of the database name and the domain name. The domain can be obtained from the DB_DOMAIN parameter in the init*SID*.ora file. An example of a service name is OP.COM, where OP is the database name and COM is the domain.

Troubleshooting the Listener



Error ORA-12154

Connection requests that are received too quickly for a listener to handle, which exceed the listener's backlog (determined by QUEUESIZE in the `listener.ora` and `names.ora` files), return this message. To correct this problem, follow these steps:

1. Stop the listener.
2. Reconfigure QUEUESIZE in the `listener.ora` file.
3. Restart the listener and try to connect again.

Note: The default for the QUEUESIZE parameter is 17 and is valid only for TCP/IP and DECnet protocols.

Error ORA-12224

The connection request could not be completed because the listener is running. To correct this problem, follow these steps:

1. Ensure that the supplied destination address on the client matches the one used by the listener.
2. Verify that there is not a version compatibility problem.

Error ORA-12500

The listener could not start a process connecting the user to the database server. To correct this problem, follow these steps:

1. Verify that the SID_LIST section of the `listener.ora` file and the system identifier (SID) in the CONNECT DATA section of the `tnsnames.ora` file are correct.
2. Verify that the user has adequate privileges to access the database.

Error ORA-12545

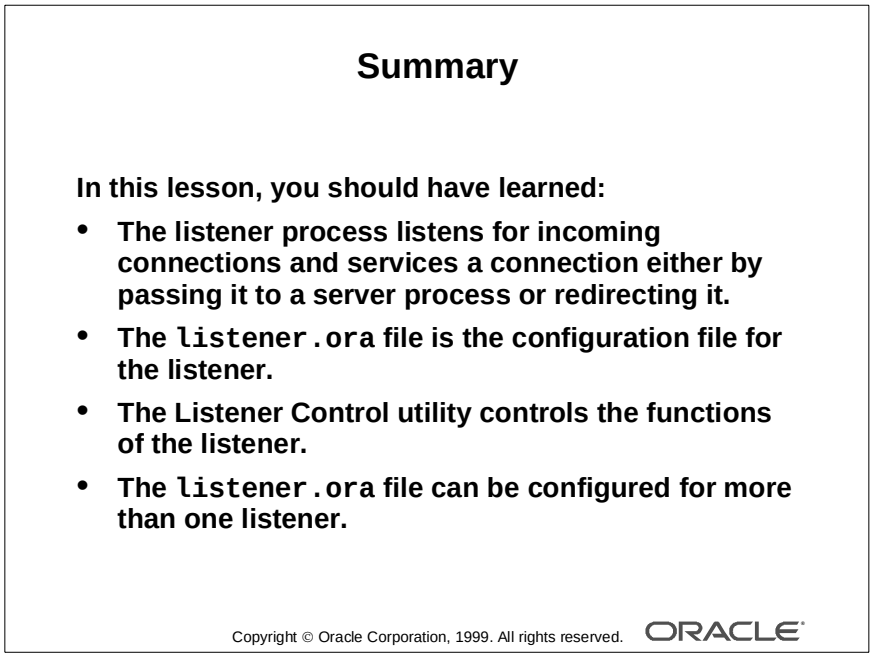
The listener on the remote node cannot be contacted. Follow these steps to correct the problem:

1. Verify that the ADDRESS in the `tnsnames.ora` file or the `listener.ora` file is correct.
2. Verify that the listener on the remote node has been started. You may check its status with the STATUS command of the Listener Control utility and start it with the START command if necessary.

Error TNS-01169

To correct this problem, enter the SET PASSWORD command from LSNRCTL and then the STOP command to stop the listener.

Summary


Summary
In this lesson, you should have learned:
• The listener process listens for incoming connections and services a connection either by passing it to a server process or redirecting it.
• The listener.ora file is the configuration file for the listener.
• The Listener Control utility controls the functions of the listener.
• The listener.ora file can be configured for more than one listener.
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Frequently Asked Questions

The following discussion points are taken from the frequently asked questions regarding topics in this chapter. This information is retrieved and analyzed in cooperation with Oracle Worldwide Support.

Problem

How do you register a unique client identifier with the Net8 listener?

Solution

To register a unique client identifier with the listener during a connection request and to include it in an audit trail, edit the `sqlnet.ora` file. Use any alphanumeric string up to 128 characters long to specify the value of the SQLNET.CLIENT_REGISTRATION parameter in the client's `sqlnet.ora` file. Net8 sets the SQLNET.CLIENT_REGISTRATION to OFF by default.

Problem

When starting a Net8 TNS listener in UNIX, the following error occurs:

```
01151, 00000, "Missing listener name, %s, in
LISTENER.ORA"
// *Cause:  The listener could not find the listener
name specified.
// *Action: Make sure valid addresses on which to
listen are specified // for
the listener name in LISTENER.ORA.
```

Solution

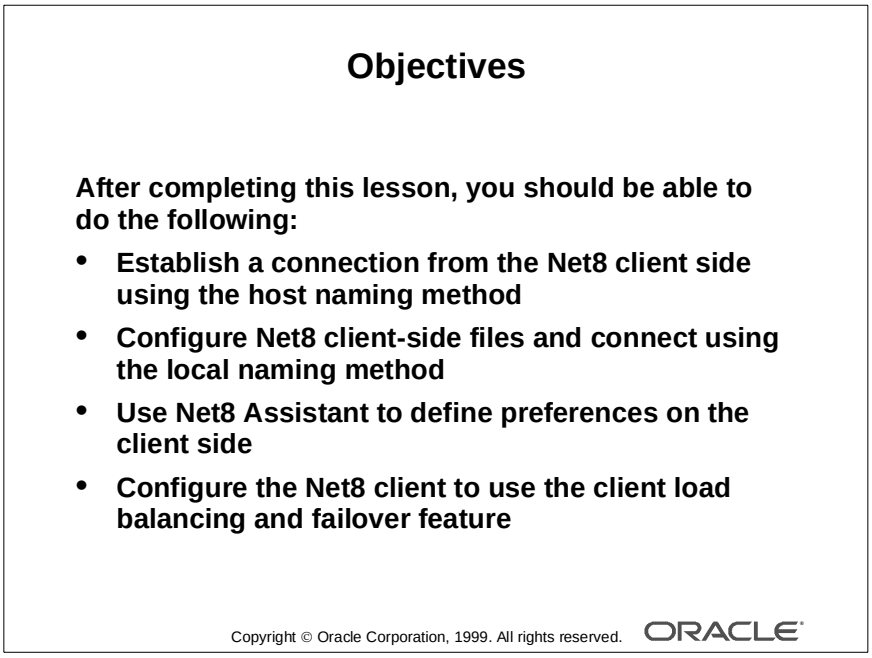
If the `listener.ora` file is correct and it was working before, then check the environment for the Oracle account that starts the listener. Check the environment variables for $ORACLE_HOME, $ORACLE_SID, $PATH, and so on. Even if you set these variables from a script, check them at the UNIX prompt:

```
env | more
```

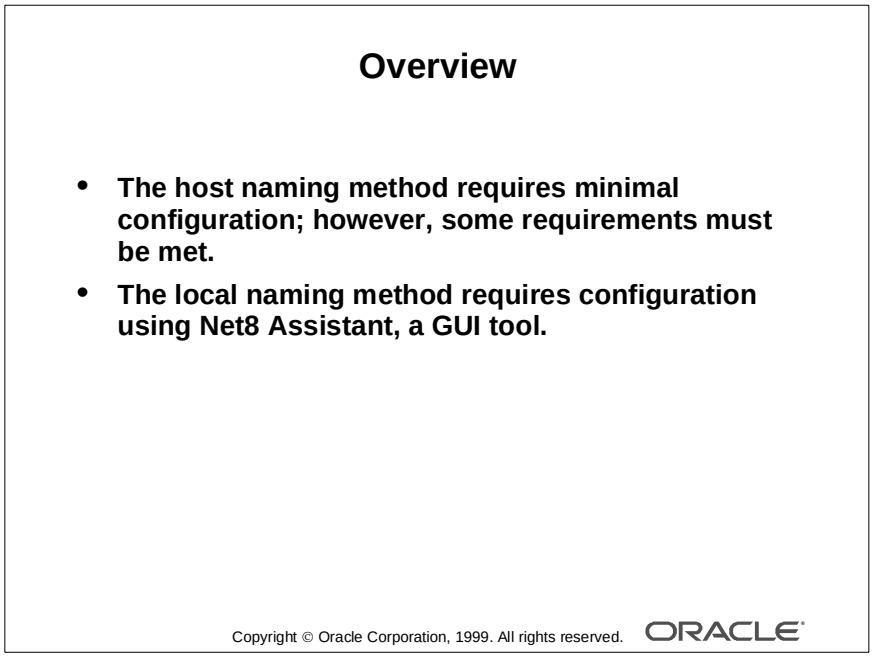
4

Basic Net8 Client-Side Configuration

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Establish a connection from the Net8 client side using the host naming method
• Configure Net8 client-side files and connect using the local naming method
• Use Net8 Assistant to define preferences on the client side
• Configure the Net8 client to use the client load balancing and failover feature
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Overview



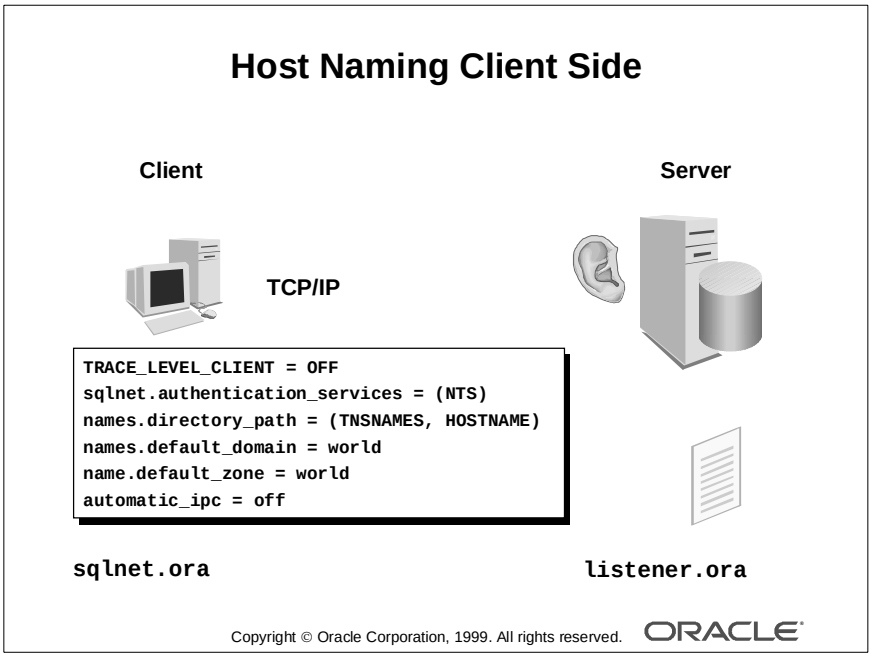
Host Naming Method

- Requires minimal user configuration. The user can provide only the name of the host to establish a connection.
- Eliminates the need to create and maintain a local names configuration file (`tnsnames.ora`).
- Eliminates the need to understand Oracle Names administration procedures.

Local Naming Method

- Provides a relatively straightforward method for resolving service name addresses.
- Resolves service names across networks running different protocols.

Host Naming

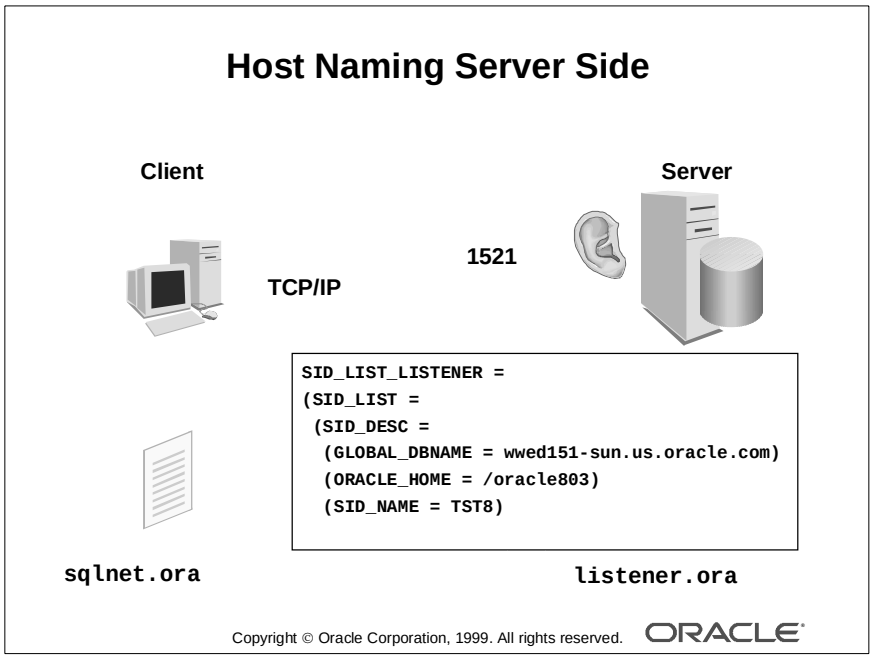


Client-Side Requirements

If you are using the host naming method, you must have TCP/IP installed on your client machine. In addition you must install Net8 and the TCP/IP protocol adaptor.

The host name is resolved through an IP address translation mechanism such as Domain Name Services (DNS), Network Information Services (NIS), or a centrally maintained TCP/IP host file: that means that this should be configured from the client side before attempting to use the host naming method.

The Oracle Connection Manager features are not supported through the host naming method.



Server-Side Requirements

If you are using the host naming method, you must have TCP/IP installed on your server machine as well. You also need to install Net8 and the TCP/IP protocol adaptor on the server side.

A listener process must be started on port 1521 and must include the line:

```
GLOBAL_DBNAME = host name
```

The host name must match the connect string you specify from your client. The additional information included is the database to connect to. In the host naming method, only one database can be specified.

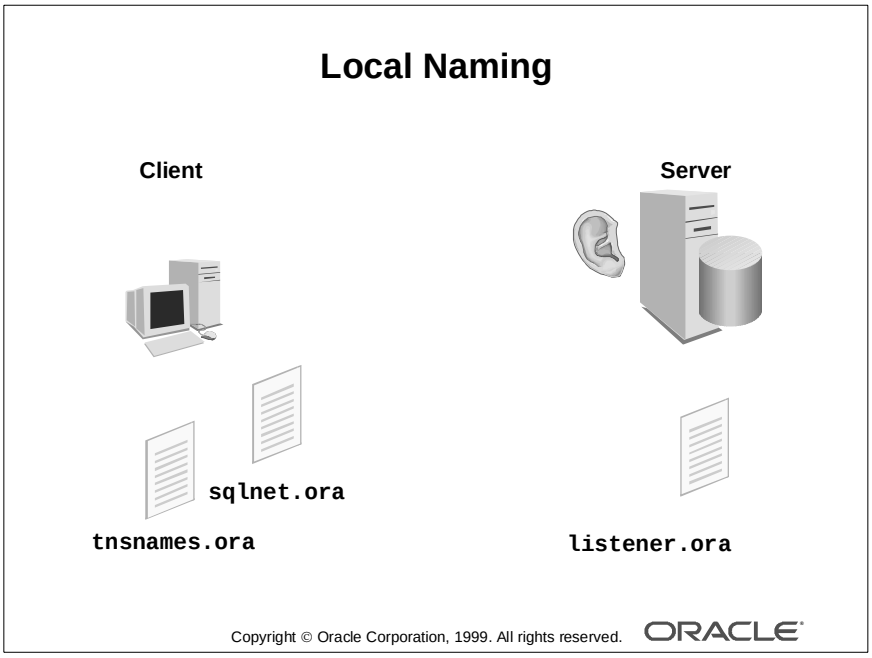
Example

If all of the requirements are met on the client and server side, you can issue the connection request:

```
sqlplus username/password@wwed151-sun
```

from the client, and this connects you to the instance TST8.

Local Naming



Local Naming Requirements

Local naming requires configuration of a local `tnsnames.ora` file.

The local naming is easy to configure through a GUI Net8 Assistant.

The information in the `tnsnames.ora` file is basically an address that is needed for directing your connection to a given listener on a given node listening for a specified database.

The `tnsnames.ora` file is covered in detail later in this lesson.

Configuring Local Naming Using Net8 Assistant



Configuring Local Naming

You can use Net8 Assistant to configure local naming. This method configures the client-side files `tnsnames.ora` and `sqlnet.ora`.

Because Net8 Assistant is implemented in Java and is packed with the Java Runtime Environment, you can run it on any platform where Net8 is installed.

Starting Net8 Assistant from NT

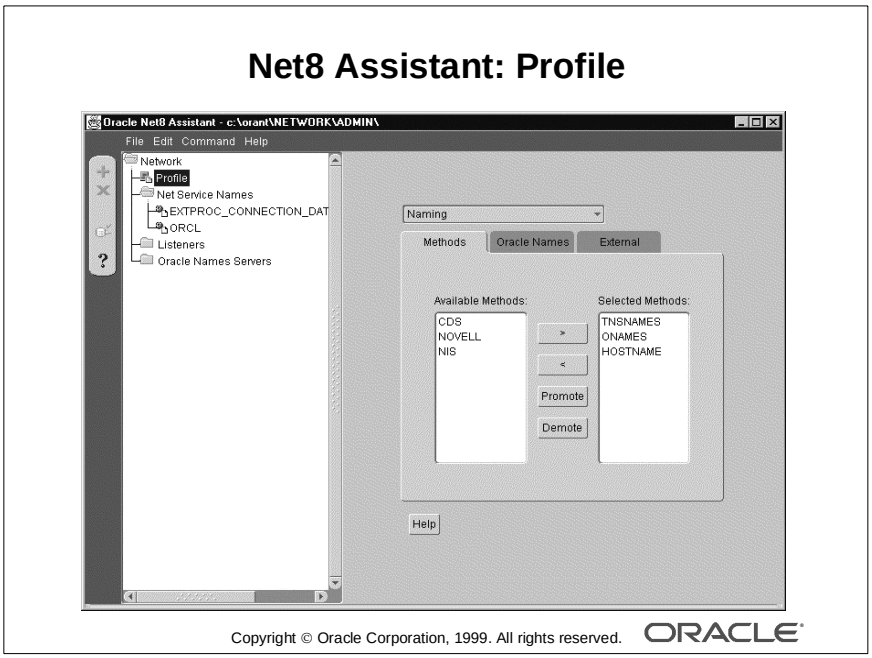
From the NT menu bar, select Start—>Programs—>Oracle for NT—> Oracle Net8 Assistant.

In Oracle8*i* for NT, select Start—>Programs—>Oracle—> Network Administration—>Net8 Assistant.

If you want to start Net8 Assistant from a file manager, the name of the executable file is `n8a.exe`.

Starting Up Net8 Assistant on UNIX

Run the executable file, `net8asst.sh`.



Configuring the Profile

To configure local naming, make sure that the TNSNAMES method is part of your selected method in Net8 Assistant. The TNSNAMES method enables local naming, and this information is stored in the `sqlnet.ora` file.

If you have multiple methods listed in your selected methods, then order them according to the method that you want Net8 to try first when resolving a service name.

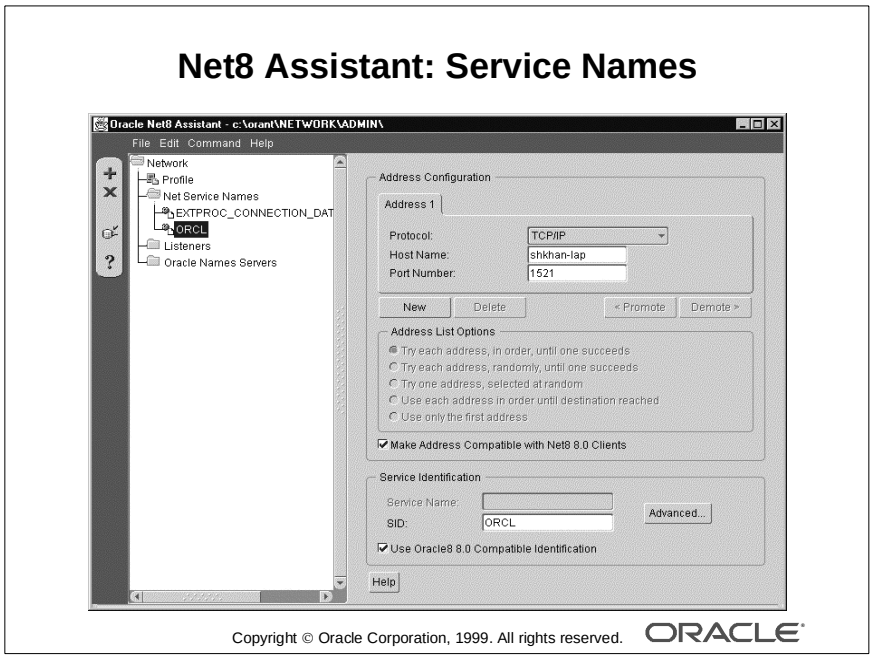
The default installation provides the TNSNAMES and HOSTNAME methods.

In addition to the TNSNAMES and HOSTNAME methods, you can also choose:

- ONAMES: Oracle Names, which allows service names to be resolved centrally through a name server (This is covered in a later lesson.)

The following external naming services must be available from an operating system level before you can configure from Net8 Assistant:

- CDS: Cell Directory Services, which is available with the Advanced Networking Option. This is also the choice for DCE.
- NDS: Netware Directory Service is available for NetWare 4.1 or later.
- NIS: Network Information Service is used on many UNIX servers.



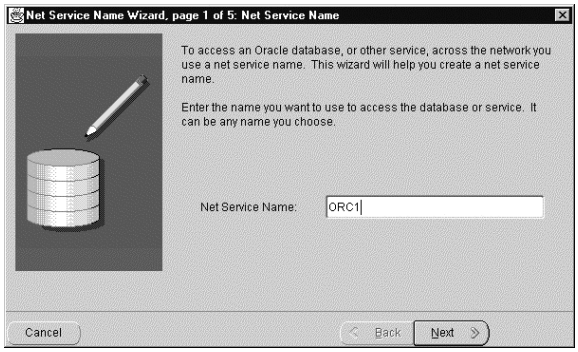
Configuring Service Names

After making sure that the local naming (TNSNAMES) method is chosen, you should configure a service name.

A *service name* is a short, convenient name that is mapped to a network address contained in a connect descriptor that is stored in the `tnsnames.ora` file. Users need know only the appropriate service name to make a connection and not the full connect descriptor.

From the Net8 Assistant menu, select Service Names and then select Edit —>Create.

Net8 Assistant: Service Names



Copyright © Oracle Corporation, 1999. All rights reserved.

Configuring Service Names (continued)

Select a service name and click Next.

When you create a service name, the Net8 Assistant actually starts another application, the Oracle Service Name Wizard, which is used to add, modify, delete, and test service names.

In the Oracle Service Name Wizard window, enter the service name for the address that you want to connect to. An example could be that you are connecting to a database containing financial data, and therefore choose the name FIN.

In the example, the database is a test database running on Oracle8 and the database SID is TST8, therefore the name TST8 is entered as a service name.

Technical Note

The Oracle Service Name Wizard can also be started as a stand-alone application.

Starting Oracle Service Name Wizard from NT

From the NT menu bar, select Start—>Programs—>Oracle for NT—>Oracle Net8 Easy Config.

In Oracle8*i* for NT, select Start—>Programs—>Oracle—>

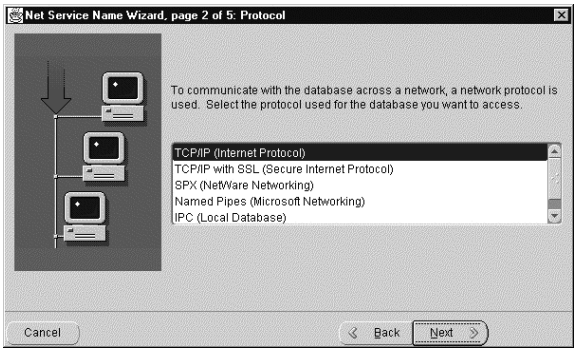
Network Administration—>Net8 Easy Config.

If you want to start the Oracle Service Name Wizard from a file manager, the name of the executable file is `n8sw.exe`.

Starting Oracle Service Name Wizard on UNIX

From the UNIX prompt, run the executable file, `net8wiz.sh`.

Net8 Assistant: Protocol



Copyright © Oracle Corporation, 1999. All rights reserved.

Selecting a Protocol

Select the network protocol you want to use and click Next.

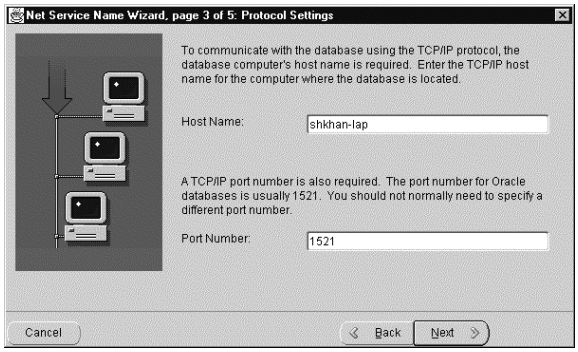
It is required that the network protocol you use be installed and working.

In the example, the TCP/IP protocol is selected.

Technical Note

The network protocol is not the same as the protocol adaptor. The network protocol is the protocol installed on the client and the server prior to the Oracle installation. You should have tested your connection from the client to the server through a tool using the network protocol. An example could be the use of a Telnet or FTP session to test the TCP/IP protocol.

Net8 Assistant: Host Name and Port



Copyright © Oracle Corporation, 1999. All rights reserved.

Configuring the Host Name and Port Number

Enter the host name and the port number, and click Next.

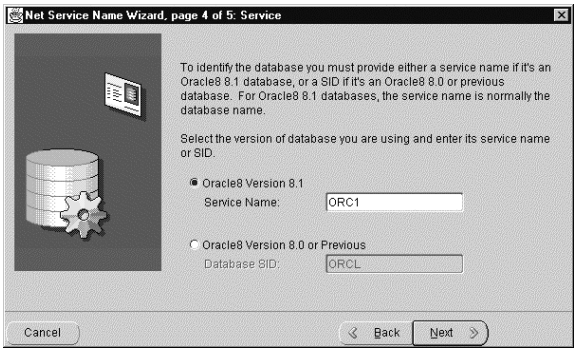
Host Name

The name of the machine on which the database you want to connect and communicate with resides.

Port Number

The number of the port on which Net8 listener listens to connection requests for the server (host). By default, Net8 Assistant sets the listener port to 1521. If required, you can specify an alternative port number.

Net8 Assistant: Database SID

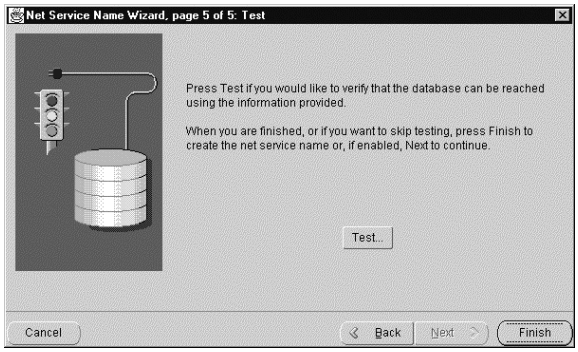


Copyright © Oracle Corporation, 1999. All rights reserved.

Configuring the Database SID

Enter the system ID (SID) of the database to which you want to connect in the Database SID field, and click Next. The default SID is ORCL for Oracle8. For Oracle8*i*, you can specify the SID.

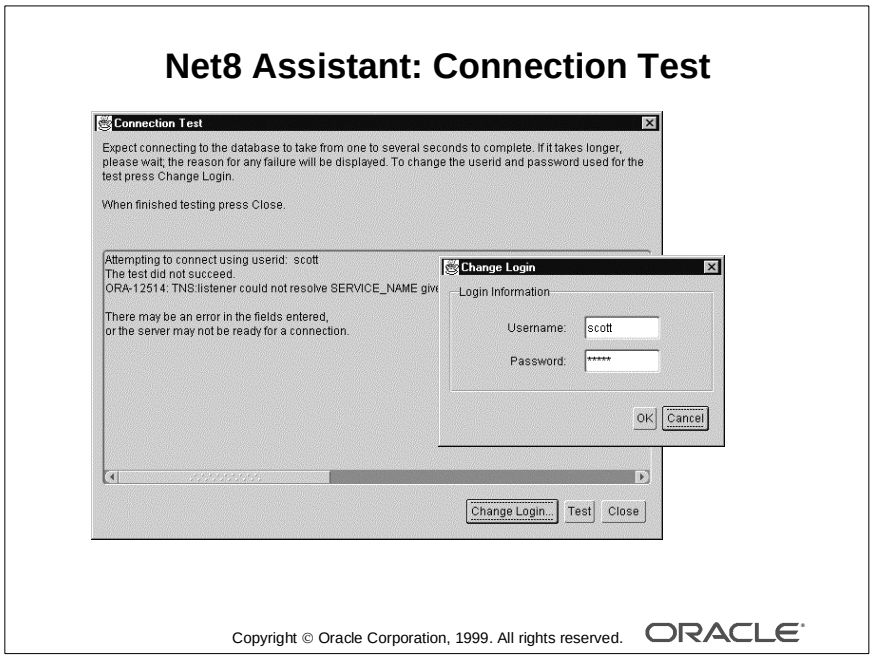
Net8 Assistant: Test Service



Copyright © Oracle Corporation, 1999. All rights reserved.

Testing the Service

Click Finish and save the network configuration. Select Test Net Service Name from the Command menu. Net8 Assistant tries to connect to the service using SCOTT and TIGER as the username and password, respectively.



Testing the Database Connection

Enter a valid username and password for the database to which you want to connect, and click Test.

The status of the test is displayed in the field below the Test button. Click Done when the test is successfully completed.

In Oracle8*i*, Net8 Assistant automatically tests the SCOTT username using TIGER as the password for the default starter database. The option to test the connection against a different user is available by clicking the Change Login button.

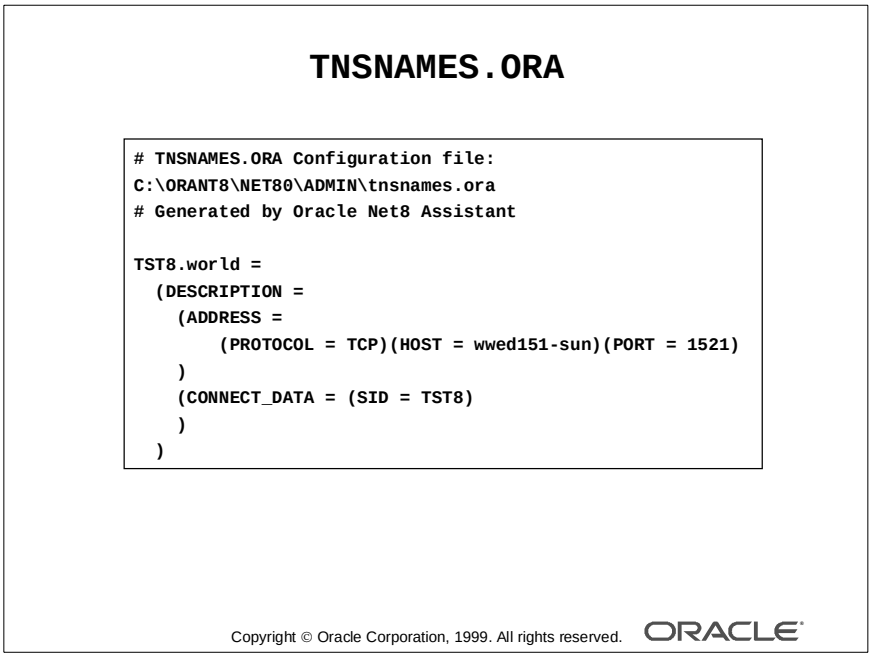
You now return to the Net8 Assistant. Select File—>Save Network Configuration.

If you need to generate additional service names, repeat the procedure.

To modify an existing service name, you can start the Oracle Service Name Wizard and select Modify.

To delete an existing service name, double-click, Service Names, and highlight your service name, then select Edit—>Delete.

Generated Files



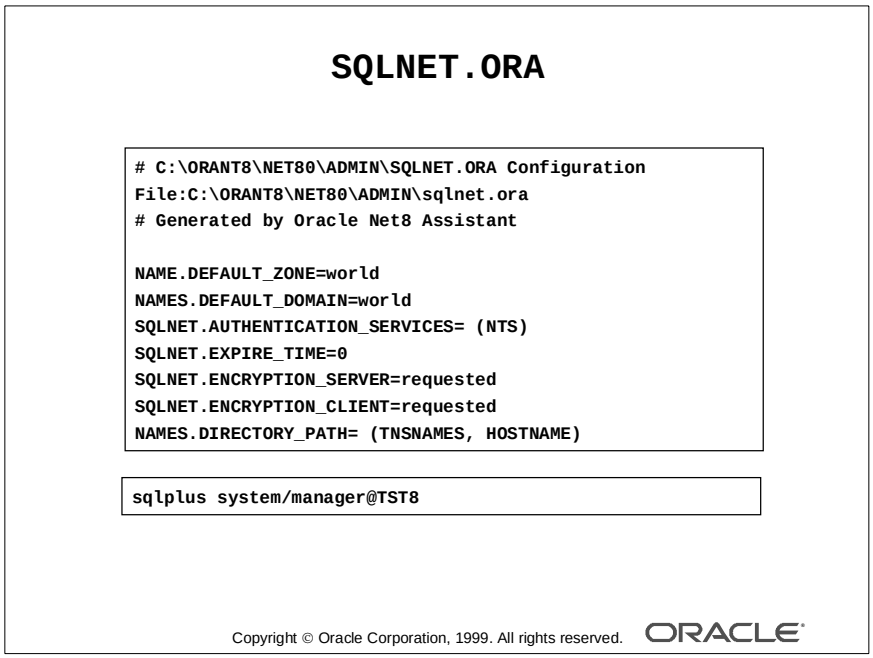
The TNSNAMES.ORA File

The `tnsnames.ora` file is generated from the configuration file.

The default location is `$ORACLE_HOME/network/admin` on UNIX and `ORACLE_HOME\network\admin` on NT.

The content of the `tnsnames.ora` is as follows:

Parameter	Description
TST8.world	Service name and domain name.
DESCRIPTION	Keyword for describing the connect descriptor. Descriptions are always specified the same way.
ADDRESS	Keyword for the address specification. If multiple addresses are specified, use the keyword ADDRESS_LIST prior to the ADDRESS.
PROTOCOL	Specifies the protocol used.
HOST	Protocol-specific information for TCP/IP—specifies the host name of the server or IP address. Can differ for another protocol.
PORT	Protocol specific information for TCP/IP—specifies the port number on which the server side listener is listening.
CONNECT_DATA	Specifies the database SID to which to connect.



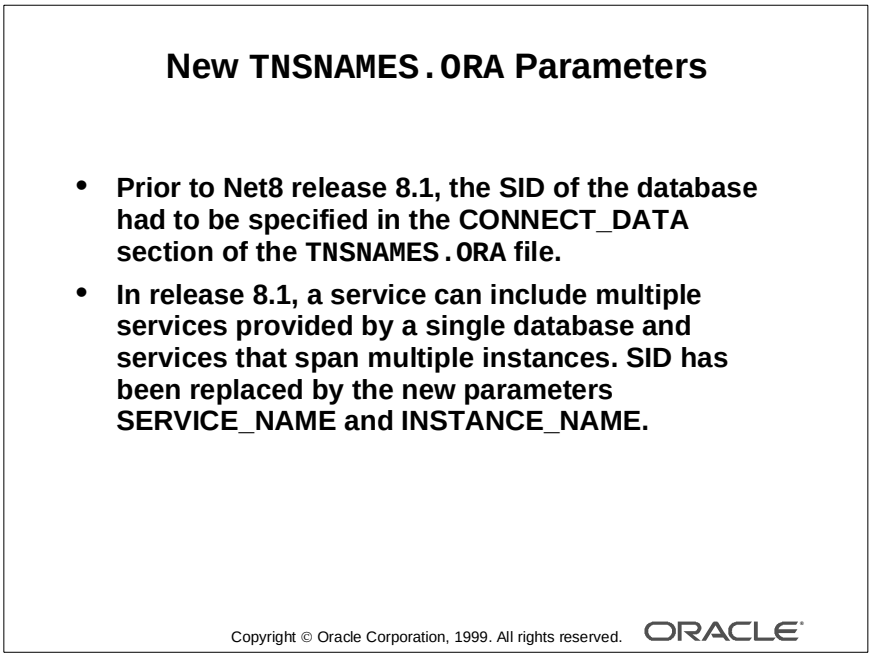
The SQLNET.ORA File

The `sqlnet.ora` file is also generated from the Net8 Assistant.

The default location is `$ORACLE_HOME/network/admin` on UNIX and `ORACLE_HOME\network\admin` on NT, or what is defined in the TNS_ADMIN environment variable.

For now, make sure the connection method you want to use is present and in the correct order. The other arguments are covered in another lesson. Once the service name has been configured and tested successfully, you can connect to the server from the client using any Oracle tool.

New TNSNAMES.ORA Parameters

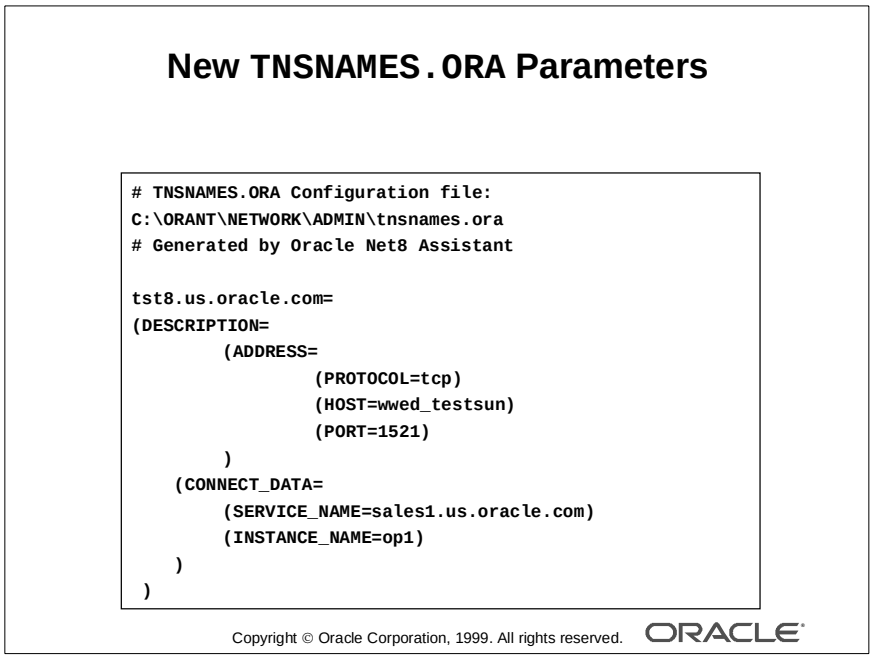


Multiple Instances and Databases

In Oracle8*i*, a service can include multiple services provided by a single database and services that span multiple instances. In earlier releases, the SID identified a database instance, but it did not identify a database. Because of this a database could not have more than one service associated with it.

A database can serve multiple instances, and the SID parameter in the connect descriptor has been replaced by SERVICE_NAME and INSTANCE_NAME.

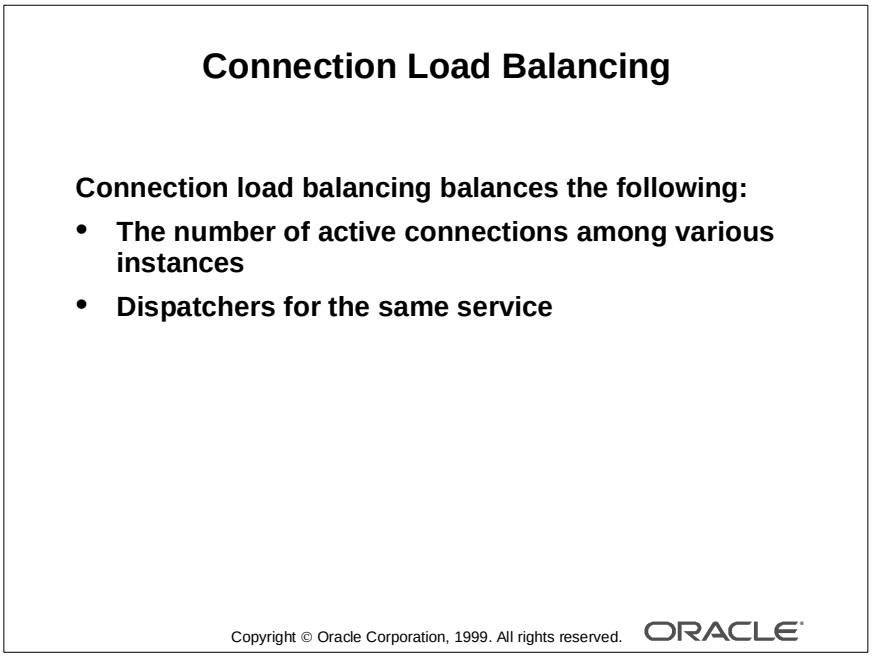
The SERVICE_NAME is typically set to the global database name, which consists of the database name and the domain name. The parameter INSTANCE_NAME is required only for a parallel server configuration.



Configuring the New Oracle8*i* TNSNAMES.ORA Parameters

1. Start Net8 Assistant: Select Start—>Programs—>Oracle—> Network Administration—>Net8 Assistant.
2. Double-click the Net Service Names folder to list the service names.
3. Click the service name that needs to be modified.
4. In the right window, clear the Use Oracle8*i* 8.0 Compatible Identification check box.
5. The Service Name field is highlighted.
6. Enter the database name with the domain name: U*nn.domain*
7. Save the network configuration.
8. Browse the contents of the TNSNAMES.ORA file.

Connection Load Balancing

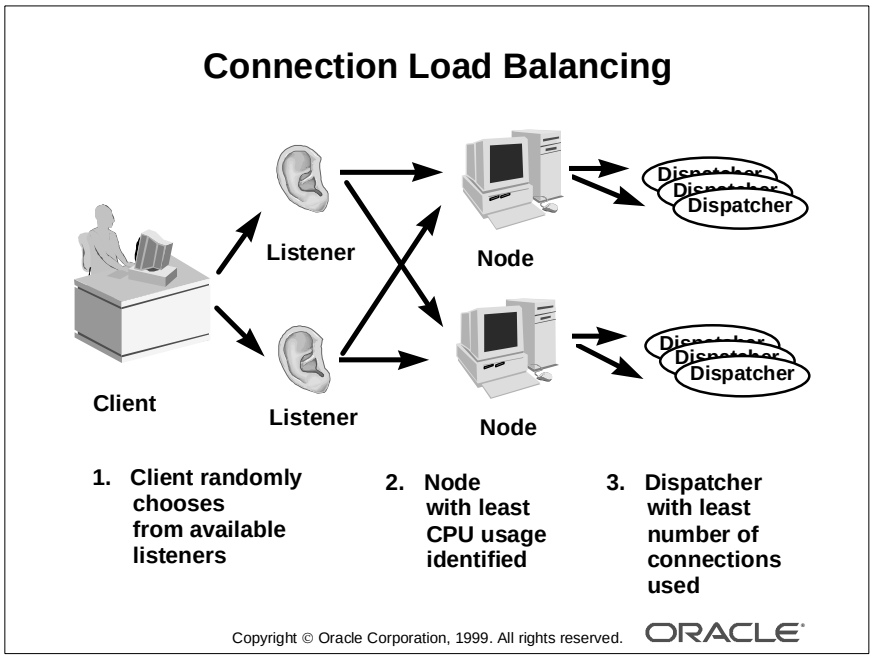


Overview of Connection Load Balancing

When automatic instance registration is enabled, as described in lesson 3, connection load balancing is also enabled. Connection load balancing evenly distributes the number of active connections among various instances and dispatchers for the same service.

Connection load balancing is new in Oracle8*i* and enables balancing the load at two levels: dispatchers and nodes. The load of an instance and dispatcher is determined by the number of connections. Connection load balancing is only enabled for the multithreaded server environment.

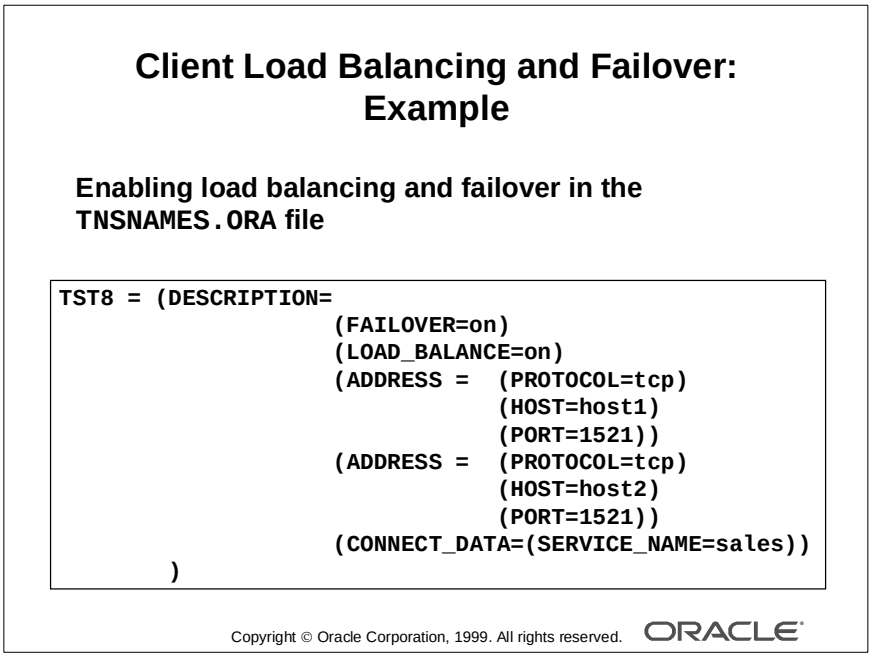
A listener sends an incoming client request for a specific service to the least-loaded dispatcher and instance. For example, if the service has multiple instances on multiple nodes, it chooses a dispatcher based on the least-loaded instance, where the instance load is based on the node load.



Connection Load Balancing Process

1. A client requesting a connection to a service randomly picks a listener from the list of listeners identified for the service in the `tnsnames.ora`. If the chosen listener is not available, the connection request can failover to the next available listener. This capability is called *client load balancing and failover.*
2. Each listener has information about the load on all available nodes, as each instance has registered with all identified listeners and is providing continuous load information (CPU usage). For each connection request, the listener identifies which node has the least amount of usage.
3. The listener then also identifies which dispatcher (multithreaded server, or MTS configuration) is currently handling the least number of connections on that node. The connection is then routed to that dispatcher on the chosen node.

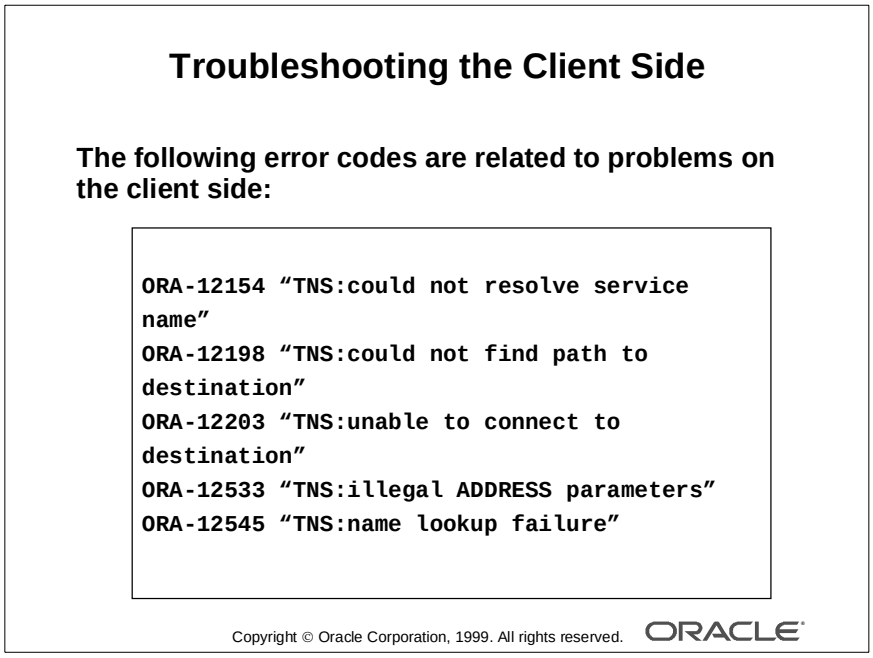
Connection Load Balancing and Failover: Example



Enabling Load Balancing and Failover

Failover is now supported by default for the ADDRESS_LIST and DESCRIPTION_LIST parameters. Failover can also be explicitly specified by the FAILOVER parameter for a set of addresses.

Troubleshooting the Client Side



Troubleshooting

The following describes common errors and how they may be resolved.

`ORA-12154: “TNS:could not resolve service name”`

Cause Net8 could not locate the connect descriptor specified in the `tnsnames.ora` configuration file.

Actions

1. Verify that a `tnsnames.ora` file exists and that it is accessible.
2. Verify that the `tnsnames.ora` file is in the location specified by the TNS_ADMIN environment variable.
3. In your `tnsnames.ora` file, verify that the service name specified in your connection string is mapped to a connect descriptor in the `tnsnames.ora` file. Also, verify that there are no syntax errors in the file.
4. Verify that there are no duplicate copies of the `sqlnet.ora` file.
5. If you are connecting from a login dialog box, verify that you are not placing an *at* symbol (@) before your connection service name.

Troubleshooting (continued)

`ORA-12198: "TNS:could not find path to destination"` and
`ORA-12203: "TNS:unable to connect to destination"`

Cause The client cannot find the desired database.

Actions

1. Verify that you have correctly entered the service name for the database you want to reach.
2. Verify that the service name ADDRESS parameters in the connect descriptor of your `TNSNAMES.ORA` file are correct.
3. Verify that your `TNSNAMES.ORA` file is stored in the directory defined in the TNS_ADMIN environment variable.
4. Verify that the listener on the remote node has started and is running. If not, start the listener by using the Listener Control utility.
5. If you are connecting from a login dialog box, verify that you are not placing an *at* symbol (@) before your connection service name.

`ORA-12533: "TNS:illegal ADDRESS parameters"`

Cause The protocol-specific parameters in the ADDRESS section of the designated connect descriptor in your `TNSNAMES.ORA` file are incorrect.

Action For more information about protocol-specific keywords, refer to the Oracle operating system documentation for your platform.

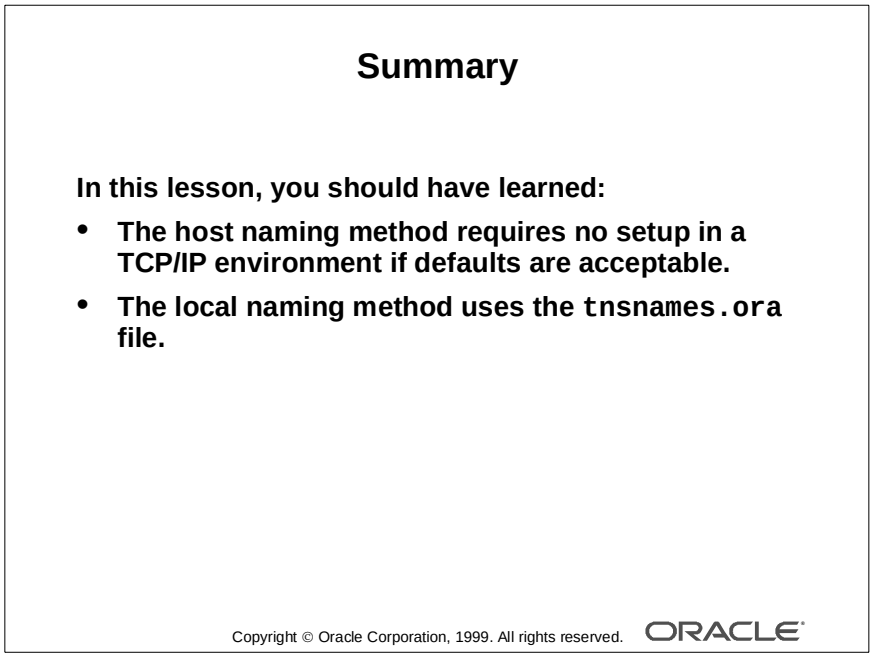
`ORA-12545: "TNS:name lookup failure"`

Cause The listener on the remote node cannot be contacted.

Actions

1. Verify that the ADDRESS in the `TNSNAMES.ORA` file or the `LISTENER.ORA` file is correct.
2. Verify that the listener on the remote node has been started. You can check its status with the STATUS command of the Listener Control utility and start it with the START command if necessary.

Summary


Summary
In this lesson, you should have learned:
• The host naming method requires no setup in a TCP/IP environment if defaults are acceptable.
• The local naming method uses the tnsnames.ora file.
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Frequently Asked Questions

The following discussion points are taken from the most frequently asked questions regarding topics in this chapter. This information is retrieved and analyzed in cooperation with Oracle Worldwide Support.

Problem

After you upgrade to Oracle8 and upgrade clients to Net8, the following error occurs when you try to connect through SQL*Plus:

```
06413, 00000, “Connection not open.”
// *Cause:  Unable to establish connection.
// *Action: Use diagnostic procedures to ascertain exact
problem.
```

Solution

Verify that SQL*Net 2.*x* and Net8 are not in the same directory. Both of these cannot be installed in the same directory and share tools.

Problem

How to disable *out of band breaks* using Net8.

Solution

To disable *out of band breaks*, set the DISABLE_OOB parameter to ON in the client’s `sqlnet.ora` file. Net8 sets the DISABLE_OOB to OFF by default.

Problem

When you attempt to connect to an Oracle8 database in loopback mode (a connection from the same machine to the same machine, acting as the client and the server as well) using Net8, the following error occurs:

```
ORA-06401: NETCMN: invalid driver designator.
```

The loopback is attempted by using either of the following two strings:

```
sqlplus system/manager@2:
sqlplus system/manager@2:orcl
```

Solution

The 2: and 2:orcl (2:sid) connection strings for local connections to Oracle8 are not supported. Instead, use the default connection string for a local connection through the Bequeath protocol adapter. By default, `tnsnames.ora` contains the following alias for the bequeath protocol adapter:

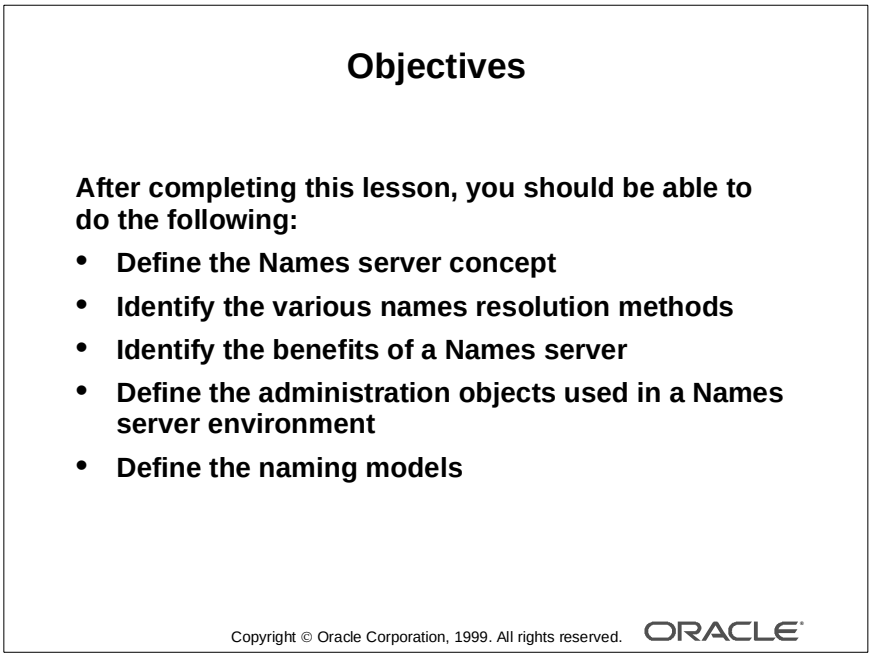
```
Beq-local.world =
   (DESCRIPTION =
      (ADDRESS =
         (PROTOCOL = BEQ)
         (PROGRAM = oracle80)
         (ARGV0 = oracle80ORCL)
        (ARGS = '(DESCRIPTION=(LOCAL=YES)(ADDRESS=(PROTOCOL=BEQ)))')
      )
   (CONNECT_DATA = (SID = ORCL))
)
```

Note: When the PROTOCOL parameter is defined with the BEQ variable, any connection made to service name will use an internal protocol called Bequeath. The Bequeath protocol internally spawns a server thread for each client application. In a sense, it performs the same operation that a remote network listener does for your connection, yet locally.

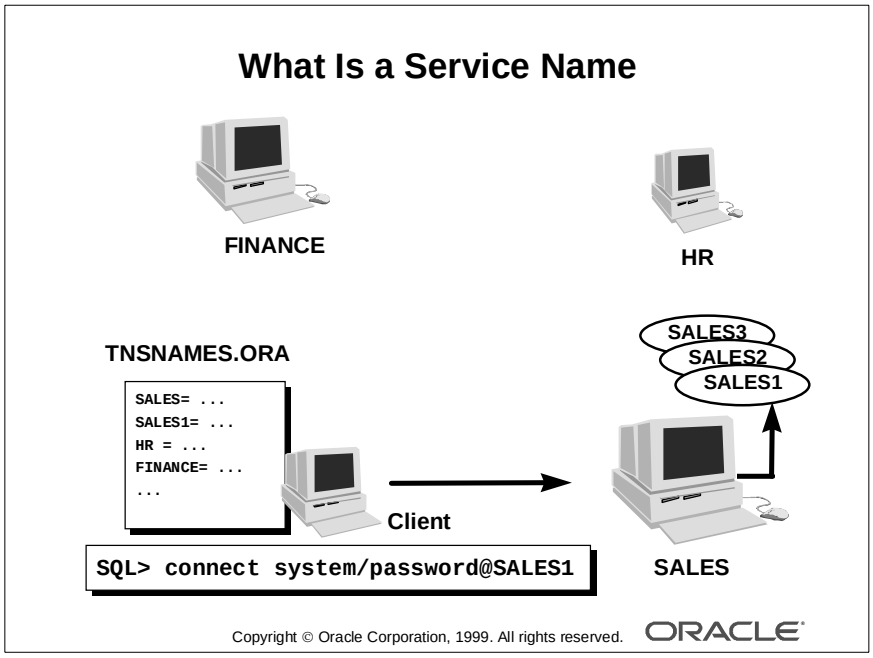
5

Centralized Naming Concepts

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Define the Names server concept
• Identify the various names resolution methods
• Identify the benefits of a Names server
• Define the administration objects used in a Names server environment
• Define the naming models
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

What Is a Service Name?



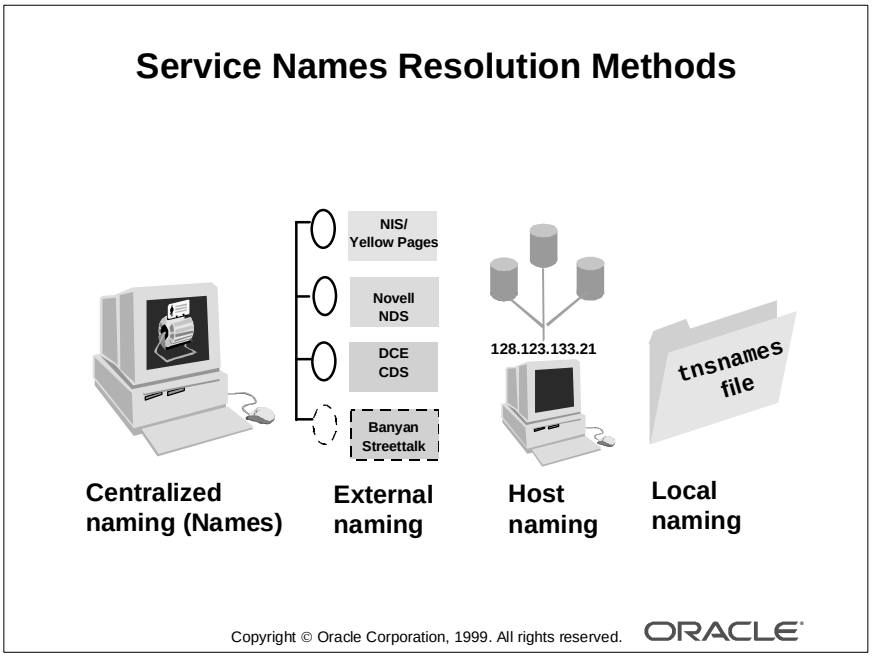
Service Names

A service name is an alias for a logical representation of a service, such as a database. It is used to resolve complicated connection strings used to connect to a service. It is the way a database or other service is presented to clients. A service name could be any one of the following examples:

- SALES could be created to match the database name (Oracle8*i*)
- SALES1 could be an instance of the SALES database
- HR could be created to match the name of the personnel system
- FINANCE could be created to match the name of the finance system

With Net8, you can connect to services other than Oracle databases, such as non-Oracle databases, gateways, and external procedures, and functions that can be called from PL/SQL code.

Service Name Resolution Methods



Resolving a Service Name

Complicated addresses of databases, database links, and nodes can be given simpler names using various methods. When a network name or service name, such as the name of a database or database link, is "resolved" to an address, which may contain a connection string, the mechanism is called *name resolution*. Oracle resolves service names to connection strings using one of the following methods:

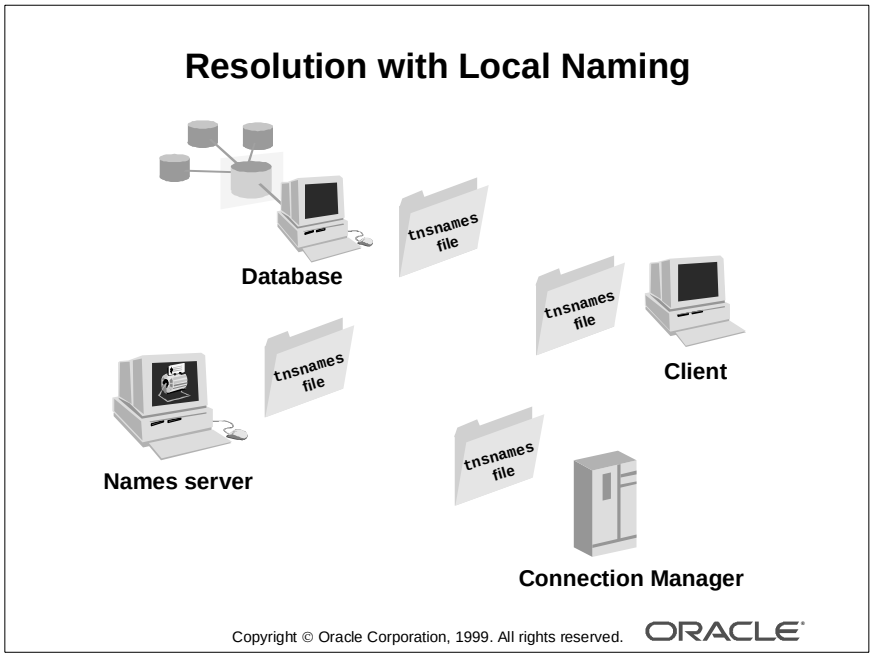
Local Naming This method uses a `tnsnames.ora` file to look up service names.

Host Naming The need to configure a `tnsnames.ora` file is no longer required using this method. Instead, the actual name of the machine on which the database resides is used to establish a connection to the service.

External Naming Various third-party name resolution mechanisms can integrate with Oracle without the need of eliminating the existing mechanism and starting again.

Centralized Naming Using Oracle Names This method uses Oracle Names, which is a names service that maintains central storage of network service addresses.

Resolution with Local Naming



Resolution with Local Naming

If there are a few nodes on a network, managing changes in the environment is not a difficult task. Each node has its own `tnsnames.ora` file, and changes can be replicated across the nodes.

Disadvantages of Using Local Naming in a Distributed Environment

If a `tnsnames.ora` file is used within a large distributed environment, a number of issues present themselves:

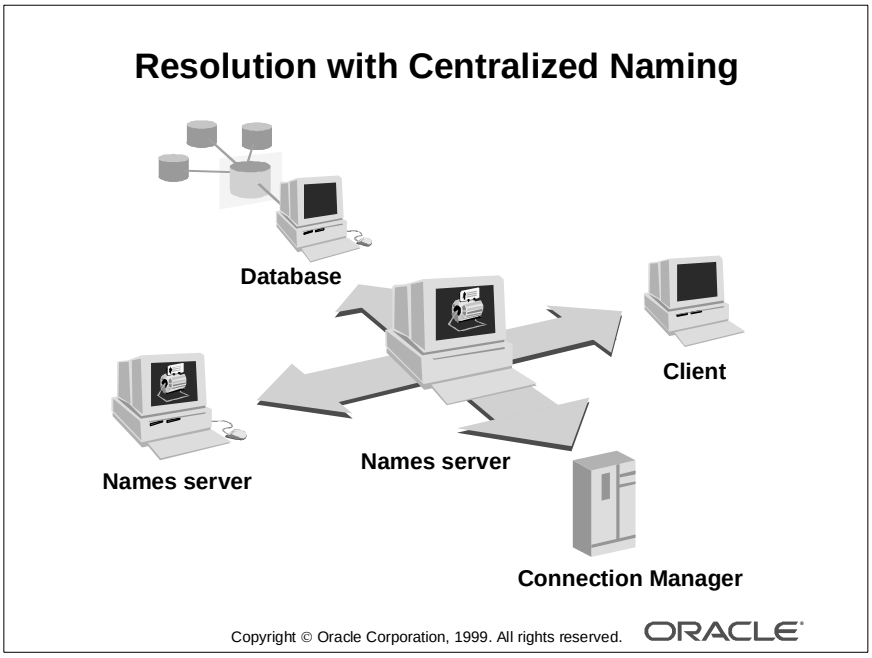
- Changes must be propagated efficiently across the network nodes.
- Accuracy of the `tnsnames.ora` file contents must be maintained.
- Additional administration time and resources are needed when rapid changes occur.
- Consistency must be maintained between all the nodes.

Disadvantages of Using Local Naming

This service is available only if the following are true:

- The TCP/IP protocol is used for connectivity between the clients and the servers.
- A centrally located host file, a Domain Name Server (DNS), or Network Information Services (NIS) server is present.

Resolution with Centralized Naming



Resolution with Centralized Naming

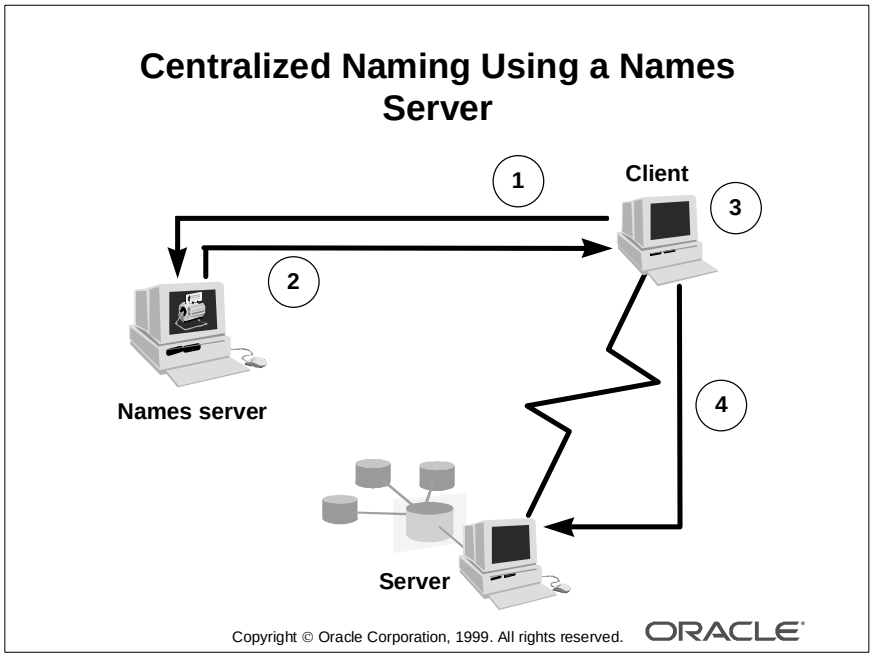
When implementing client-based and server-based solutions as a means of achieving critical business objectives, the biggest challenge occurs when the computing environment is not dynamically configurable. The centralized naming technology greatly reduces administrative burdens associated with such networks.

Benefits of Using Centralized Naming

Using a Names server offers the following benefits:

- Simplified administration tasks: Administrators need to define the names of servers and their location only once, and all clients can then access them.
- Increased efficiency: The server addresses are stored on one or more machines designated as Oracle Names server nodes, eliminating the need to store these addresses on every node.
- Eliminates redundancy: Server address changes are made in one place and all clients can access the new addresses immediately.
- Facilitates location transparency: Using a Names server facilitates server location transparency by allowing the client to use a symbolic service name (stored in a database) for the server location.

Centralized Naming Using a Names Server



Resolving Addresses with Oracle Names

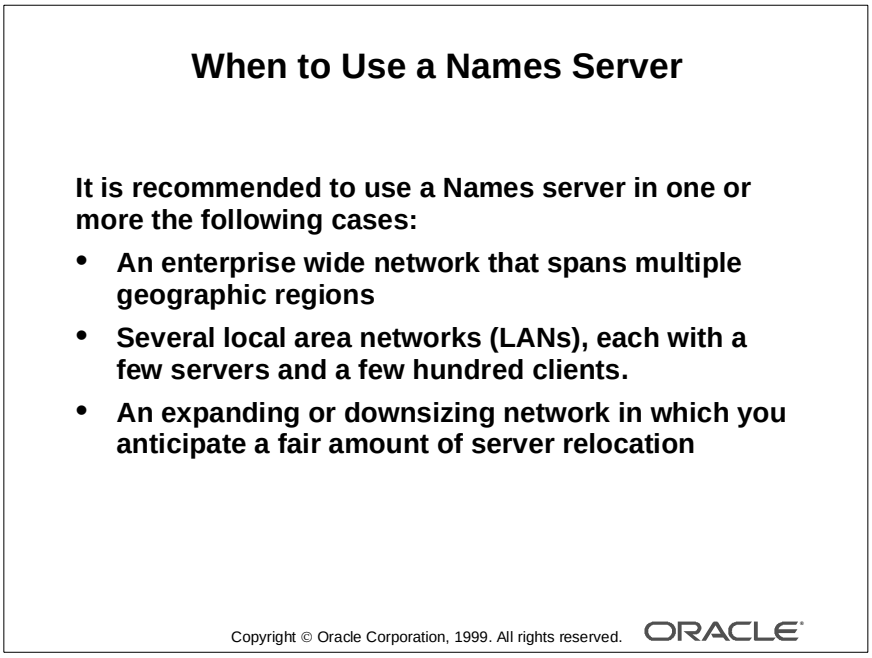
The Names server resolves a service name client request by translating the service name to a connect string.

Steps in Service Name Resolution

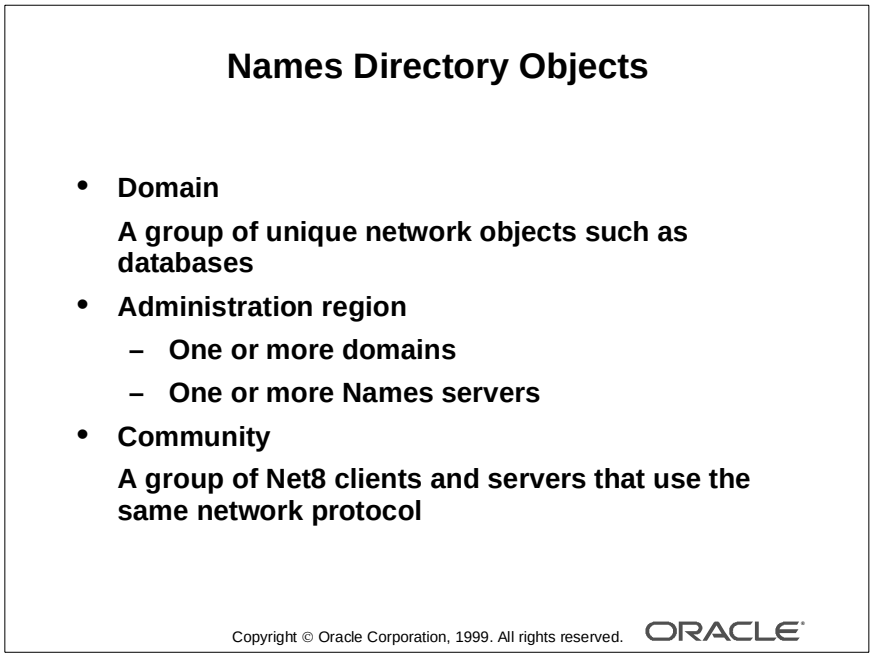
In the figure in the slide, the client is attempting to connect to the server using a service name. The client is configured to use a Names server to resolve the service name. The following steps describe how the service name is resolved and the connection established:

1. Net8 sends the appropriate Names server a request to resolve the service name.
2. The Names server receives the request, looks it up the name in the cache or database, and sends the result back to the client.
3. The client receives the result and substitutes the connect string in place of the service name.
4. The client contacts the server and establishes a Net8 client-server connection.

When to Use a Names Server



Oracle Names Directory Objects



Domain

A domain is a grouping of network objects, such as machines or databases, that simplify the naming of network services. Within a domain, all names for the object must be unique. A domain may use one of the two following naming models:

- Flat naming model
- Hierarchical naming model

Note: These models are covered on the next page.

Administrative Region

A region or administrative region is an organizational entity for administrating Net8 network components. There can be any number of Oracle Names servers.

Each region includes the following:

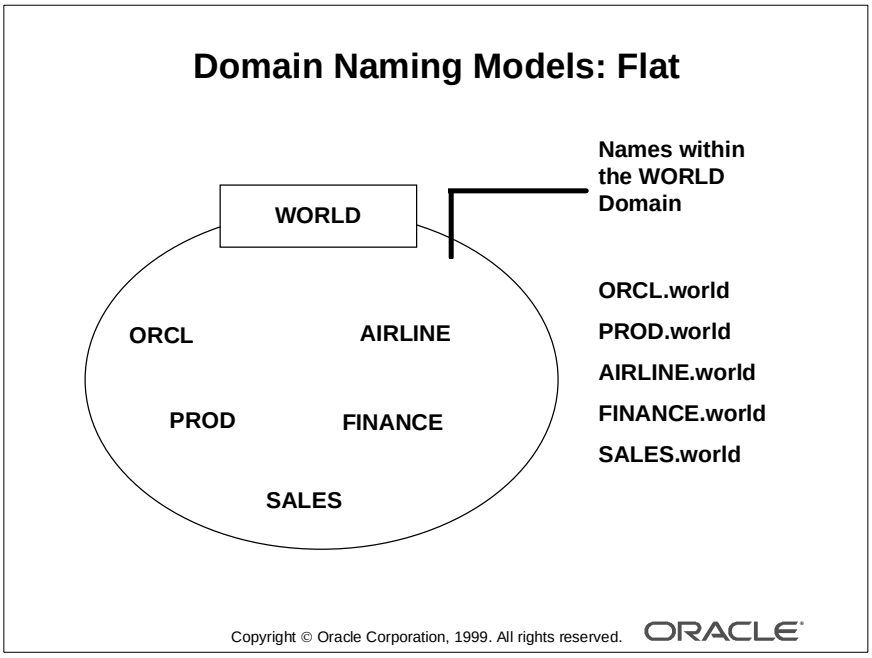
- One or more domains
- One or more Names servers

All connect information is stored in a single data repository, which has the authority to interpret a service name. All Oracle Names servers within an administrative region query information from this data repository. If the administrative region uses a database for storage, there is one database per administrative region.

Community

A Community refers to a group of Net8 network clients and servers using the same industry-standard protocol. It is recommended to have one Names server per community.

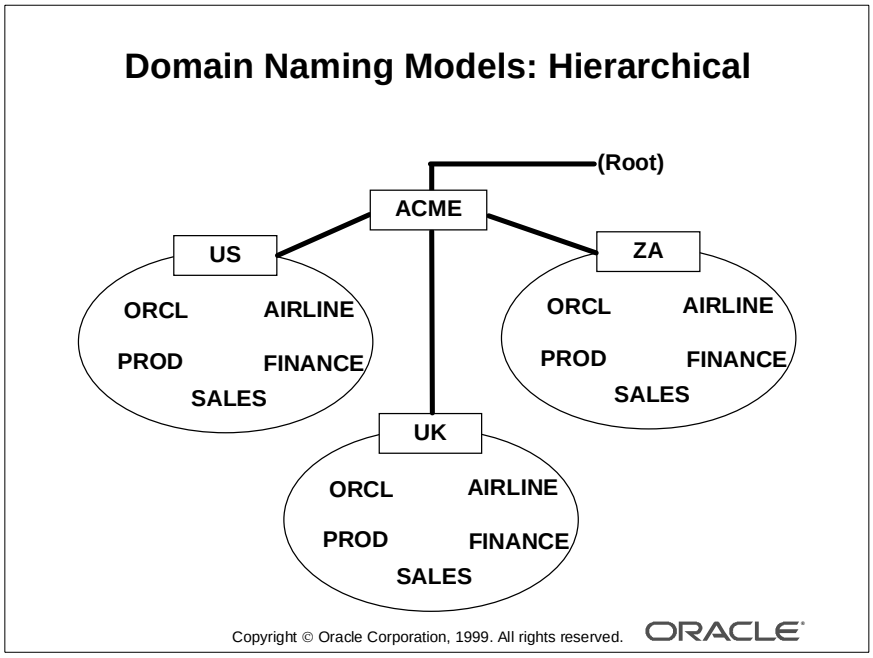
Domain Naming Models: Flat



Flat Naming Model Characteristics

- Consists of a single domain (default domain name is .WORLD)
- All names must be unique in the single domain.
- Used when the entire network is centrally administered
- Most common naming model

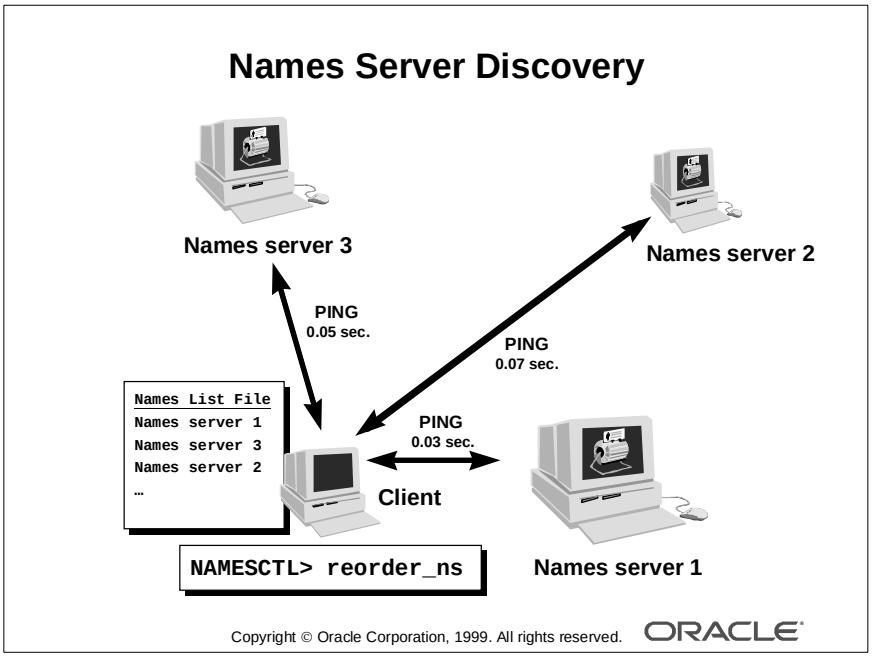
Domain Naming Models: Hierarchical



Hierarchical Naming Model Characteristics

- Enables distribution of network administration responsibilities
- Maintains domains in a hierarchical structure below a top-level domain called the root domain
- Contains unique names for each domain but permits names to be repeated across domains
- Allows the domain hierarchy to be extended to any number of levels (for example, ORCL.US.ACME.COM)
- Commonly used in large organizations

Names Server Dynamic Discovery

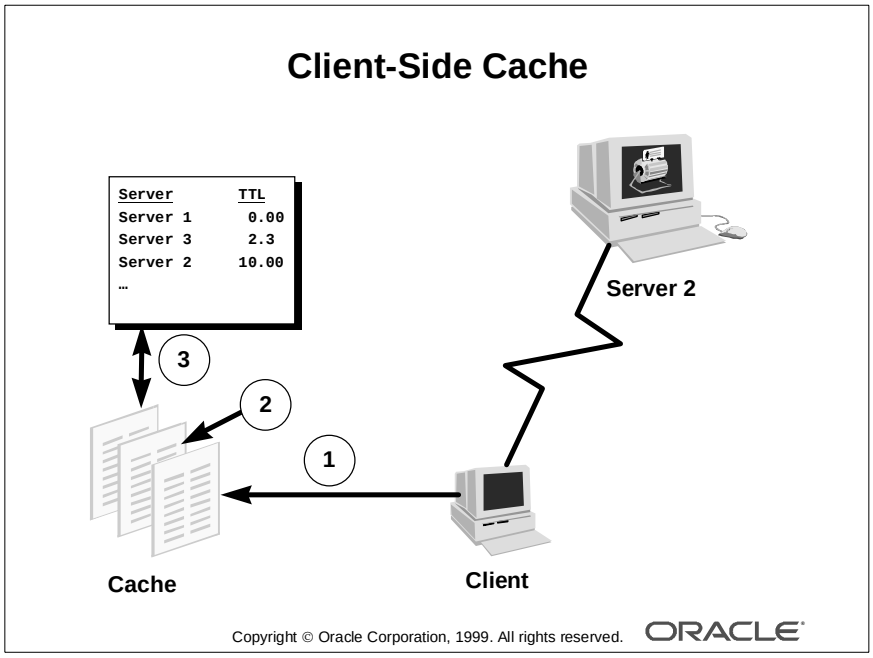


Names Server Discovery

Clients and servers on the network discover Names servers on the network when the REORDER_NS command is issued using the Names Control utility (NAMESCTL). When this command is issued, the following occurs:

1. A Names server is contacted by using the address specified in:
   - The command itself
   - The preferred Names server parameter in the local profile (`sqlnet.ora` file)
   - An existing Names server list
   - A well-known Names server
2. If a Names server is found, NAMESCTL sends a query for all the Names servers in the region.
3. A PING command is then sent to each of these servers to determine the time it takes for each Names server to respond.
4. The list of the Names servers is sorted in increasing order of response time.
5. The existing list is replaced by the sorted Names server list. If the Names server list does not exist, it is created.

Client-Side Cache



Client-Side Cache

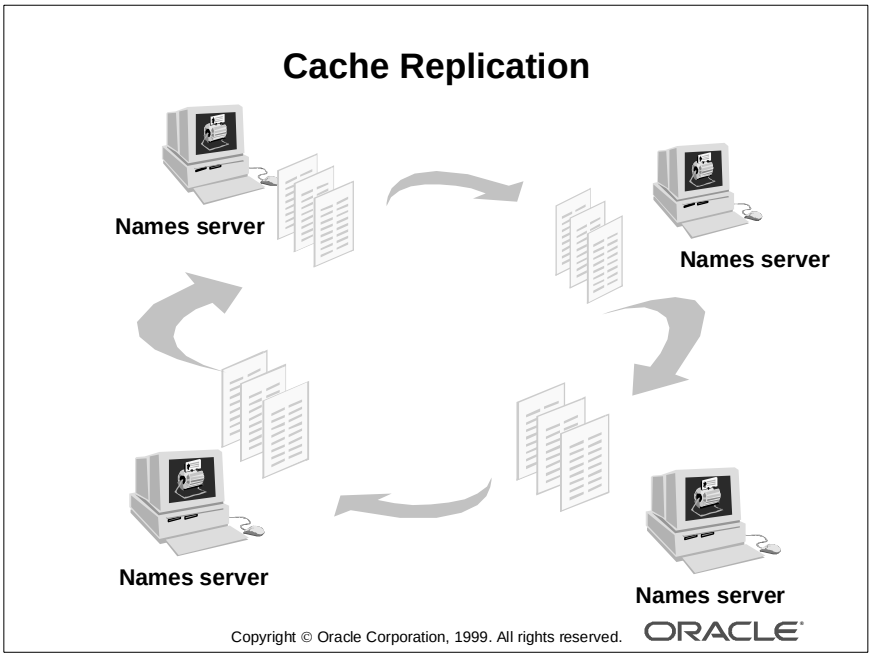
A client cache daemon is installed with the Names server that enables clients on most platforms to store information retrieved from that Names server in a local cache. The client cache has the following functions:

1. When a client connects to a server, the connection information of the server is stored in the client cache.
2. When a subsequent connection request is made to the same service name, the client checks its local cache for the connection information.
3. If the information is located, the time to live (TTL) is checked.
4. If the TTL has not expired, the client uses the information to connect to a server, saving the time it would normally take to requery the Names server.

The local client-side cache is particularly advantageous if a Names server is not available. In this case, the local client-side cache has a list of recently accessed service names.

To start the client cache daemon process, use the START_CLIENT_CACHE command from Names Control utility. More about the START_CLIENT_CACHE command is covered in lesson 6.

Cache Replication



Cache Replication

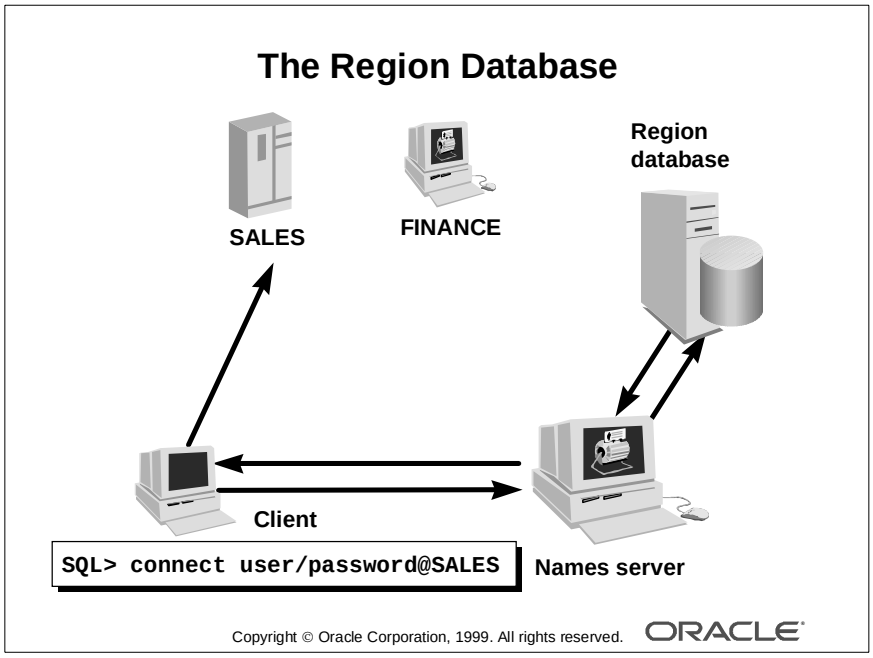
Data in Oracle Names servers is updated through continuous replication among all the Oracle Names servers in the region, or by writing to and reading from a common Oracle database.

For smaller workgroup environments where all of the services are registered dynamically, administrators can configure Oracle Names servers to replicate data continuously among themselves. When a listener registers a new service, information about that service will immediately be passed along to other Oracle Names servers in the administrative region. In this way, service names and addresses are replicated among other Names servers in the network.

In larger networks, it is safer and more efficient to use a region database. Cache replication is best suited for networks with just a few Names servers.

Cache replication occurs, by default, when a region database is not defined.

The Region Database



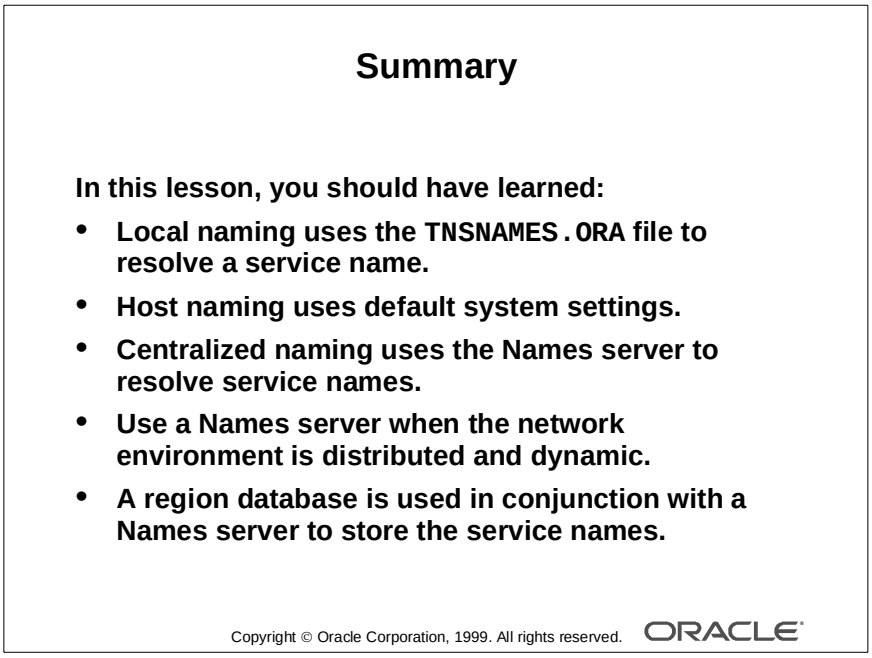
Using a Region Database

Administrators in large environments normally store their service name in an Oracle database, rather than have the service names replicated between Names servers (as in Cache replication).

If Names servers are configured to use an Oracle database as a repository, all registered service names are written to the database. Additional Names servers in a given administrative region periodically poll the region database for updated service names.

In this way, new service name are communicated in a timely manner to all of the Oracle Names servers in a given region. At the same time, it relieves Names servers of the necessity to communicate directly with each other, as well as provides better reliability.

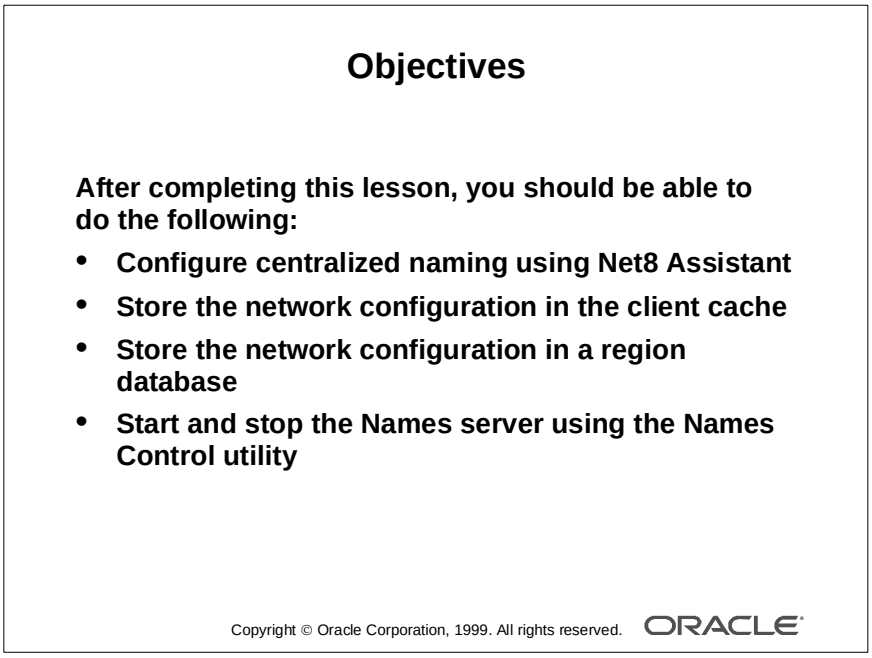
Summary


Summary
In this lesson, you should have learned:
• Local naming uses the TNSNAMES.ORA file to resolve a service name.
• Host naming uses default system settings.
• Centralized naming uses the Names server to resolve service names.
• Use a Names server when the network environment is distributed and dynamic.
• A region database is used in conjunction with a Names server to store the service names.
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

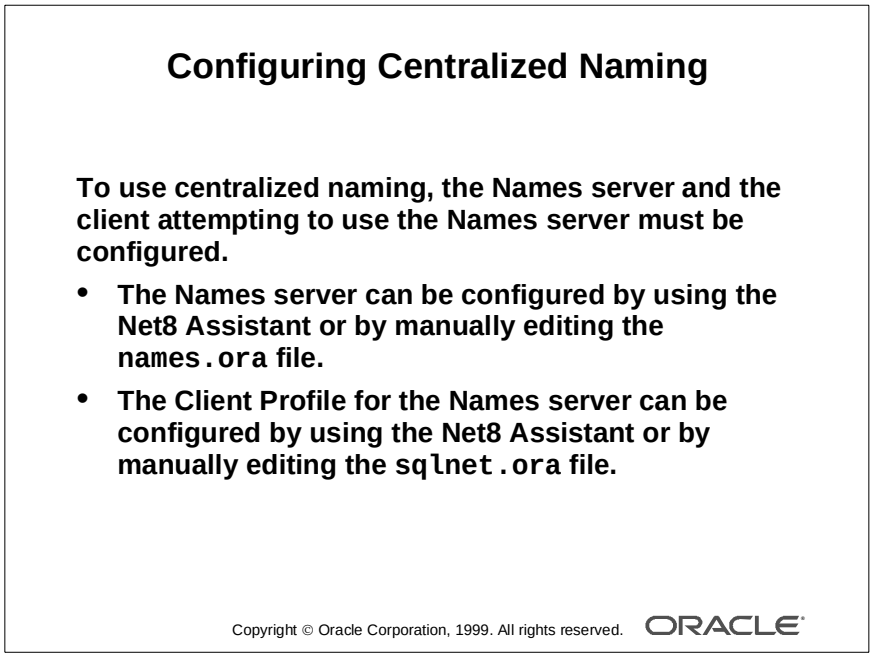
6

Oracle Names Usage and Configuration

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Configure centralized naming using Net8 Assistant
• Store the network configuration in the client cache
• Store the network configuration in a region database
• Start and stop the Names server using the Names Control utility
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Configuring Centralized Naming



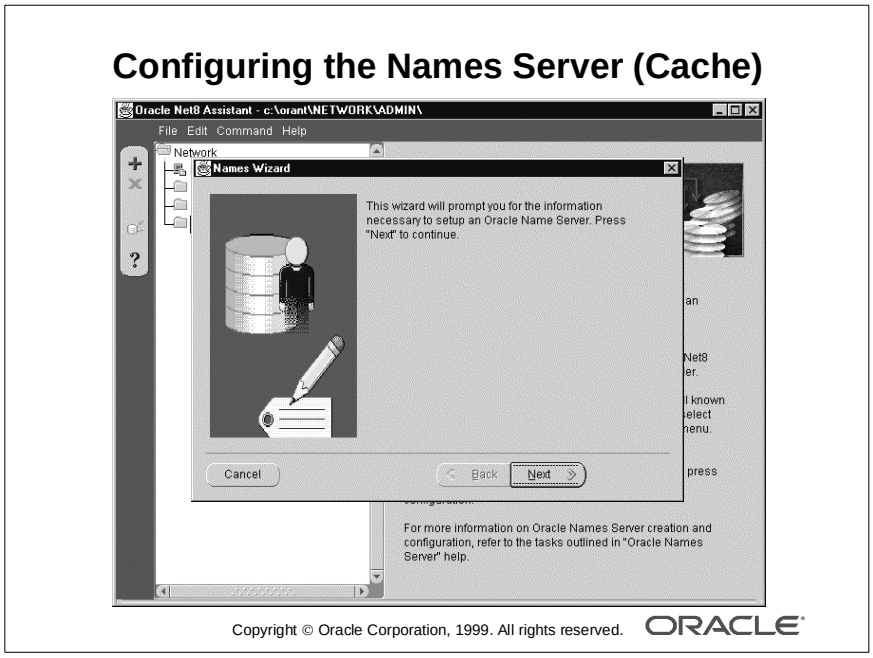
Configuring Centralized Naming

The following steps summarize how to configure the Names server:

1. **Create and configure the Names server using the Names Wizard.**
2. **Start up the Names server.**
3. **Enter service names into the Names server.**

Copyright © Oracle Corporation, 1999. All rights reserved.

Configuring the Names Server (Cache)



Creating and Configuring the Names Server (Cache Replication)

1. Start Net8 Assistant: Select Start—>Programs—>Oracle—> Network Administration—>Net8 Assistant
2. Click the Oracle Names Servers icon in Net8 Assistant.
3. Click OK after reading the instructions in the pop-up dialog box.
4. Select Create from the Edit menu.

   This starts up the Names Wizard, which is a tool that steps you through the configuration process of a Names server.
5. Click Next as you step through the instructions of the Names Wizard.
6. Enter a name for the Names server. Attach a domain name with the name of the Names server if needed.
7. Select the protocol, and enter the name of the node on which the Names server is to reside and the port at which it is to accept incoming resolution requests.

   Note: In this class, the Names server is set up on your local NT machine. Thus, a service name query of a database server originates from your local machine to the Names server located on the same machine.
8. Select Next as you step through the instructions of the Names Wizard.
9. Click the “Don’t use a region database” option button, and click Next.

Creating and Configuring the Names Server (Cache Replication) (continued)

First Names Server in the Region If this is the first Names server in this region, follow the steps below:

1. Select the "Names server is first in its region" option button, and click the Next button.

   If this is the first Names server in your network, then the region is your root region.
2. Select the Yes option button to acknowledge that the Names server is in the root region.
3. Click the Finish button to complete the Names server basic configuration.

   The `names.ora` file is created at this point with default parameters.
4. Click OK when the dialog box appears informing you of a new Names server.

   This step may take some time, as the Names server attempts to discover any existing Names servers in the same domain.

Names Server Is Not First in the Region If this is not the first Names server in this region, follow the steps below:

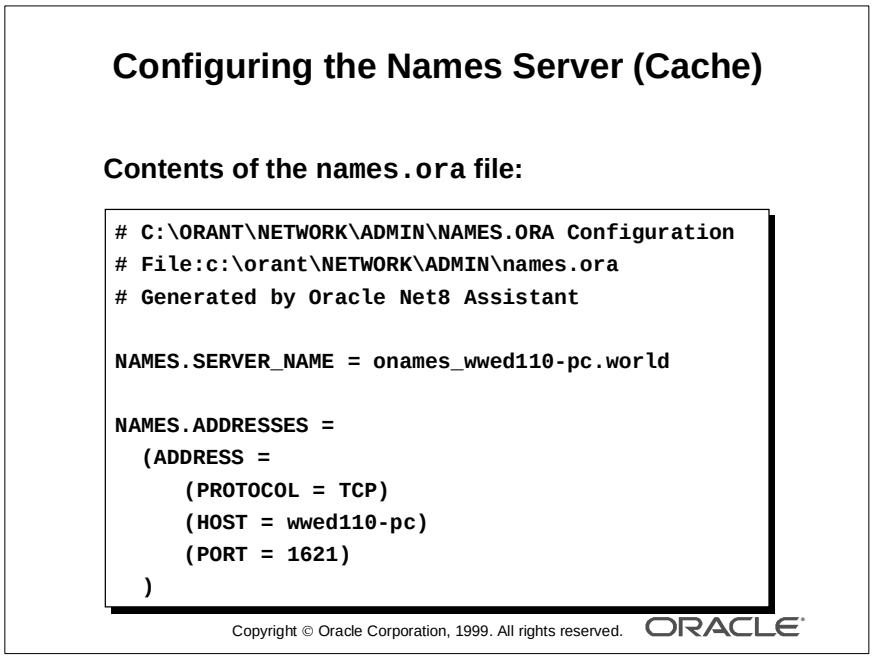
1. Select the "Names server is not first in its region" option button, and click the Next button.
2. If there are well-known Names servers in the region, select "Discover name servers in the region."

   This step may take some time, since an attempt is made to discover the different well-known Names servers in the region.

   Note: Well-known Names servers are addresses for one or more Oracle Names servers hardcoded into the Oracle Names server and its clients. Oracle Names servers then become available at these well-known addresses, so that clients do not need to be told, by way of configuration files, where to find the server.
3. If no well-known Names servers are discovered, you are prompted to select the protocol, and enter the host name and port number of the existing Names server in the region.

   This step may take some time, since an attempt is made to connect to the existing Names servers in the region.
4. Click the OK button when the message appears that other Names servers were discovered.
5. Select the Yes option button to acknowledge that the Names server is in the root region; otherwise select No, and follow the instructions of the Names Wizard.
6. Click the Finish button to complete the Names server basic configuration.

   The `names.ora` file is created at this point with default parameters.



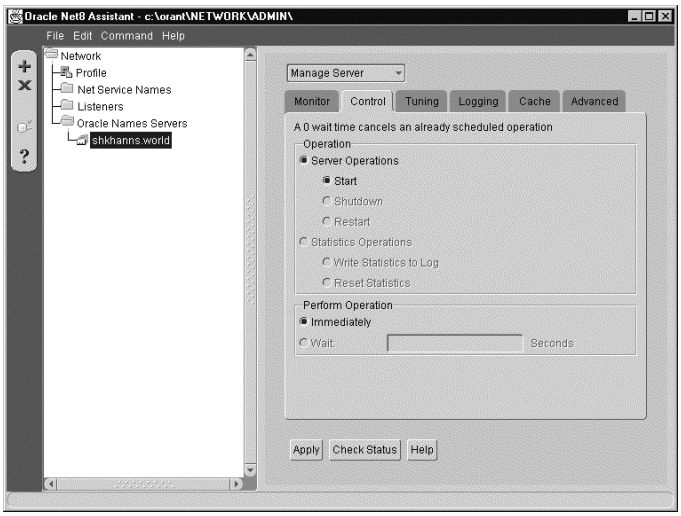
Viewing the Contents of the NAMES.ORA File

Browse the `names.ora` file in the directory that contains the configuration files.

On the NT operating system and UNIX operating system, this directory is `$ORACLE_HOME\network\admin`.

The file now reflects the new information for the newly created Names server.

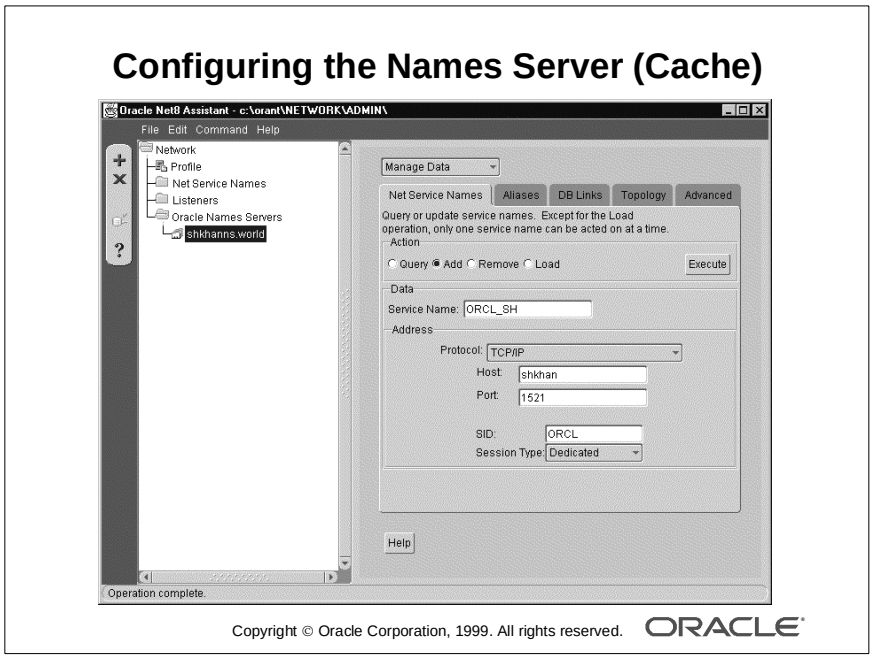
Configuring the Names Server (Cache)



Copyright © Oracle Corporation, 1999. All rights reserved.

Starting Up the Names Server

1. Select Manage Server from the pull-down menu.
2. Select the Start check box.
3. Click the Apply button.
4. Click OK when the dialog box appears indicating a successful startup.



Adding a Service Name

1. Select Manage Data from the pull-down menu.
2. Click the Add button.
3. Enter a service name for this Names server, its host name, and its port in the appropriate fields.

   Note: This is an example of how a service name is added. Add any other service names and their configuration data as required. In this class, you add the host name and port number of the database server you connect to, using the Names server to resolve the name. You can find this port number in the `listener.ora` file of the server you are to connect to.
4. Verify whether the SID, protocol, and server type values are correct.
5. Click Execute.

   The bottom right corner of Net8 Assistant indicates that the operation was complete.

   Note: When a service name is added to the Names server, there is no need to save the configuration, nor is there a need to restart the Names server. These changes are recorded dynamically when the Execute button is clicked.
6. Add more service names as required.

Note: When the Names server is shut down, service names are written to the `ckpreg.ora` file located in the `ORACLE_HOME\network\names` directory on NT or `ORACLE_HOME/network/names` directory on UNIX.

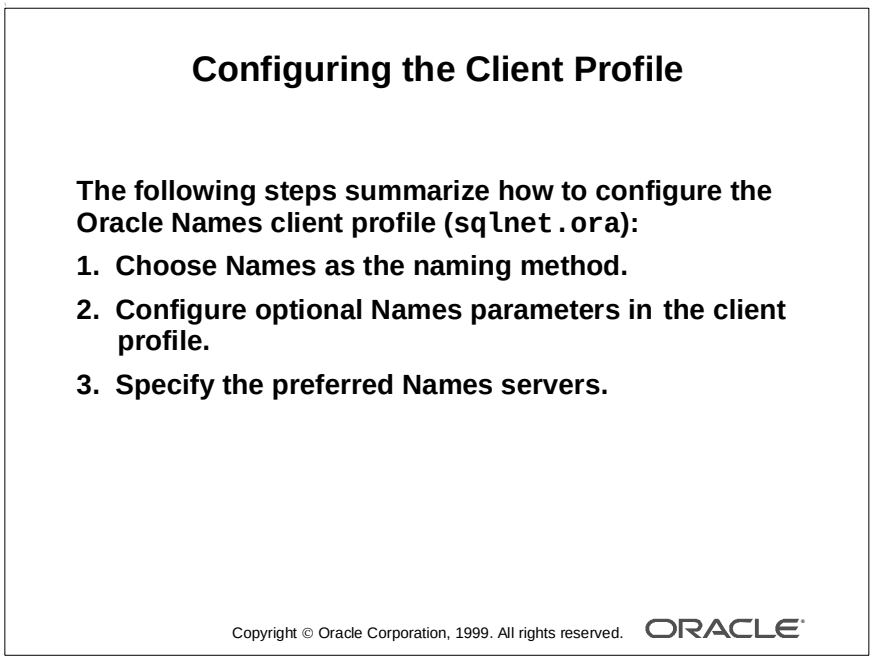
Deleting a Names Server on NT

1. Shut down the Names Server.
2. Click the Names Server that you want to delete.
3. Select Delete from the Edit menu.
4. Remove the `$ORACLE_HOME\network\ckp*.ora` file.
5. Remove the `$ORACLE_HOME\network\admin\names.ora` file.
6. Save the network configuration.
7. Restart Net8 Assistant.

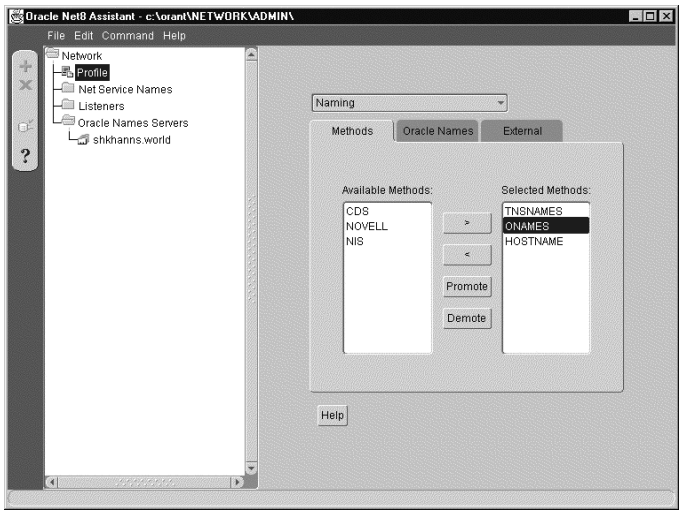
Note: On the NT operating system, the NT service for the Names server must be deleted; otherwise, a different name than the previous one of the Names server must be given to any newly created Names server.

If you delete the file mentioned in the preceding steps before you attempt to shut down the Names server, you will not be able to shut down the names server. You will need to restart the system.

Configuring the Client Profile



Configuring the Client Profile

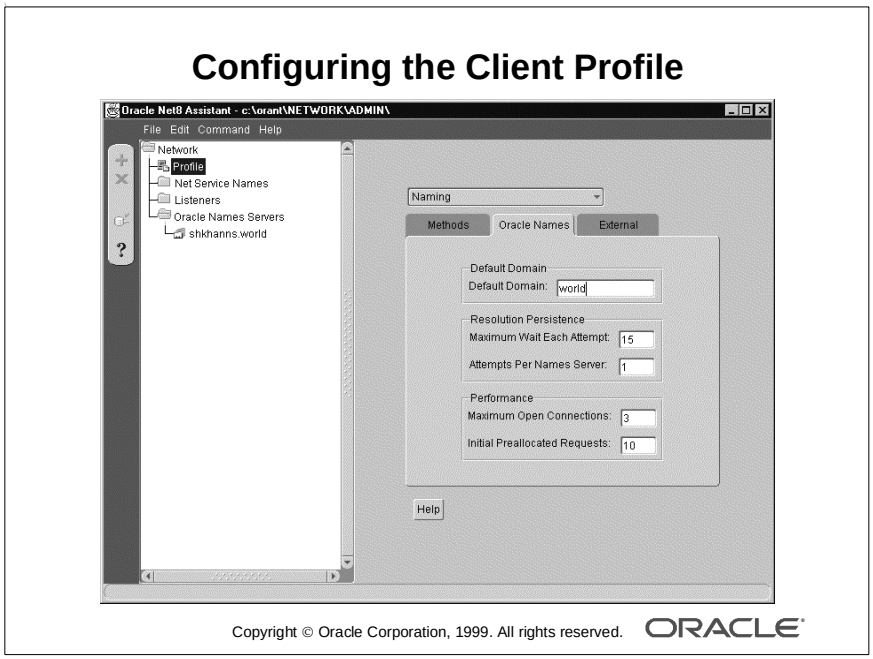


Copyright © Oracle Corporation, 1999. All rights reserved.

Choosing the Naming Method

1. Start up Net8 Assistant.
2. Click the Profile icon.
3. Add ONAMES to the selected methods if that method is not already selected.

Note: In the client profile (`sqlnet.ora` file), the parameter used to specify the naming method or methods is the `NAMES.DIRECTORY_PATH`.



Configuring Optional Names Parameters

1. Click the Oracle Names tab.
2. Enter your domain address (`my_domain.com`), such as oracle.com.

   Note: On UNIX, the domain is set to NULL (blank), whereas on NT it is set to your default domain.
3. Configure other optional Names client profile parameters as needed.

Optional Names Client Profile Parameters

- NAME.DEFAULT_DOMAIN

  This is your local domain. If set, this name is automatically appended to any unqualified service name in a Names server request.
- NAMES.INITIAL_RETRY_TIMEOUT

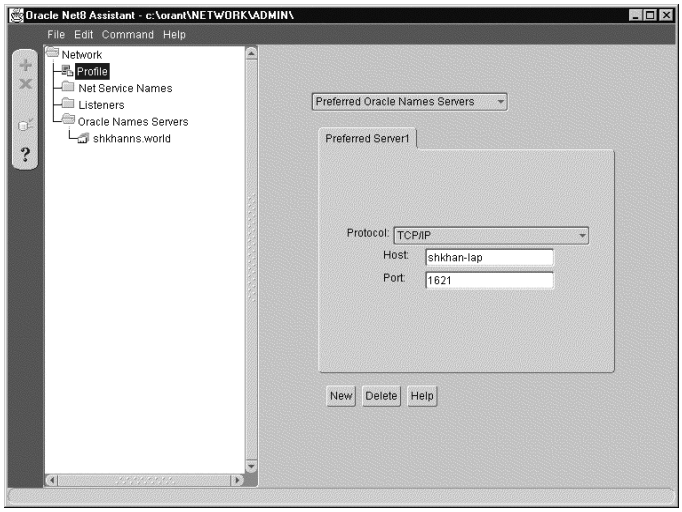
  The time a client is to wait for a response from a Names server before sending the request to another Names server. The default is 15 seconds.
- NAMES.MAX_OPEN_CONNECTIONS

  The number of connections the Names server can have open at one time. The default for this parameter is set to three connections.
- NAMES.MESSAGE_POOL_START_SIZE

  The number of messages in a client's message pool. These messages may be used for future requests to Names servers. By default, the pool is set to ten messages.
- NAMES.REQUEST_RETRIES

  The number of times a client attempts to connect to a Names server before the operation fails. The default is once.

Configuring the Client Profile



Copyright © Oracle Corporation, 1999. All rights reserved.

Specifying the Preferred Names Servers

1. Select Preferred Oracle Names Servers from the Profile pull-down menu.
2. Click the New button.
3. Choose the appropriate protocol.
4. Enter the host name of the preferred Names server.
5. Enter the port number at which the Names server is to listen for incoming requests.

   Note: This is the port number of the Names server, not the listener. In this class, this port number is the one you specified when creating the Names server. You can find this port number in the `names.ora` file.
6. Add any additional Names servers.
7. To save the network configuration, select “Save Network Configuration” from the File menu.

Configuring the Client Profile

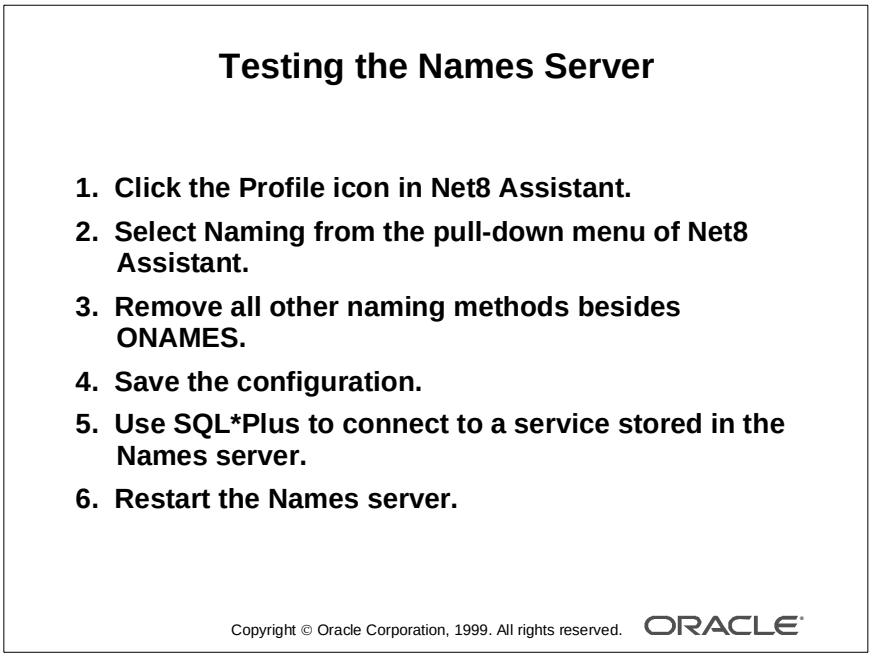
Contents of the `sqlnet.ora` file:

```
# C:\ORANT\NETWORK\ADMIN\SQLNET.ORA Configuration
# File:c:\orant\NETWORK\ADMIN\sqlnet.ora
# Generated by Oracle Net8 Assistant

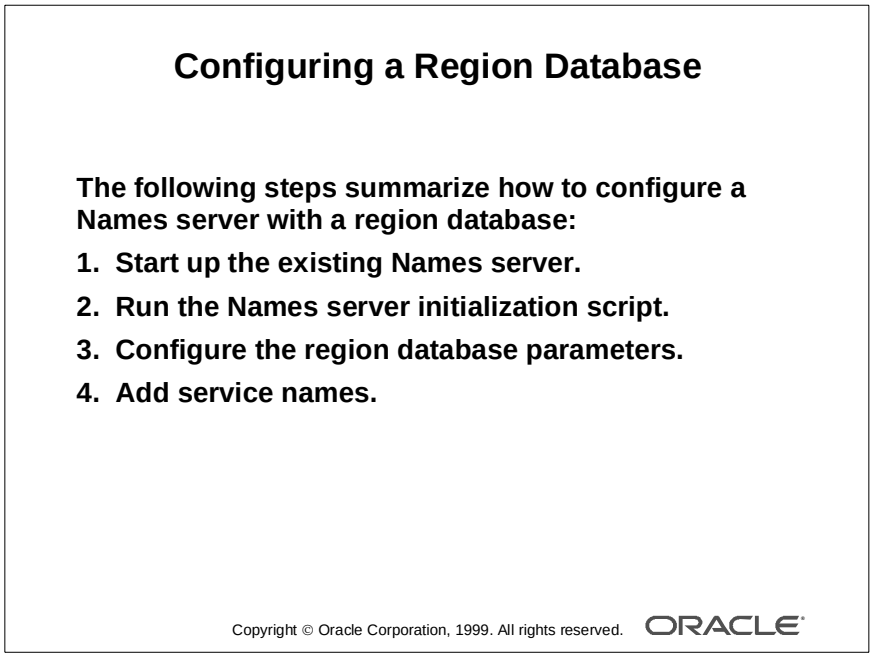
NAMES.PREFERRED_SERVERS =
  (ADDRESS_LIST =
     (ADDRESS =
         (PROTOCOL = TCP)
         (HOST = shkhan-lap)
         (PORT = 1621)  ))
NAMES.DEFAULT_DOMAIN = world
NAMES.DIRECTORY_PATH= (ONAMES)
```

Copyright © Oracle Corporation, 1999. All rights reserved.

Testing the Names Server



Configuring a Region Database



A Region Database

A region database is a repository for Names server information.

Running the Names Server Initialization Script

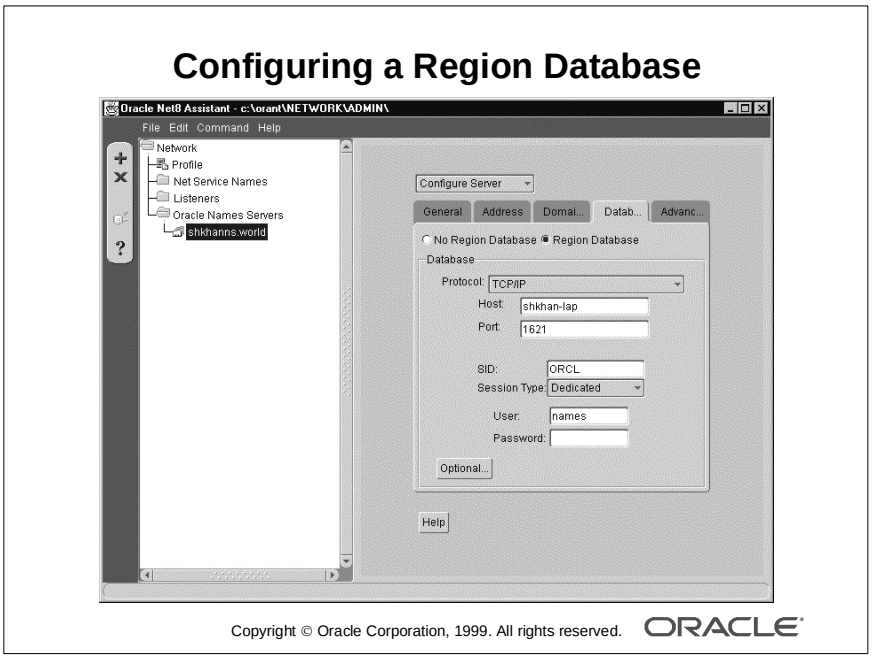
1. Change the directory to the names configuration directory.

   On UNIX, this directory is `$ORACLE_HOME/network/names`, and on NT it is `$ORACLE_HOME\network\names`.
2. Using SQL*Plus, connect as a user that you later use in Net8 Assistant to connect to the region database. Generally, a user called `names` is created by the administrator for this purpose.
3. From the SQL prompt, enter the following command for NT machines:

   ```
   SQL> @$ORACLE_HOME\network\names\namesini.sql
   ```

   On UNIX:

   ```
   SQL> @$ORACLE_HOME/network/names/namesini.sql
   ```

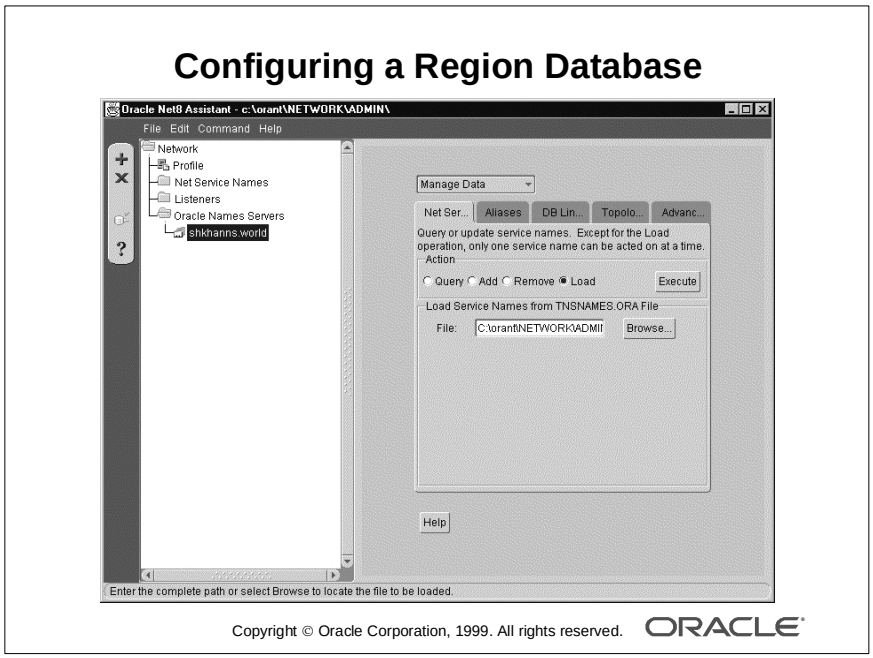


Configuring a Region Database

The following setup assumes that a Names server using cache replication is configured and running.

1. Select Configure Server from the pull-down menu for the appropriate Names server.
2. Click the Database tab.
3. Click the Region Database button.
4. Enter the host name and port number of the listener that is running on the machine that is configured as the region database.

   In this class, the region database is created on the UNIX server, so the port number is the same as the one used to configure the listener on the UNIX server (the port used in the `listener.ora` file on the UNIX server).
5. Enter the SID of the database that is to store the region information.
6. Enter the username (the `names` user) and password that are to be used to connect to the new region database.

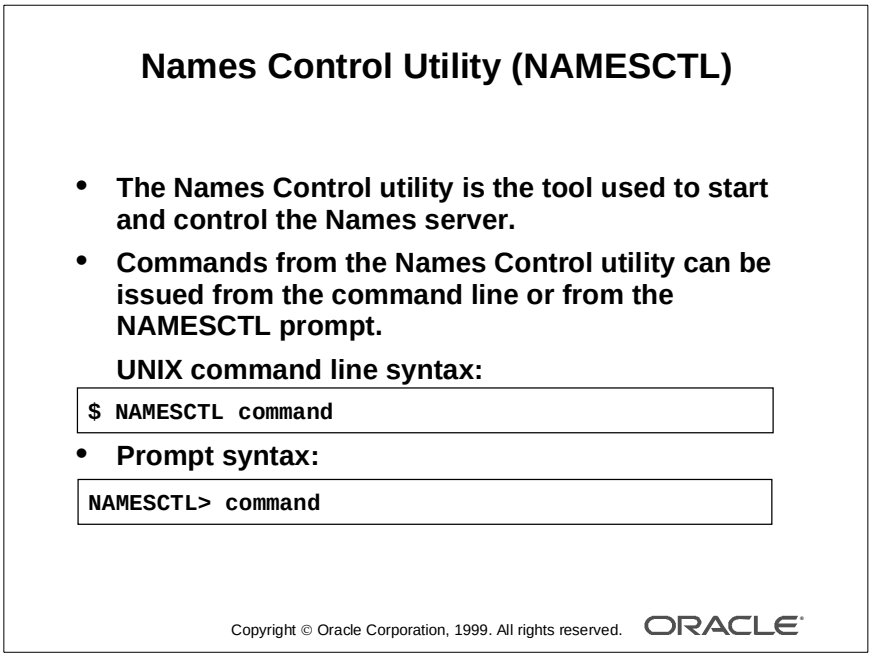


Adding Service Names

1. Restart the Names server.
2. Select Manage Data from the pull-down menu.
3. Click the Add button.
4. Enter a service name, host name, and port in the appropriate fields.
5. Verify whether the SID, protocol, and server type values are correct.
6. Click Execute.
7. Alternatively, service names can be read in from an existing `tnsnames.ora` file by using the LOAD command. You must type in the explicit path and file name of the `tnsnames.ora` file for the names to successfully load in the region database.

   Note: When a service name is added to the Names server, there is no need to save the configuration. These changes are recorded dynamically when the Execute button is clicked.
8. Test the region database by connecting to the newly added service in the preceding steps.

Names Control Utility (NAMESCTL)



NT Platform Command Line Syntax

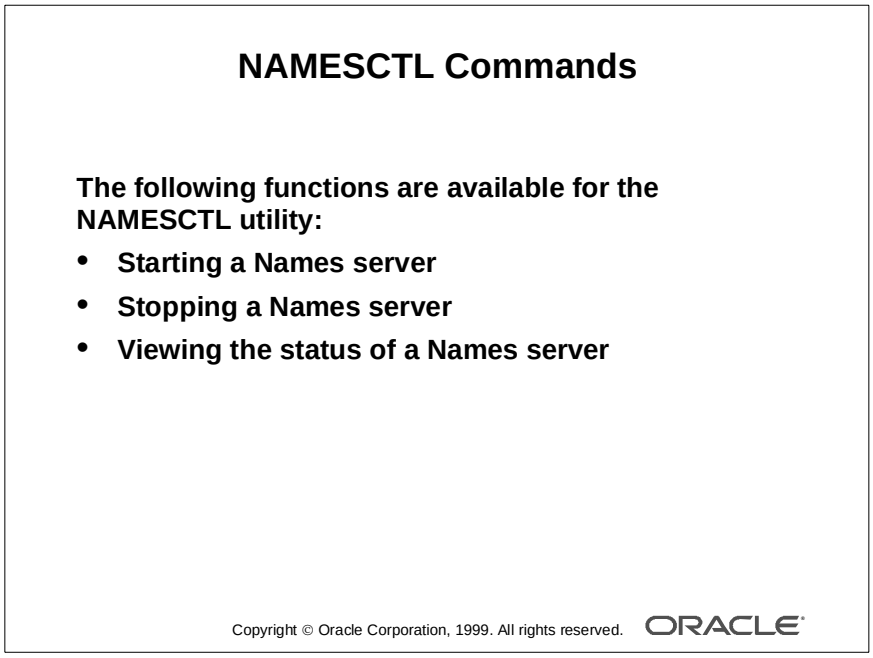
On the NT operating system, the following denotes the Names Control utility command and its syntax:

```
C:\> NAMESCTL command
```

The prompt syntax on UNIX is the same as for NT.

Note: In Oracle8, the command for the Names Control utility is NAMESCTL80.

NAMESCTL Commands



Starting a Names Server

To start up a Names server, issue the following command:

```
NAMESCTL> START
```

Stopping a Names Server

To shut down a Names server, issue the following command:

```
NAMESCTL> STOP
CONFIRM [YES OR NO]:
```

Viewing the Status of a Names Server

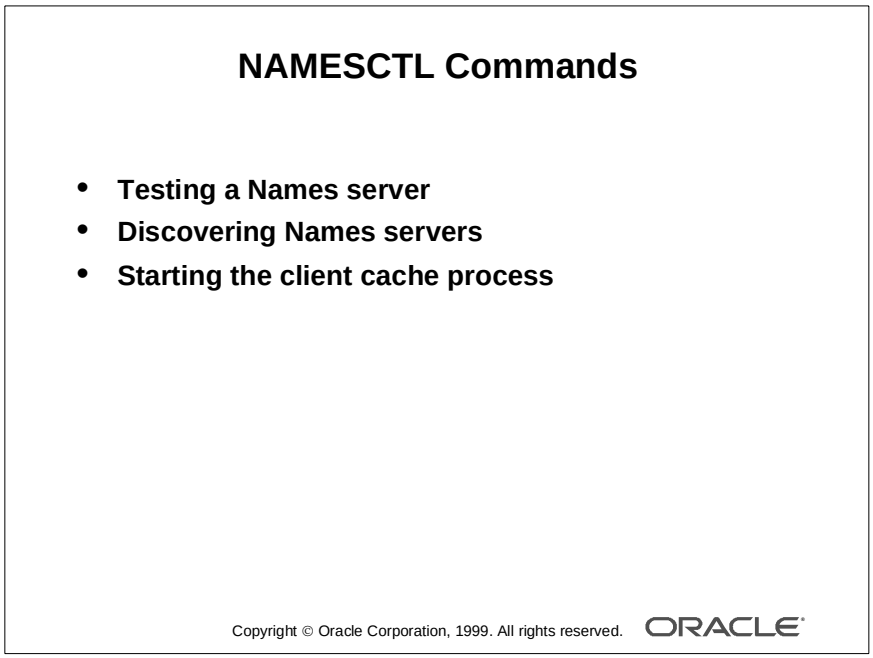
To view status information, such as the name and location of the `names.ora` file, the trace file, the log file, and other status information, the following status command is used:

```
NAMESCTL> STATUS
```

Note: Only one Names server is allowed per machine.

Example of the Status Output

```
NAMESCTL> Status
Server name:                          onames_wwed110-pc
Server has been running for:          1 hour 30 minutes 11.42
                                      seconds
Request processing enabled:           yes
Request forwarding enabled:           yes
Requests received:                    5
Requests forwarded:                   0
Foreign data items cached:            0
Region data next checked for reload in: not set
Region data reload check failures:    0
Cache next checkpointed in:           not set
Cache checkpoint interval:            not set
Cache checkpoint file name:
C:\ORANT\NETWORK\names\ckpcch.ora
Statistic counters next reset in:     not set
Statistic counter reset interval:     not set
Statistic counters next logged in:    not set
Statistic counter logging interval:   not set
Trace level:                          0
Trace file name:
C:\ORANT\NETWORK\trace\names83.trc
Log file name:
C:\ORANT\NETWORK\log\names.log
System parameter file name:
C:\ORANT\NETWORK\admin\names.ora
Command-line parameter file name:     ""
Administrative region name:           ""
Administrative region description:    ""
ApplTable Index:                      0
Contact                               ""
Operational Status                    0
Save Config on Stop                   no
NAMESCTL>
```



Testing a Names Server

To test if a Names server is up and running, the PING command is used. This command returns the time it takes to contact a Names server and an acknowledgment. The following is an example of the PING command attempting to test the default Names server:

```
NAMESCTL> ping nserver1.oracle.com
roundtrip time is 0.02 seconds
```

Discovering Names Servers

The REORDER_NS command creates the file that lists local Names servers and their addresses. It does this by using the Names server discovery process described earlier.

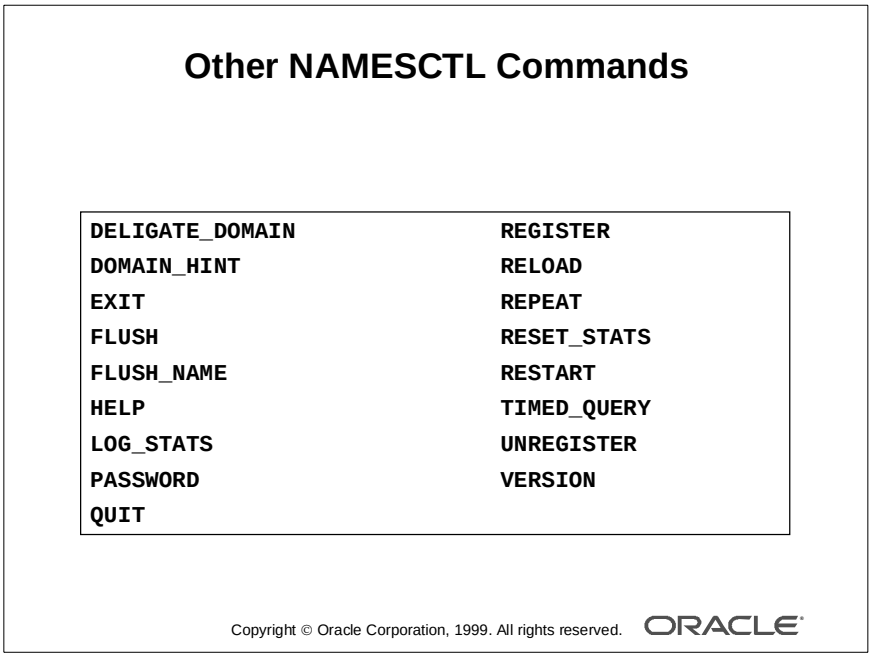
```
NAMESCTL> reorder_ns
```

Starting the Client Cache Process

The START_CLIENT_CACHE command starts the client cache daemon process. The client cache daemon process must be stopped before this command is issued, and a Names server list must exist. The Names server list is created with the REORDER_NS command.

```
NAMESCTL> start_client_cache
```

Other NAMESCTL Commands



Other NAMESCTL Commands

- DELEGATE_DOMAIN: Defines a domain as the start of a subregion of the current region.
- DOMAIN_HINT: Provides the Names servers in the current region with the name and address of a Names server in another region.
- EXIT: Closes the NAMESCTL utility.
- FLUSH: Drops all stored nonauthoritative data from the Names server cache.
- FLUSH_NAME: Drops one or more specific nonauthoritative names from the cache of the current Names server.
- HELP: Provides details about the NAMESCTL command.
- LOG_STATS: Logs the current set of Names server statistics to the configured log file for that Names server.

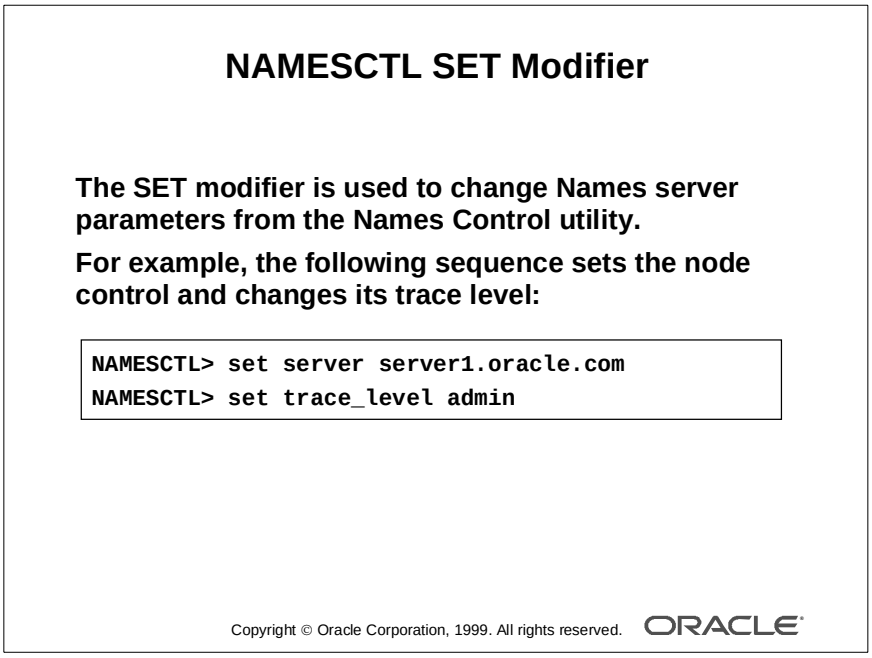
Note: Further details on the syntax and examples of these commands can be found in "Appendix A" of *Net8 Administrator's Guide.*

Other NAMESCTL Commands (continued)

- PASSWORD: Registers the password for privileged Names server operations such as reload and stop.
- QUIT: Quits the NAMESCTL program.
- REGISTER: Registers a network object to a Names server.
- RELOAD: Forces the server to check immediately for data changes in its administrative region and, if there are any, reloads all database service names, global database links, and aliases.
- REPEAT: Used to perform query, register, timed_query, or unregister multiple times to compute average return rates.
- RESET_STATS: Resets the Names server statistics to the original values of the Names server at startup.
- RESTART: Initiates a reset of a Names server to its original state at startup.
- TIMED_QUERY: Shows all registered data in the Names server cache.
- UNREGISTER: Removes a network object from the Names server.
- VERSION: Displays the current version and name of the Names server.

Note: Further details on the syntax and examples of these commands can be found in "Appendix A" of *Net8 Administrator's Guide.*

NAMESCTL SET Modifier

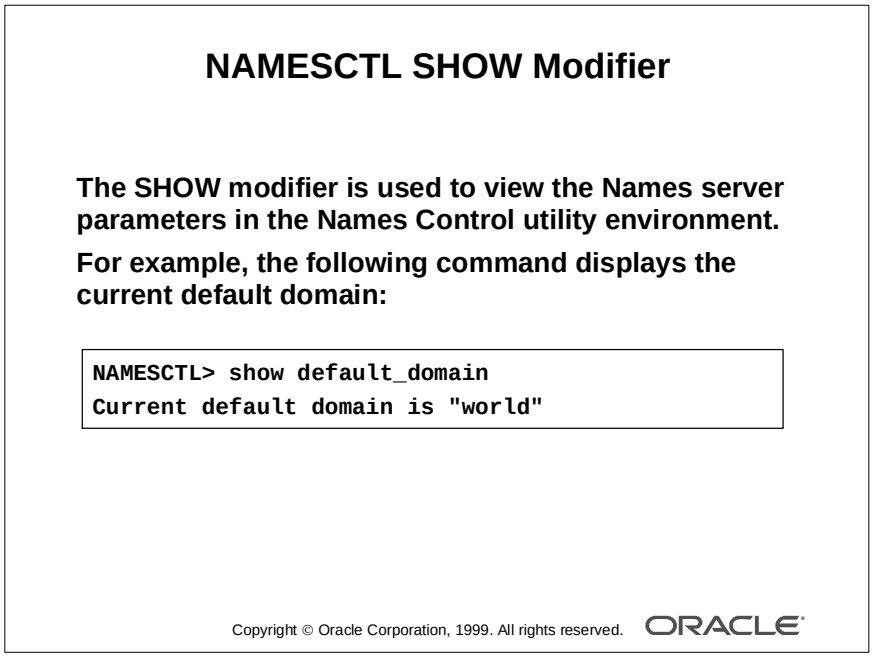


NAMESCTL SET Modifier

- SET CACHE_CHECKPOINT_INTERVAL: Sets the frequency with which all collected information about foreign regions is saved in the Names server cache file.
- SET DEFAULT_DOMAIN: Sets or changes the default domain for the NAMESCTL client.
- SET FORWARDING_AVAILABLE: Turns on or off name request forwarding for a Names server.
- SET LOG_FILE_NAME: Changes the log file name.
- SET LOG_STATS_INTERVAL: Changes the frequency with which the statistics are logged to the log file.
- SET NAMESCTL_TRACE_LEVEL: Sets the level at which the NAMESCTL program can be traced.
- SET PASSWORD: Registers the password for privileged Names server operations such as reload and stop.
- SET REQUESTS_ENABLED: Determines whether the current Names server responds to requests.
- SET RESET_STATS_INTERVAL: Changes the time between the statistics being reset to zero or initial values in the current server.
- SET SERVER: Changes the current Names server.
- SET TRACE_FILE_NAME: Changes the trace destination file name.
- SET TRACE_LEVEL: Changes the TRACE_LEVEL for tracing the current Names server.

Note: Further details of this command can be found in *Net8 Administrator's Guide*.

NAMESCTL SHOW Modifier



NAMESCTL SHOW Modifier

- SHOW CACHE_CHECKPOINT_INTERVAL: Shows the frequency with which the cache of the Names server is written to the checkpoint file.
- SHOW DEFAULT_DOMAIN: Displays the default domain for the NAMESCTL client.
- SHOW FORWARDING_AVAILABLE: Shows whether the Names server is forwarding requests for foreign names or redirecting them.
- SHOW LOG_FILE_NAME: Shows the name of the file in which the Names Server writes the logging information.
- SHOW LOG_STATS_INTERVAL: Displays the frequency with which the statistics are logged to the log file.
- SHOW NAMESCTL_TRACE_LEVEL: Displays control of the level at which the NAMESCTL program is being traced.
- SHOW REQUESTS_ENABLED: Shows whether or not the Names server is responding to requests.

Note: Further details of this command can be found in *Net8 Administrator's Guide.*

NAMESCTL SHOW Modifier (continued)

- SHOW RESET_STATS_INTERVAL: Shows how often the statistics are dumped to the log file.
- SHOW SERVER: Displays the current Names server.
- SHOW STATUS: Displays the general status information about the Names server.
- SHOW SYSTEM_QUERIES: Displays the next occurrence of all system queries.
- SHOW TRACE_FILE_NAME: Displays the TRACE_FILE_NAME of the current server.
- SHOW TRACE_LEVEL: Displays the TRACE_LEVEL for the current Names server.
- SHOW VERSION: Displays the current version and name of the Names server.

Note: Further details of this command can be found in *Net8 Administrator's Guide*.

Summary

Summary

In this lesson, you should have learned:

- **A Names server is a centrally located service by which service names are resolved to connect strings.**
- **`names.ora` is the configuration file for the Names server.**
- **The Names server can be configured by either manually modifying the `names.ora` file or by using Net8 Assistant.**

Copyright © Oracle Corporation, 1999. All rights reserved.

Summary

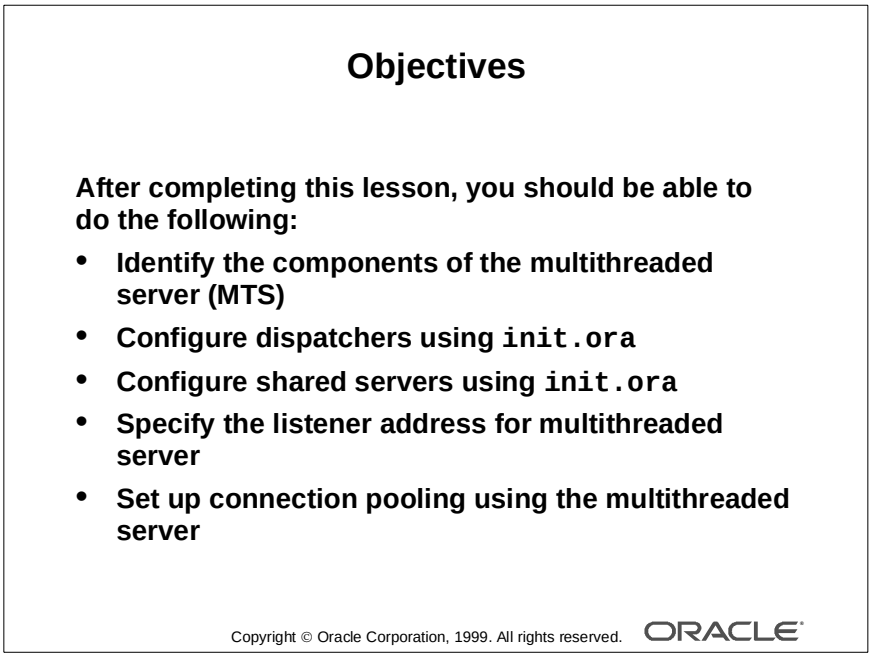
- **NAMESCTL is the utility used to configure the Names server.**
- **A region database is a database repository of service names stored by the Names server.**
- **The NAMESCTL SET and SHOW modifiers are used to change and display the settings of the Names server, respectively.**

Copyright © Oracle Corporation, 1999. All rights reserved.

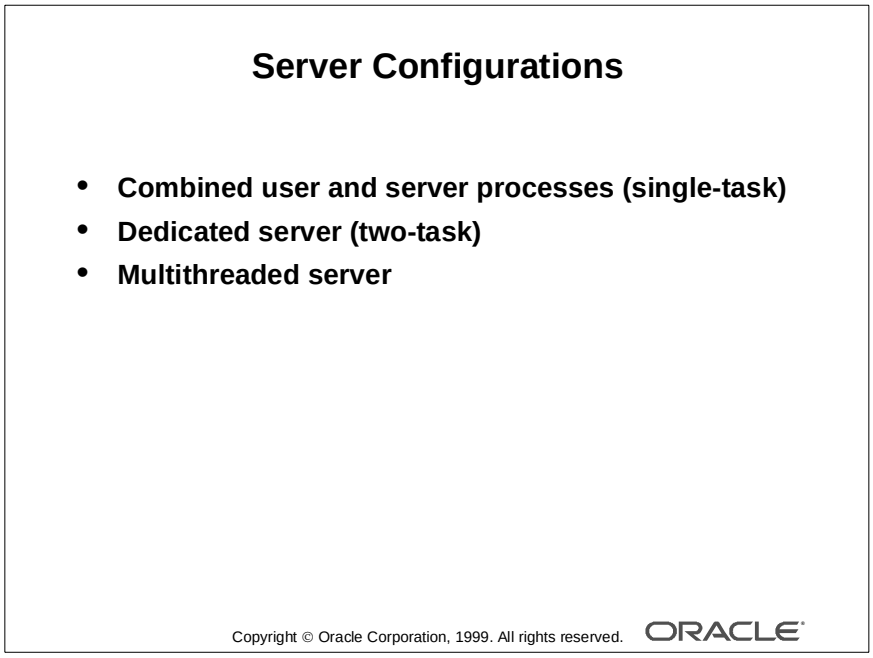
7

Multithreaded Server Usage and Configuration

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Identify the components of the multithreaded server (MTS)
• Configure dispatchers using init.ora
• Configure shared servers using init.ora
• Specify the listener address for multithreaded server
• Set up connection pooling using the multithreaded server
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Server Configurations



Oracle Server Configuration Options

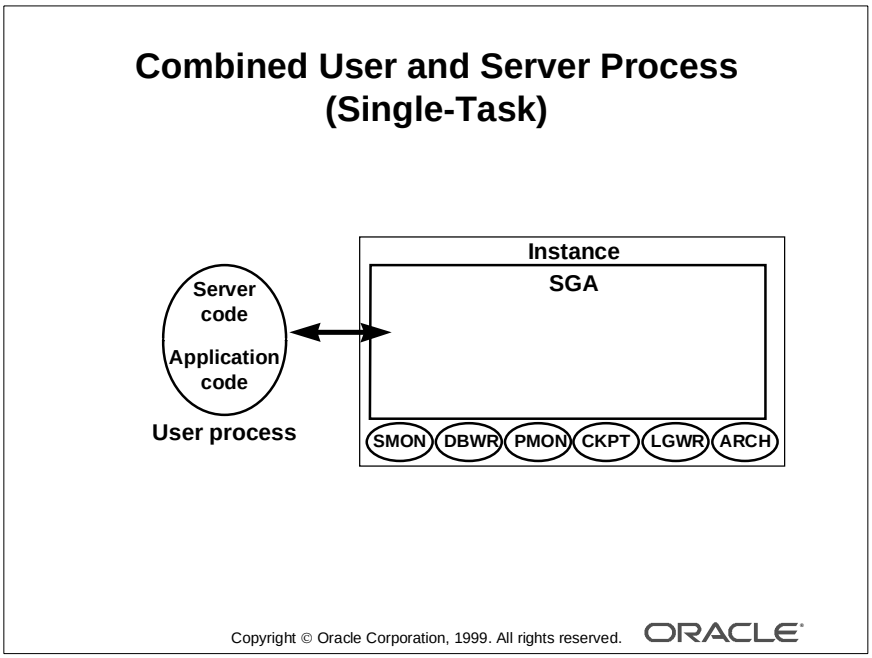
The Oracle8 and Oracle8*i* Servers can be configured in different ways to vary the number of user processes per server process.

You can configure the servers in three possible combinations:

Combined User and Server Processes The user and server processes are combined into a single user process. This is also known as *single-task architecture*. This configuration is only allowed on certain platforms.

Oracle with Dedicated Servers A dedicated server process handles requests for a single user process. This is also known as *two-task architecture*.

Oracle with Multithreaded Servers With a multithreaded server, a dispatcher process enables many user processes to share a few server processes. This is also known as a multithreaded server (MTS) configuration.

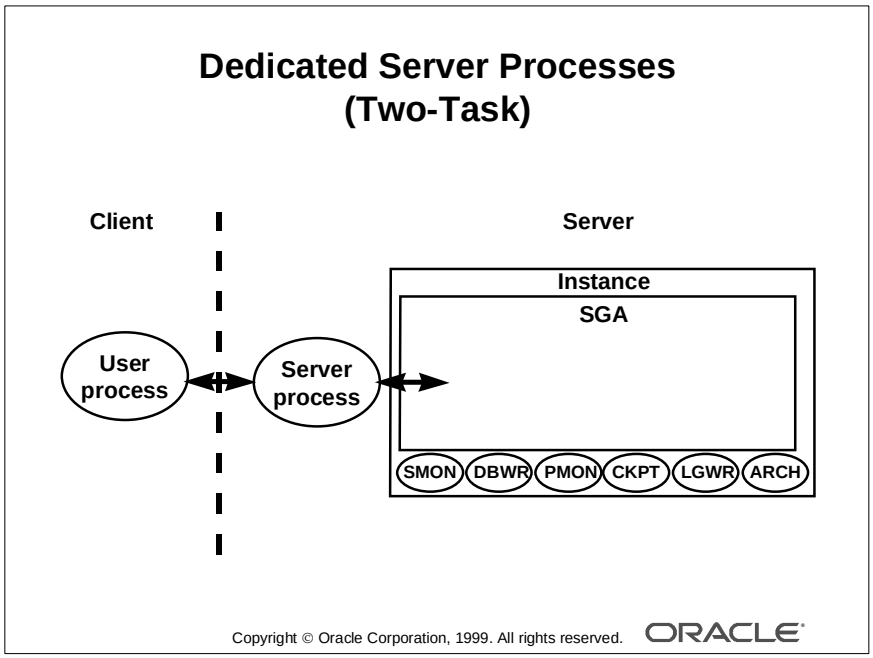


Single-Task Oracle

When one process executes both the application code and the Oracle code, this configuration is called *single-task Oracle*.

Single-Task Processes

- The operating system must be able to maintain separation of the application code and the Oracle code to prevent the application code from damaging the Oracle code. An example of an operating system that can maintain this separation is Open VMS.
- In contrast, UNIX cannot provide this separation and so must have separate processes to run the application code and the Oracle code.
- The process executing the application code is also called the *user process*. A process can make only one Oracle connection at any time, except in an SQL*Net environment where multiple connections can be maintained.



Two-Task Oracle

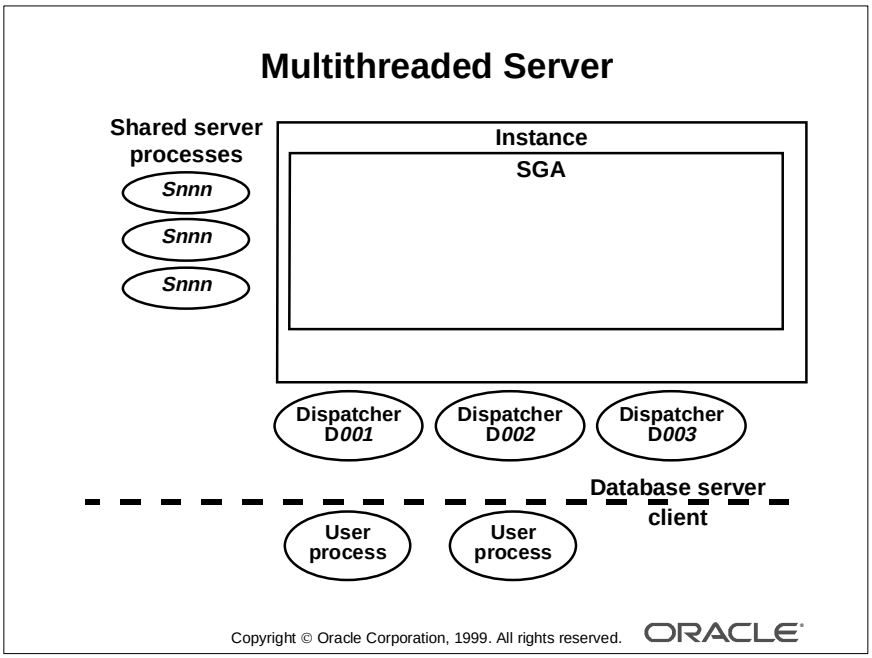
If the user and server processes are separate, the term *two-task* is used. In two-task Oracle, the servers can be shared or dedicated. Two-task simply means the database application code and the Oracle code are executed by different processes.

Dedicated Server Processes

- The user process and server process are separate.
- Each user process has its own server process.
- The user and server processes can run on different machines to take advantage of distributed processing.
- There is a one-to-one ratio between the user and server processes.
- Even when the user process is not making a database request, the dedicated server exists but remains idle.
- The dedicated server process is sometimes referred to as a *shadow* process, because it is acting on behalf of one user process only.

The example shown is also applicable if the user process and server process are running on the same machine in two-task mode, as with UNIX.

The program interface in use here depends on whether the user and the dedicated server processes are on the same machine. If they are, the host operating system's interprocess communication mechanism is used for the program interface between processes.



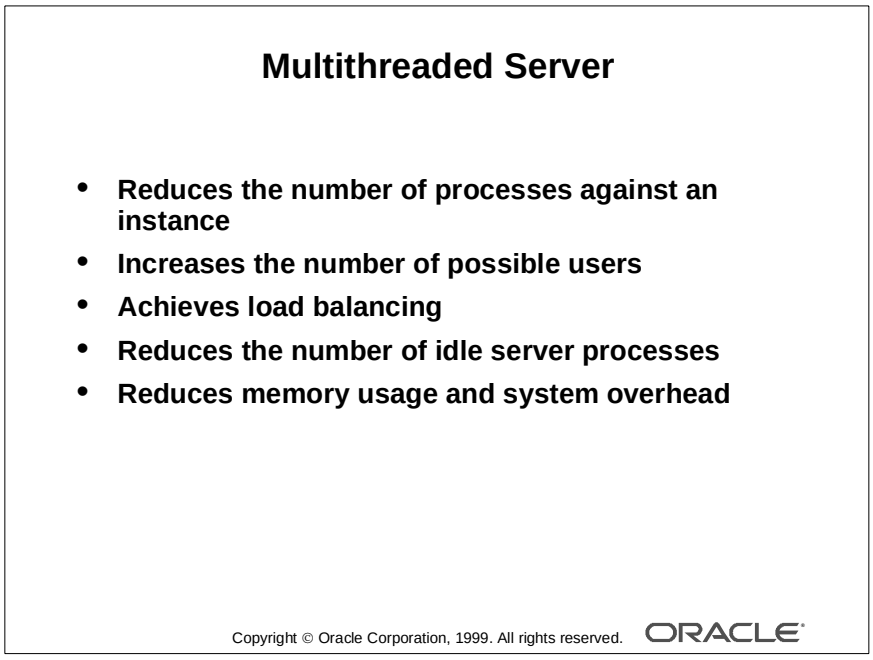
The Oracle Multithreaded Server

The Oracle multithreaded server configuration enables shared servers, dedicated servers, and combined users and servers to exist within the same instance.

- In an online transaction processing environment (OLTP), for example, order entry applications, in which users enter data through an application interface, the server process could be idle 90 percent or more of the connected time.
- MTS improves server efficiency, because any server can process an incoming request, rather than wait for a specific server to process work on a request.
- When using MTS, you can support many more users than you can in the dedicated server configuration with the same number of servers, because users in the MTS architecture share the server processes, and thus, fewer server processes can be configured.

Technical Note

If configuring MTS on Windows NT, dispatchers can use only the TCP/IP protocol.



Using Multithreaded Server

- Server processes are shared among user processes.
- A user can still request a dedicated server.

MTS architecture reduces memory usage by reducing the number of server processes required. For example, for 100 users, 100 server processes are needed for dedicated server connections. With MTS, you may need only 10 shared server processes for the 100 users.

When to Use a Dedicated Server

- Submitting batch jobs (it is expected that there will be little or no idle time)
- Connecting with Server Manager to start up, shut down, or perform recovery
- Connecting as internal

Technical Note

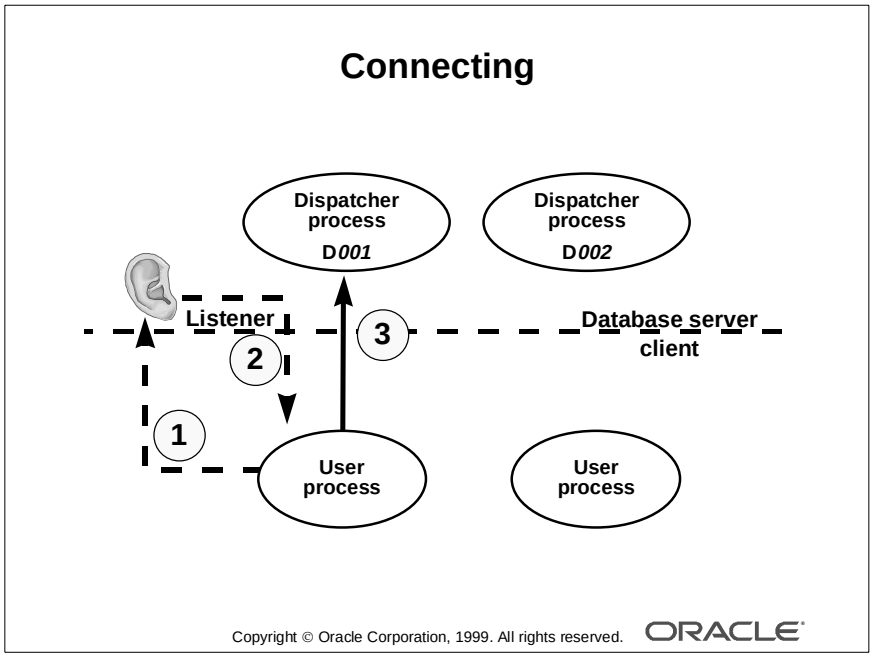
For most platforms, if your machine has plenty of memory to support dedicated servers, you should use that configuration. In this situation, performance is likely to be better.

There are exceptions such as NT, in which performance may improve using the multithreaded server configuration due to the asynchronous nature of MTS.

To request a dedicated server, the clause SERVER=DEDICATED must be included in the SQL*Net TNS connection string within the `tnsnames.ora` file:

```
TST8i.world =
  (DESCRIPTION =
    (ADDRESS =
          (PROTOCOL = TCP)
          (HOST = wwed151-sun)
          (PORT = 1521)
    )
    (CONNECT_DATA = (SID = TST8i)
          (SERVER=DEDICATED)
    )
  )
```

Multithreaded Server Architecture



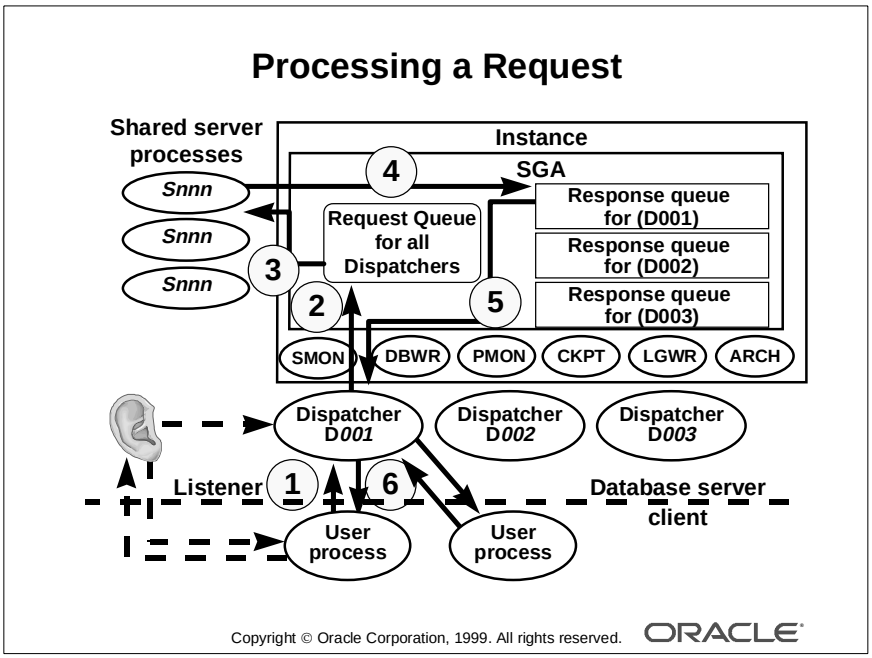
Connecting to the Multithreaded Server

1. The listener process waits for any connection requests from a user process. When a process requests a connection, the listener determines whether to connect the user process a dispatcher (depending on the load of the dispatcher) or assign it a dedicated server process.
2. If the user process can connect to a dispatcher, the listener gives the user process the address of a dispatcher process. If the user process requests a dedicated server, the listener creates a dedicated server process and connects the user process to it.
3. Once the connection has been established, either through a dispatcher or a dedicated server process, the connection is maintained for the duration of the session.

Technical Note

If the user call is from across a network, the dispatcher process chosen by the listener must match the protocol of the network being used.

If the user connects as OPS$ on a remote client, remember to set the initialization parameter REMOTE_OS_AUTHENT to TRUE.



Processing a Request

1. A user sends a request to its dispatcher.
2. The dispatcher places the request into the request queue in the System Global Area (SGA).
3. A shared server picks up the request from the request queue and processes the request.
4. The shared server places the response on the calling dispatcher's response queue.
5. The response is handed off to the dispatcher.
6. The dispatcher returns the response to the user.

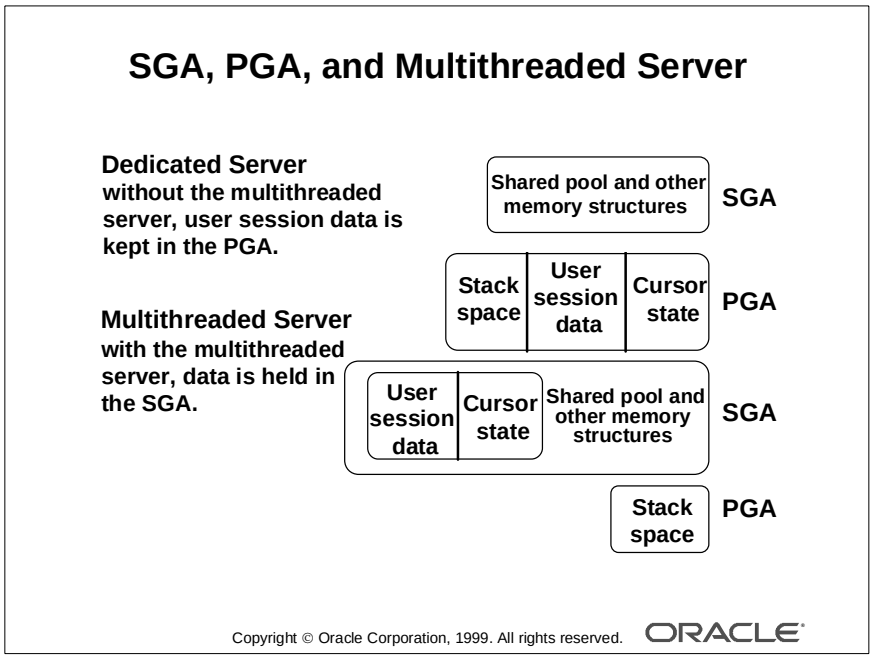
Once the user call has been completed, the shared server process is released and is available to service another user call in the request queue.

Request Queue

- One request queue is shared by all dispatchers.
- Shared servers monitor the request queue for new requests.
- Requests are processed on a first-in, first-out basis.

Response Queue

- Shared servers place all completed requests on the calling dispatcher's response queue.
- Each dispatcher has its own response queue in the SGA.
- Each dispatcher is responsible for sending completed requests back to the appropriate user process.
- Users are connected to the same dispatcher for the duration of a session.



The SGA and PGA

The contents of the System Global Area (SGA) and the Program Global Area (PGA) differ when dedicated servers or shared servers are used.

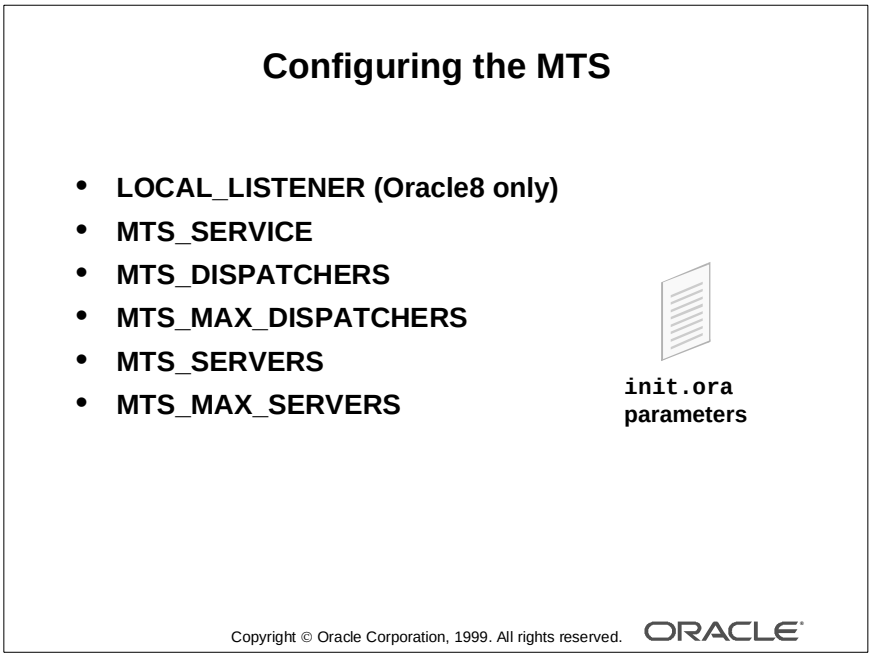
- Text and parsed forms of all SQL statements are stored in the SGA.
- The cursor state contains run-time memory values for the SQL statement, such as rows retrieved.
- User session data includes security and resource usage information.
- The stack space contains local variables for the process.

Technical Note

The change in the SGA and PGA is transparent to the user; however, if supporting multiple users, you need to increase the SHARED_POOL_SIZE per connection.

Each shared server process needs to access the data spaces of all sessions so that any server can handle requests from any session. Space is allocated in the SGA for each session's data space. You can limit the amount of space that a session can allocate by setting the resource limit PRIVATE_SGA to the desired amount of space in the user profile.

Configuring the Multithreaded Server



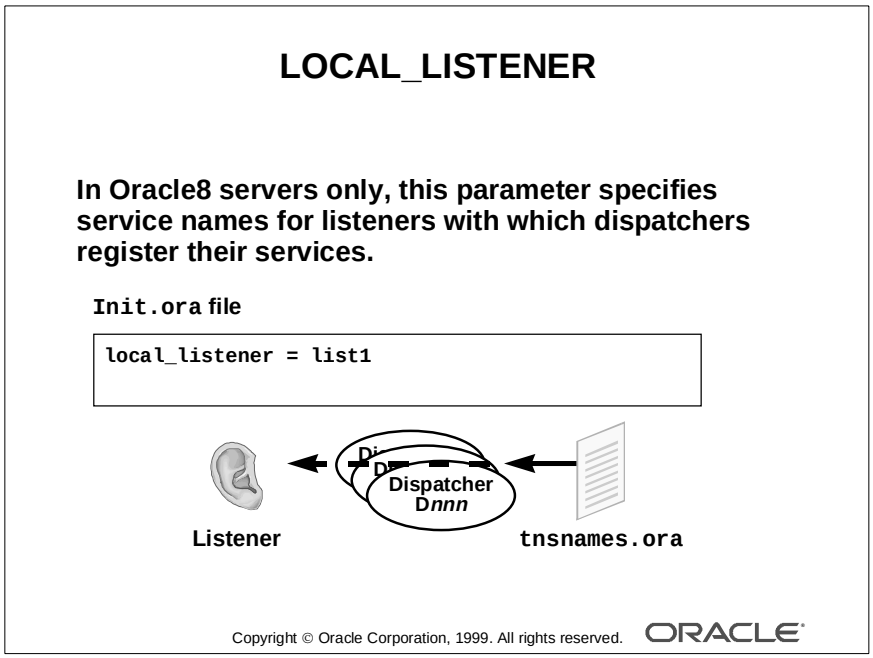
Configuring the MTS

To configure the multithreaded server, you need to edit the initialization parameter file for your instance.

- After setting these initialization parameters, restart the instance, which at this point will use the multithreaded server configuration.
  The multithreaded server architecture requires Net8.
- User processes targeting the multithreaded server must connect through Net8, even if they are on the same machine as the Oracle instance.

Technical Note

Most of the parameters have sensible defaults. On many systems, the only parameters that should be configured are MTS_SERVERS and MTS_DISPATCHERS.



The LOCAL_LISTENER Parameter

In Oracle8 servers, the LOCAL_LISTENER parameter specifies the service name defined in `tnsnames.ora` of either a single name or an address list of Net8 listeners.

Note: Oracle8*i* does not require the LOCAL_LISTENER parameter and the `tnsnames.ora` configuration associated with this parameter.

So if using the example in the slide for the local_listener, you would have to have an entry in your `tnsnames.ora` file on the node on which the listener is running that has the following entry:

```
list1 =
(DESCRIPTION =
   (ADDRESS =
      (PROTOCOL = TCP)
      (HOST = ed-testsun1)
      (PORT = 1720)
)
(CONNECT_DATA=(SID=DBA1)))
```

Note: If the connection fails**,** try modifying the `tnsnames.ora` file to look like the preceding example, including spaces.

The LOCAL_LISTENER Parameter (continued)

- The Net8 listeners need to be running on the same machine as the instance.
- The instance and dispatchers register certain information with the listener. The listener uses this information to connect clients to the appropriate dispatchers and dedicated servers.

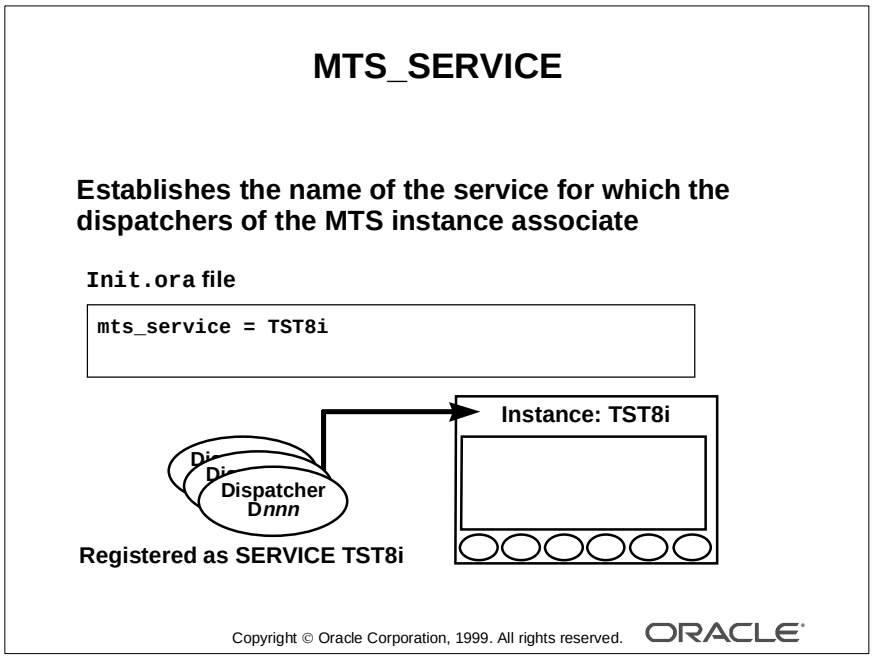
Parameter Type	**String**
Parameter class:	Static (cannot use ALTER SYSTEM to modify)
Default value:	If no service name address is specified, the behavior is as follows: First looks for default PLUG and PLAY listener, then looks for default TCP/IP port 1521, and last the ORACLE_SID value

Technical Note

MTS_LISTENER_ADDRESS and MTS_MULTIPLE_LISTENERS should be considered obsolete parameters.

They are maintained for the current release, and are included for backward compatibility.

When the LOCAL_LISTENER parameter is present, it overrides the MTS_LISTENER_ADDRESS and MTS_MULTIPLE_LISTENERS parameters.



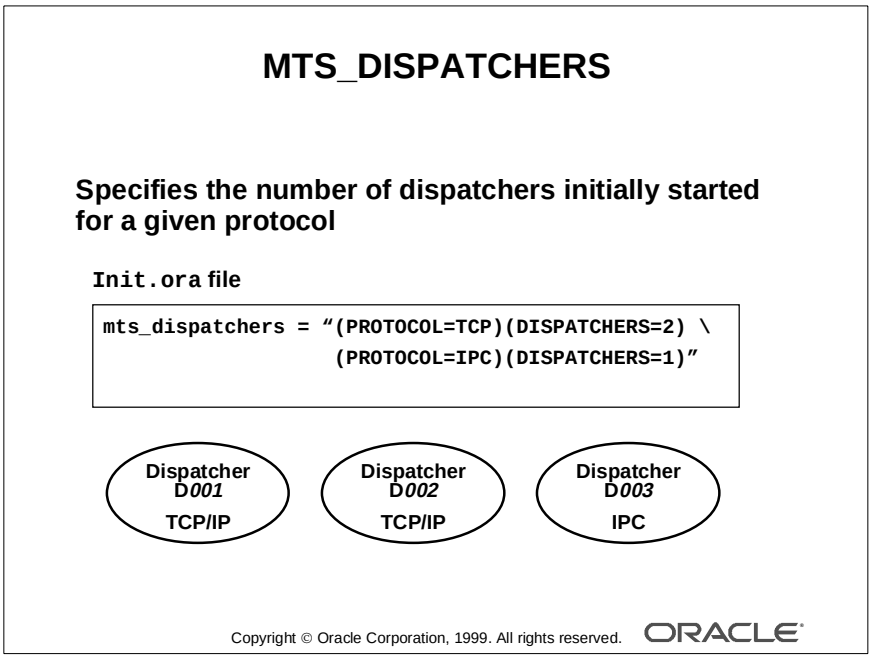
The MTS_SERVICE Parameter

MTS_SERVICE specifies the name of the database you want to be associated with the dispatcher.

Oracle always checks for such a service before establishing a normal database connection.

Parameter Type	String
Parameter class:	Static (cannot use ALTER SYSTEM to modify)
Default value:	Value of DB_NAME initialization parameter

The name you specify must be unique. It should not be enclosed in quotation marks. It is a good idea for this name to be the same as the instance name. That way, if the dispatcher is unavailable for any reason, the CONNECT string still connects the user to the database.



The MTS_DISPATCHERS Parameter

The database administrator uses MTS_DISPATCHERS to enable various attributes for each dispatcher.

Oracle8 supports a name-value syntax (similar to the syntax used by Net8) to enable the specification of the existing and additional attributes in a position-independent, case-insensitive manner.

For example:

```
MTS_DISPATCHERS = "(PROTOCOL=TCP)(DISPATCHERS=3)"
```

Parameter Type	**String (Specify as a quoted string)**
Parameter class:	Dynamic (can use ALTER SYSTEM to modify)
Default value:	NULL

The MTS_DISPATCHERS Parameter (continued)

One and only one of the following attributes is required: ADDRESS, DESCRIPTION, or PROTOCOL.

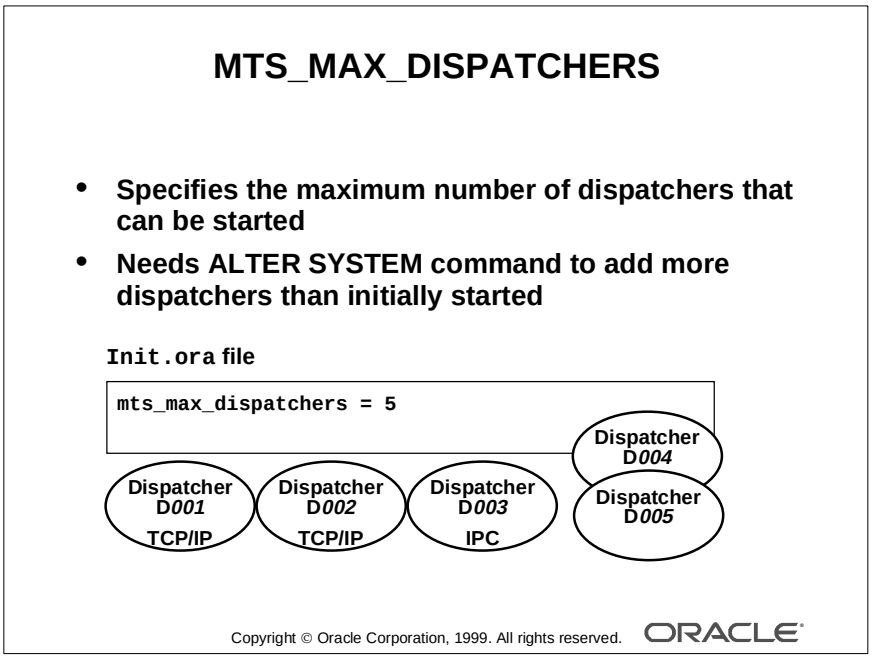
Attribute	Description
PROTOCOL (PRO or PROT)	The network protocol for which the dispatchers listen
ADDRESS (ADD or ADDR)	The network address (in Net8 syntax) on which the dispatchers listen (Includes the protocol)
DESCRIPTION (DES or DESC)	The network description (in Net8 syntax) of the end point on which the dispatchers will listen (Includes the protocol)
DISPATCHERS (DIS or DISP)	The initial number of dispatchers to start (default is 1)
SESSIONS (SES or SESS)	The maximum number of network sessions for each dispatcher
LISTENER (LIS, LIST)	The network name of an address or address list of the Net8 listeners with which the dispatchers register (The listener or listeners can reside on other nodes.)
SERVICE (SER, SERV)	The service name that the dispatchers register with the Net8 listeners (The SERVICE attribute overrides the MTS_SERVICE parameter. This attribute specifies a service name that the dispatchers use to register.)

Note: Further details on the MTS_DISPATCHERS parameter can be found in the "Initialization Parameters" section in the *Oracle8i Reference Manual*

Calculating the Initial Number of Dispatcher Processes

Once you know the number of possible connections per process for your operating system, calculate the initial number of dispatcher processes to create for each network protocol during instance startup using the following formula. Connections per dispatcher is operating system-dependent.

$$\text{Number of Dispatchers} = \text{CEIL}\left(\frac{\text{Maximum number of concurrent sessions}}{\text{Connections per dispatcher}}\right)$$



The MTS_MAX_DISPATCHERS Parameter

MTS_MAX_DISPATCHERS specifies the maximum number of dispatcher processes that can run simultaneously.

After instance startup, you can start more dispatcher processes if needed; however, you can only start dispatchers that use protocols mentioned in the parameter file of the database.

For example, if the parameter file starts dispatchers for TCP and SPX, you cannot later start dispatchers for protocol DECnet without changing the parameter file and restarting the instance.

Parameter Type	**Integer**
Parameter class:	Static
Default value:	If dispatchers are configured, then defaults to whichever is greater: 5 or the number of dispatchers configured
Range of values:	Operating system–dependent

Estimating the Maximum Number of Dispatches

To estimate the maximum number of dispatcher processes an instance requires, use the following formula:

$$\text{MTS_MAX_DISPATCHERS} = \frac{\text{Maximum number of concurrent sessions}}{\text{Connections per dispatcher}}$$

For example, assume that your system typically has 900 users concurrently connected by way of TCP/IP and 600 users connected by way of SPX, and supports 255 connections per process. In this case, the MTS_DISPATCHERS parameter should be set as follows:

```
MTS_DISPATCHERS = "(PROTOCOL=TCP) (DISPATCHERS=4)"
MTS_DISPATCHERS = "(PROTOCOL=SPX) (DISPATCHERS=3)"
```

Adding or Removing Dispatchers

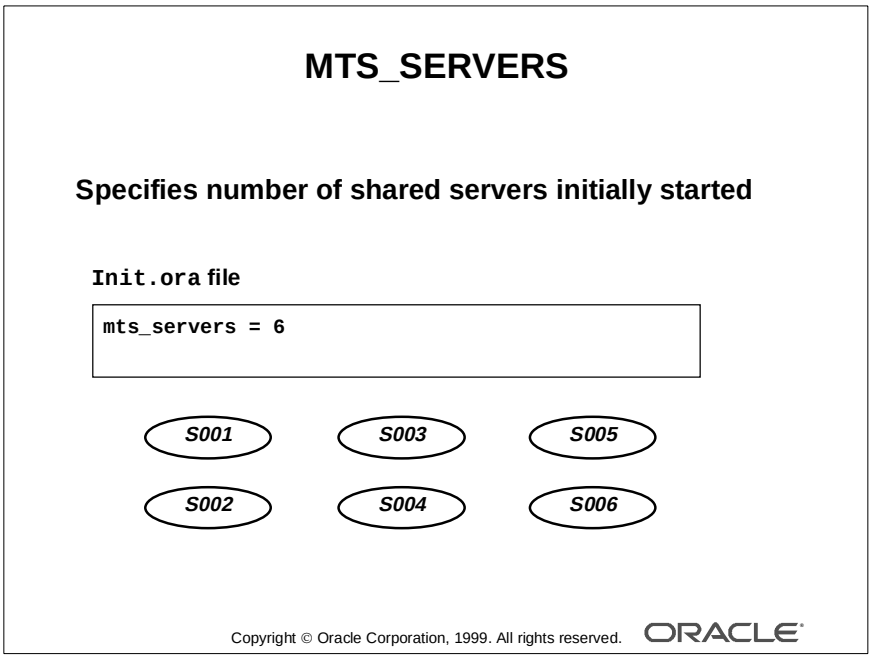
- If the load on the dispatcher processes is consistently high, start additional dispatcher processes to route user requests without waiting. You may start new dispatchers until the number of dispatchers equals MTS_MAX_DISPATCHER.
- The load on the dispatchers can be monitored using the data dictionary views V$CIRCUIT and V$DISPATCHER.

  (Detailed tuning of the MTS is covered in the performance tuning class.)
- In contrast, if the load on dispatchers is consistently low, reduce the number of dispatchers.

The following example adds a dispatcher process where the number of dispatchers was previously two:

```
ALTER SYSTEM SET MTS_DISPATCHERS='(PROTOCOL=TCP)(DISPATCHERS=3)';
```

You can also use the ALTER SYSTEM command to remove dispatchers to the number specified in MTS_DISPATCHERS. If you want to have fewer than that, edit the `init.ora` file, and bounce the database.



The MTS_SERVERS Parameter

MTS_SERVERS specifies the number of server processes that you want to create when an instance is started up.

Parameter Type	**Integer**
Parameter class:	Dynamic (can use ALTER SYSTEM to modify)
Default value:	0
Range of values:	Operating system–dependent

Setting the Initial Number of Shared Server Processes

The appropriate number of initial shared server processes for a database system depends on how many users typically connect to it and how much processing each user requires.

- If each user makes relatively few requests over a period of time, then each associated user process is idle for a large percentage of time. In that case, one shared server process can serve 10 to 20 users.
- If each user requires a significant amount of processing, a higher ratio of server processes to user processes is needed to handle requests.

If you want Oracle to use shared servers, you must set MTS_SERVERS to at least 1. If you omit the parameter or set it to 0, Oracle does not start any shared servers at all.

You can subsequently set MTS_SERVERS to a number greater than 0 while the instance is running.

It is best to estimate fewer initial shared server processes. Additional shared servers start automatically when needed and are deallocated automatically if they remain idle for too long.

- Note that the initial servers always remain allocated, even if they are idle.
- If you set the initial number of servers too high, your system might incur unnecessary overhead.
- Experiment with the number of initial shared server processes, and monitor shared servers until you find the ideal system performance for typical database activity.

Modifying the Minimum Number of Shared Server Processes

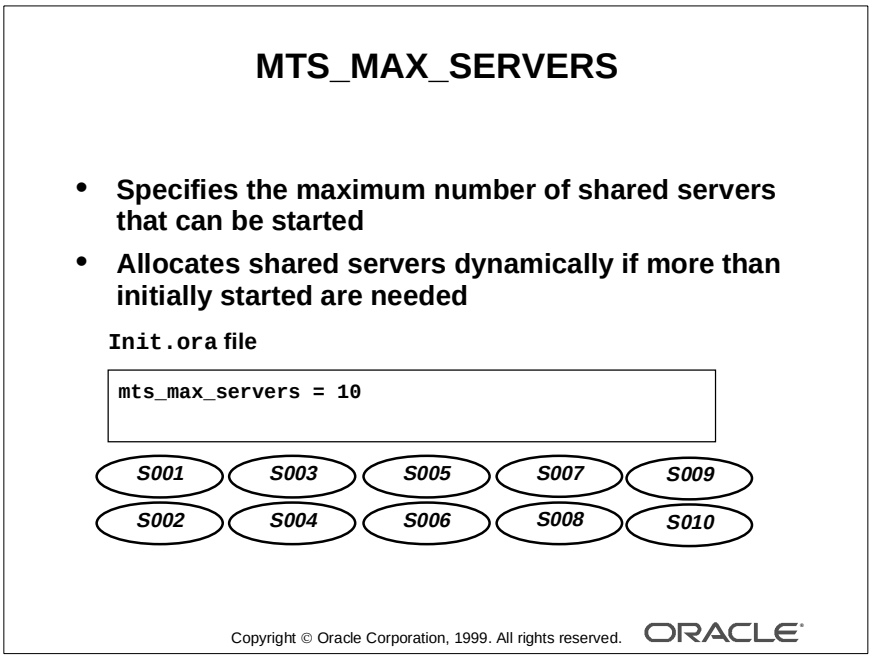
After starting an instance, you can change the minimum number of shared server processes by using the SQL ALTER SYSTEM command.

- Oracle will eventually terminate servers that are idle when there are more shared servers than the minimum limit you specify.
- If you set MTS_SERVERS to 0, Oracle terminates all current servers when they become idle and does not start any new servers until you increase MTS_SERVERS.
- Setting MTS_SERVERS to 0 effectively disables the multithreaded server temporarily.

To control the minimum number of shared server processes, you must have the ALTER SYSTEM privilege.

The following statement sets the number of shared server processes to two:

```
ALTER SYSTEM SET MTS_SERVERS = 2
```



The MTS_MAX_SERVERS Parameter

MTS_MAX_SERVERS specifies the maximum number of shared server processes that can run simultaneously.

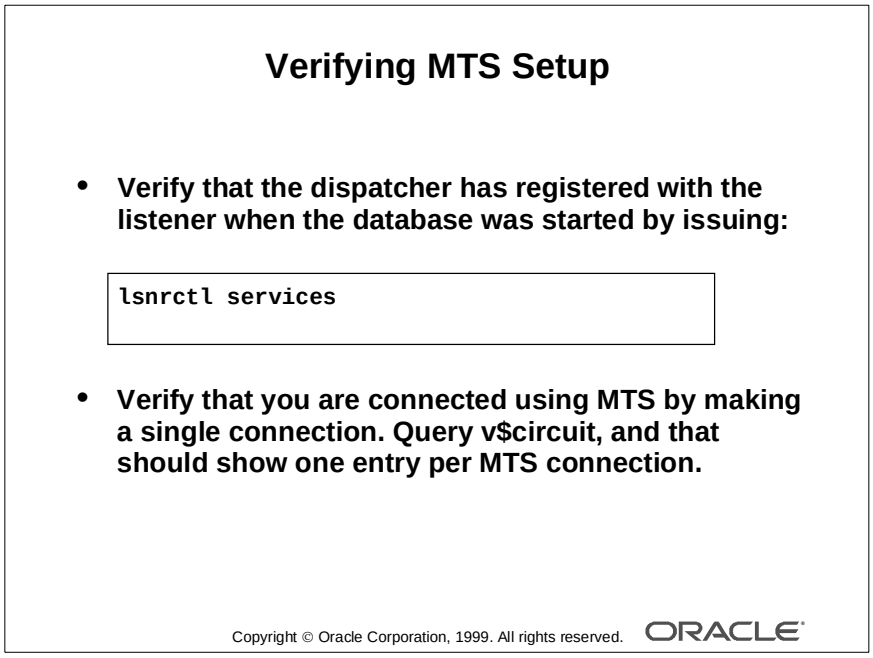
Parameter Type	Integer
Parameter class:	Static
Default value:	Defaults to whichever is greater: 20 or 2 times the value of MAX_SERVERS
Range of values:	Operating system–dependent

Estimating the Maximum Number of Shared Servers

In general, set this parameter for an appropriate number of shared server processes at times of highest activity. Experiment with this limit, and monitor shared servers to determine an ideal setting for this parameter.

To get the maximum numbers of servers started, query the data dictionary view V$MTS.

Data Dictionary



V$CIRCUIT	This view contains information about virtual circuits, which are user connections to the database through dispatchers and servers.
V$SHARED_SERVER	This view contains information on the shared server processes.
V$DISPATCHER	This view provides information on the dispatcher processes.
V$MTS	This view contains information for tuning the multithreaded server.
V$QUEUE	This view contains information on the multithreaded message queues.
V$SESSION	This view lists session information for each current session.

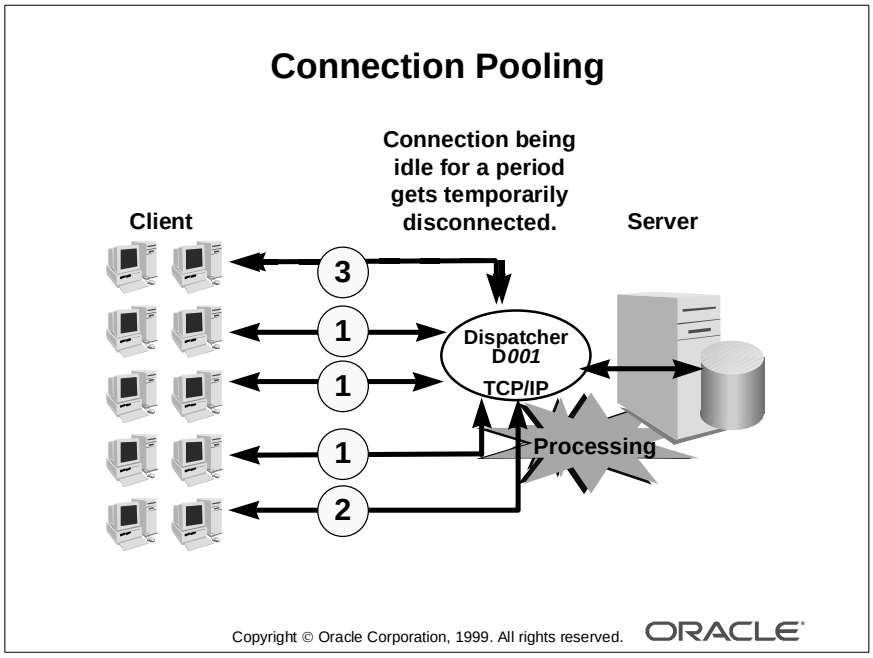
Note: When using MTS, you should first start the listener, then the database, so that the dispatchers can immediately register with the listener.

Data Dictionary Views

- **V$CIRCUIT**
- **V$SHARED_SERVER**
- **V$DISPATCHER**
- **V$MTS**
- **V$QUEUE**
- **V$SESSION**

Copyright © Oracle Corporation, 1999. All rights reserved.

Connection Pooling



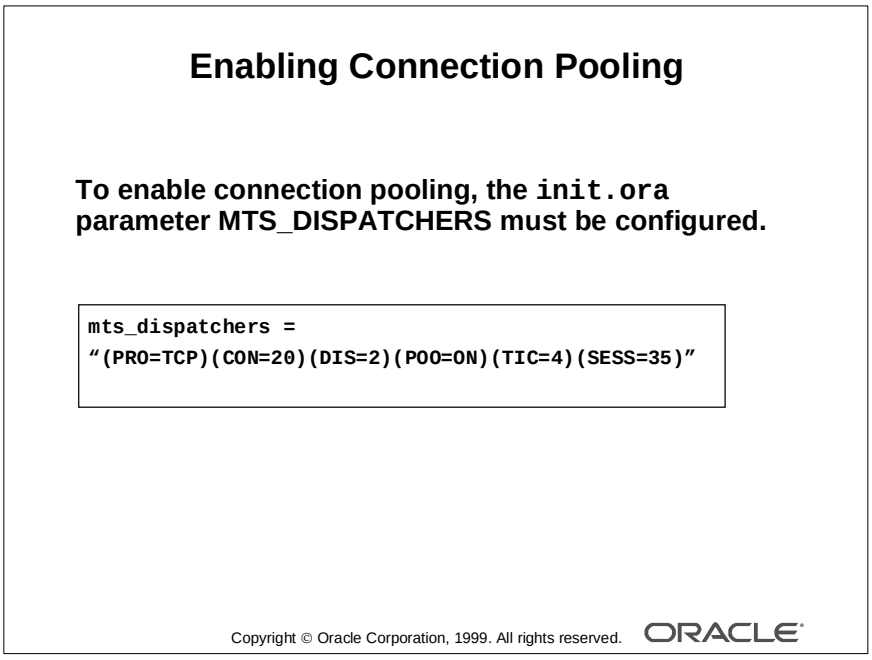
Connection Pooling

Connection pooling is a resource utilization feature you can use to maximize the number of physical network connections to a multithreaded server. This is achieved by a dispatcher sharing or pooling its set of connections among multiple client processes.

By using a time-out mechanism to temporarily release transport connections that have been idle for a specified period of time, connection pooling makes these physical connections available for incoming clients, while still maintaining a logical network session with the previous idle connection. When the idle client has more work to do, the physical connection is reestablished with the dispatcher.

The example in the slide illustrates how connection pooling works.

1. One or more connections are established to a dispatcher in a multithreaded server environment.
2. When the maximum number of connections is reached for available dispatchers for a given protocol, and connection pooling is configured, the next session that tries to establish a connection waits for an existing idle connection to be temporarily disconnected by the dispatcher and then establishes the connection.
3. The connection that was idle becomes temporarily disconnected. If the session needs to process a request, the connection is reestablished by waiting for another session to temporarily disconnect, or if the maximum number of sessions for a dispatcher is less than configured, the processing request is carried out immediately.



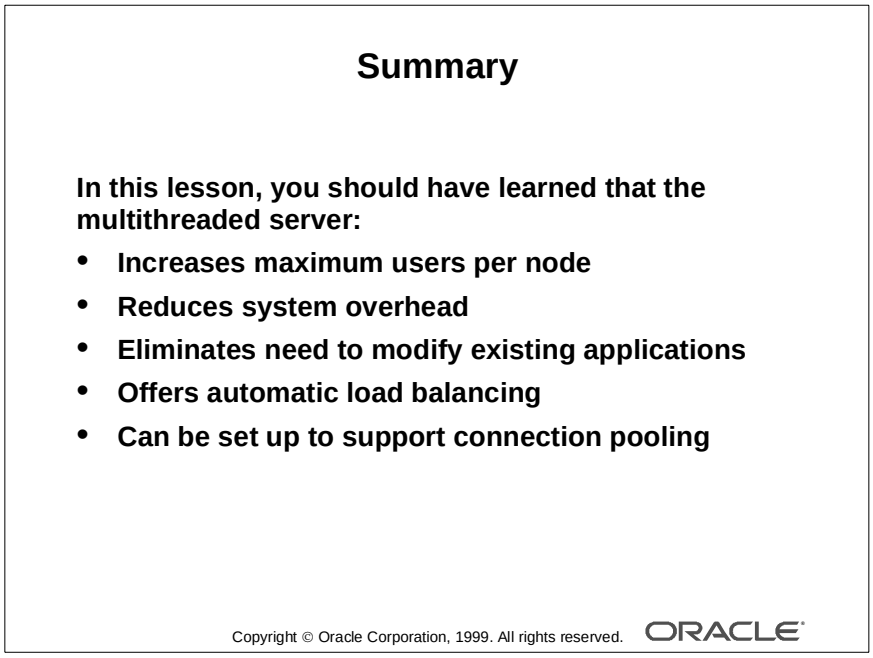
Enabling Connection Pooling

In order to enable the connection pooling, additional parameters must be configured for MTS_DISPATCHERS. The ideal scenario for connection pooling is networks on which many clients run interactive “high idle, high search time” applications, such as messaging and OLAP, and can afford to wait to get connected to a dispatcher. The additional attributes to set for MTS_DISPATCHERS are detailed in the following table.

Attributes	Description
POOL (POO)	This attribute is used to enable connection pooling. If a number is specified, then connection pooling is enabled for both incoming and outgoing network connections, and the number specified is the time-out in ticks for both incoming and outgoing network connections. If ON, YES, TRUE, or BOTH is specified, then “connection pooling” is enabled for both incoming and outgoing network connections, and the default time-out (set by Net8) is used for both incoming and outgoing network connections If IN is specified, then connection pooling is enabled for incoming network connections, and the default time-out (set by Net8) is used for incoming network connections. If OUT is specified, then connection pooling is enabled for outgoing network connections and the default time-out (set by Net8) is used for outgoing network connections. If NO, OFF, or FALSE is specified, then “connection pooling” is disabled for both incoming and outgoing network connections. POOL can also be assigned a name-value string such as: “(IN=10)”, “(OUT=20)”, or “((IN=10)(OUT=20))”, in which case, if an IN numeric value is specified, then connection pooling is enabled for incoming connections, and the number specified is the time-out in ticks for incoming network connections. If an OUT numeric value is specified, then connection pooling is enabled for outgoing network connections and the number specified is the time-out in ticks for outgoing network connections. If the numeric value of a specified time-out is 0, then the default value (set by Net8) is used. The default connection pooling is disabled on both incoming and outgoing network connections.

Attributes	Description
CONNECTION (CON or CONN)	This attribute gives the maximum number of network connections for each dispatcher. **Note:** In release 8.0.3, there is a bug that sets the number of connections to the number you specify plus 1. Default is set by Net8 and is platform-specific. This should be set lower than the number specified for SESSIONS to enable connection pooling. The relationship between CONNECTIONS and SESSIONS is that SESSIONS value specifies the maximum number of user processes you can have working against a dispatcher, while CONNECTIONS specifies the maximum number of connections that can be established against the database through the dispatcher. The difference between the number of SESSIONS and the number of CONNECTIONS specifies how many sessions can be in a waiting state for processing a request. The default is set by Net8 and is platform-specific
TICKS (TIC or TICK)	The attribute specifies the number of 10-second ticks. (Most platforms have a tick set to 10 seconds.) If TICKS is not specified, the default time to wait for a connection in an environment in which all dispatchers for a given protocol are busy is 100 seconds, or 10 ticks.

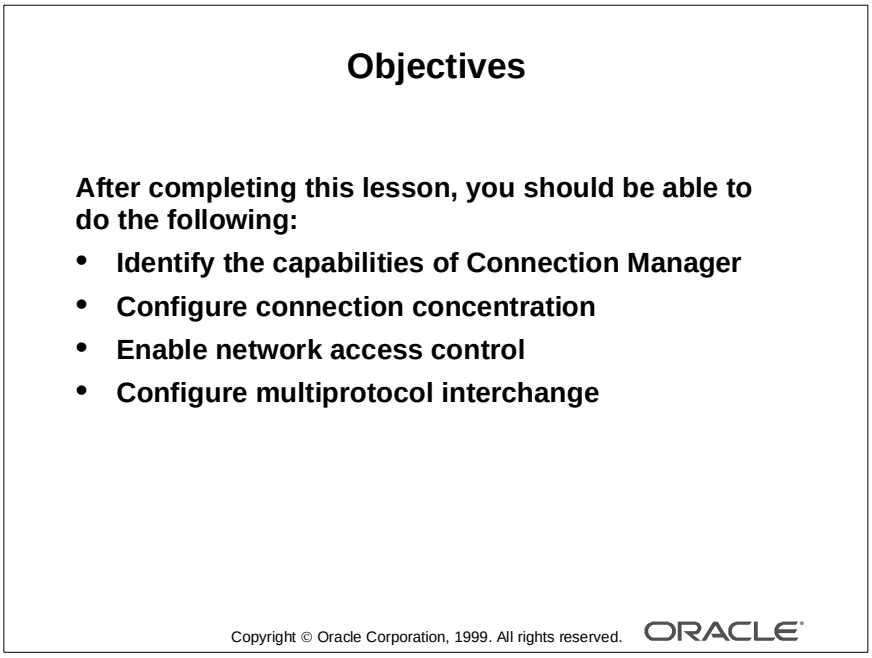
Summary

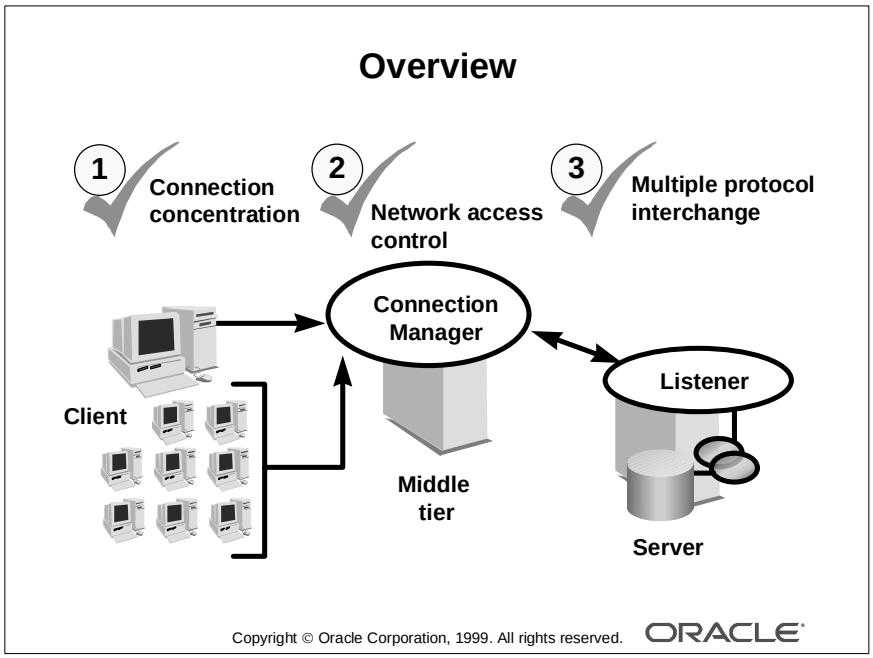

Summary
In this lesson, you should have learned that the multithreaded server:
• Increases maximum users per node
• Reduces system overhead
• Eliminates need to modify existing applications
• Offers automatic load balancing
• Can be set up to support connection pooling
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

8

Connection Manager Usage and Configuration

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Identify the capabilities of Connection Manager
• Configure connection concentration
• Enable network access control
• Configure multiprotocol interchange
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE



Connection Manager

Oracle Connection Manager is a multipurpose, networking solution for Net8 that provides greater resource utilization for increased scalability, multiprotocol connectivity, and secure network access control.

Connection Concentration

Oracle Connection Manager deployed as a connection concentrator requires no reengineering of existing Oracle-based applications. Oracle7, Oracle8, and Oracle8*i* compatible applications can immediately take advantage of the benefits of three-tier computing, which include:

- Efficient use of existing system and network resources for MTS environments
- Scalable systems to accommodate large numbers of concurrent users
- Support for multiple Oracle Connection Managers supported to efficiently partition a network for maximum performance

Access Control

Oracle Connection Manager can be set up to filter connections based on origin, destination, or Oracle system identifier (SID). Additionally, when used in conjunction with the security features of Oracle Advanced Security option, connections can be rejected or accepted on the basis of whether or not security is required.

- Restricts access based on source address, destination address, or Oracle system identifier (SID)
- Optionally implemented with Oracle Advanced Security option to restrict access based on use of security
- Network traffic statistics generated for auditing purposes

Multiprotocol Support

Oracle Connection Manager provides multiprotocol connectivity, so that a client and server with different networking protocols can communicate with one another.

- Transparent protocol conversion for protocols supported by Net8, including:
  - APPC (LU6.2)
  - DECnet
  - Named Pipes
  - SPX/IPX
  - TCP/IP
- Bidirectional protocol conversion
- Scalable from PC to mainframe class machines
- Robust diagnostics with multilevel error logging and step-by-step tracing

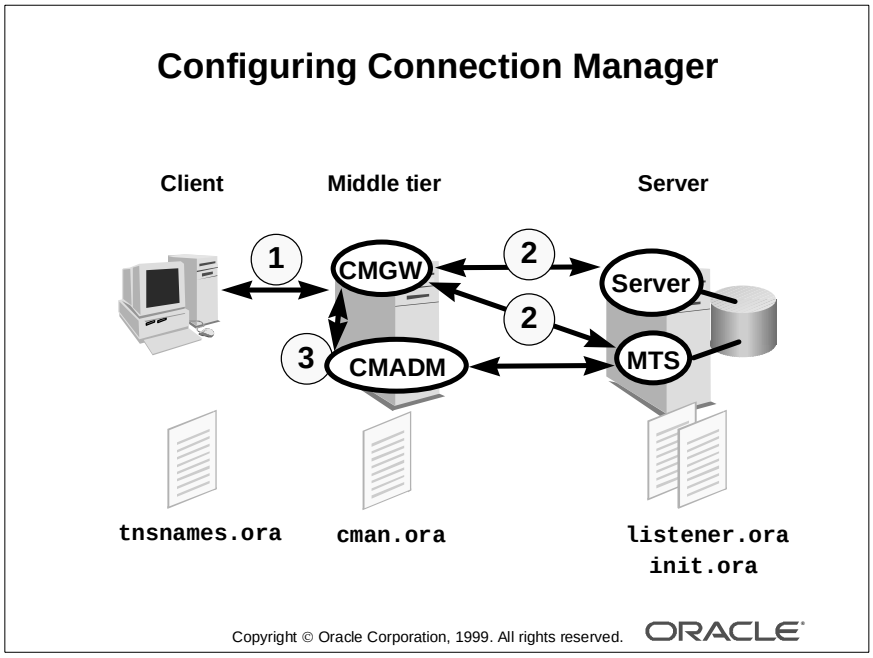
Technical Note

The Connection Manager is a stand-alone product that can be installed on any supported platform. Note, however, that the connection concentration does not work when connection pooling is enabled.

To benefit from the connection concentration feature, it is recommended that Connection Manager be installed on a middle tier in a multitiered environment. This heavily reduces the resource consumption on the end tier, and enables highly scalable configurations.

Connection concentration requires the MTS server, whereas access control and multiprotocol connectivity do not.

Configuring Connection Manager



Connection Concentration

You can configure connection concentration, network access control, or multiprotocol interchange in any combination.

There are, however, certain requirements dependent on the feature you want to configure.

CMGW Process

In general, you need to have started the CMGW process. You do this through a special control utility, CMCTL, which is covered later.

Note: In Oracle8 on NT, these utilities are called CMGW80 and CMCTL80, respectively.

The CMGW process acts as the hub of the Connection Manager. The function of CMGW is to:

- Register itself with CMADMIN for connection concentration
- Listen for incoming SQL*Net 2.*x* or NET 8.*x* connection requests
- Initiate connection requests to NET 8.*x* listeners for clients
- Relay data between the client and server
- Answer requests initiated by CMCTL

The CMGW process listens by default on port 1610 or 1600 (platform dependent).

Slide Example

1. When configuring for network access control, connection concentration, or multiprotocol interchange, you need to have started the CMGW Connection Manager process.

   The CMGW process either rejects your request if you are denied access or connects you to a listener that either spawns and redirects you to a server process or to a dispatcher in a multithreaded server environment.

2. When using Connection Manager with an Oracle Names server, an additional process, CMADMIN, needs to be started.

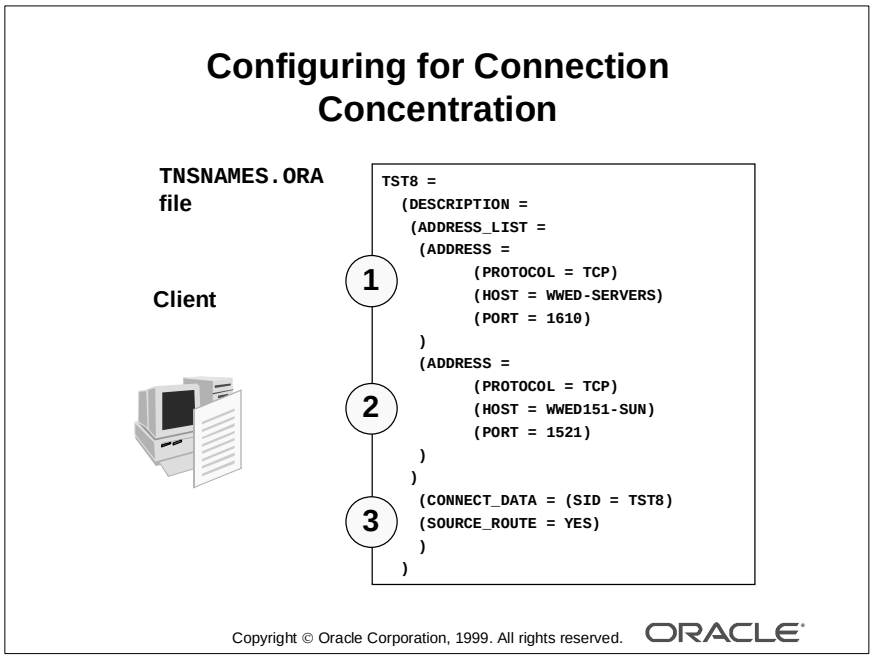
   Note: In Oracle8 on NT, this utility is called CMADM80.

3. The CMADMIN is a multithreaded process that is responsible for all administrative issues of the Connection Manager. Its primary function is to maintain address information in the Oracle Names server for the SQL*Net 2.*x* and NET 8.*x* clients.

   Other responsibilities include:

   – Processing the CMGW registration
   – Locating the local Oracle Names server
   – Identifying all listeners serving at least one database instance
   – Registering source route address information about the CMGW and listeners
   – Monitoring changes in the network and updating the Names server
   – Answering requests initiated by CMCTL

The final requirement for connection concentration is that the multithreaded server on the destination server needs to be configured with a special parameter for the dispatcher setting, which is covered later.



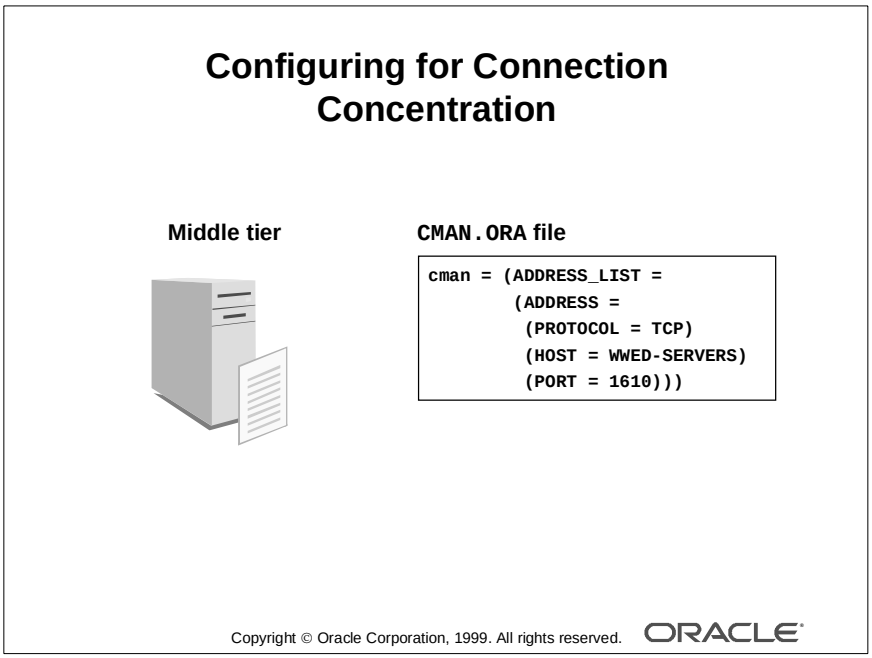
Connection Concentration Configuration

In order to configure for connection concentration, the listener address for the Connection Manager CMGW process needs to be listed as well as the destination listener.

The following are the components of the service name TST8 in the example:

1. ADDRESS specified for the Connection Manager process. Both the node on which Connection Manager is running and the port number need to be specified.
2. ADDRESS specified for the listener on the destination node. Both port and node names need to be specified.
3. The database SID or the SERVICE_NAME to which the connection needs to be established, and in order to enable connecting through Connection Manager, the SOURCE_ROUTE needs to equal YES.

The Connection Manager basically acts as a router, pointing the connection request to the next hop.



Concentration Configuration File

Configuration of CMAN requires one file, `cman.ora`.

In the `cman.ora` file, three sections can be configured:

1. CMAN

   Contains the listening address for the Connection Manager. This entry must always, unless you select the default configuration, be specified

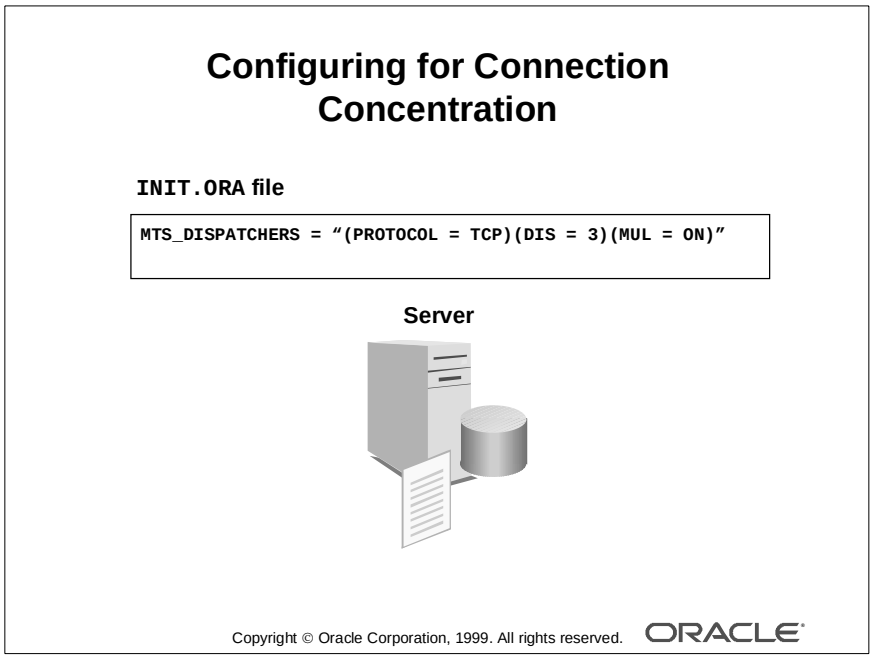
2. CMAN_PROFILE

   Contains CMAN configuration parameters

3. CMAN_RULES

   Contains the rules for filtering incoming connection requests

In the example, only the CMAN section has been configured. The Connection Manager is configured to run on the server named WWED-SERVERS, and the CMGW process listens on port 1610.



Database Parameter File Configuration

To use connection concentration, you are required to configure the multithreaded server (MTS) on the destination database.

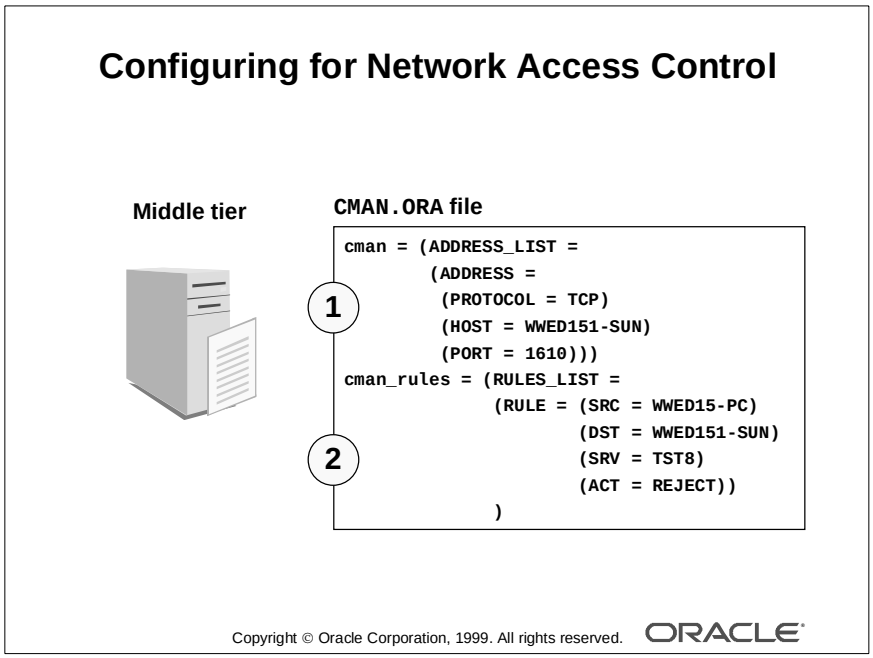
In order to enable the connection pooling, you need to specify a new parameter for (MUL = ON) MTS_DISPATCHERS.

Database Parameter File Configuration (continued)

Parameter Type	Description
MULTIPLE (MUL or MULT)	Used to enable the Net8 connection concentration feature. If 1, ON, YES, TRUE, or BOTH is specified, then connection concentration is enabled for both incoming and outgoing network connections. If IN is specified, then Network Session Multiplex is enabled for incoming network connections. If OUT is specified, then Network Session Multiplexing is enabled for outgoing network connections. If 0, NO, OFF, or FALSE is specified, then Network Session Multiplexing is disabled for both incoming and outgoing network connections. The default for MULTIPLEX is disabled on both incoming and outgoing network connections.

Technical Note

The connection concentration feature does not work with connection pooling, so be sure that the connection pooling is not configured in MTS_DISPATCHERS.



Network Access Control Configuration

The CMAN section remains the same.

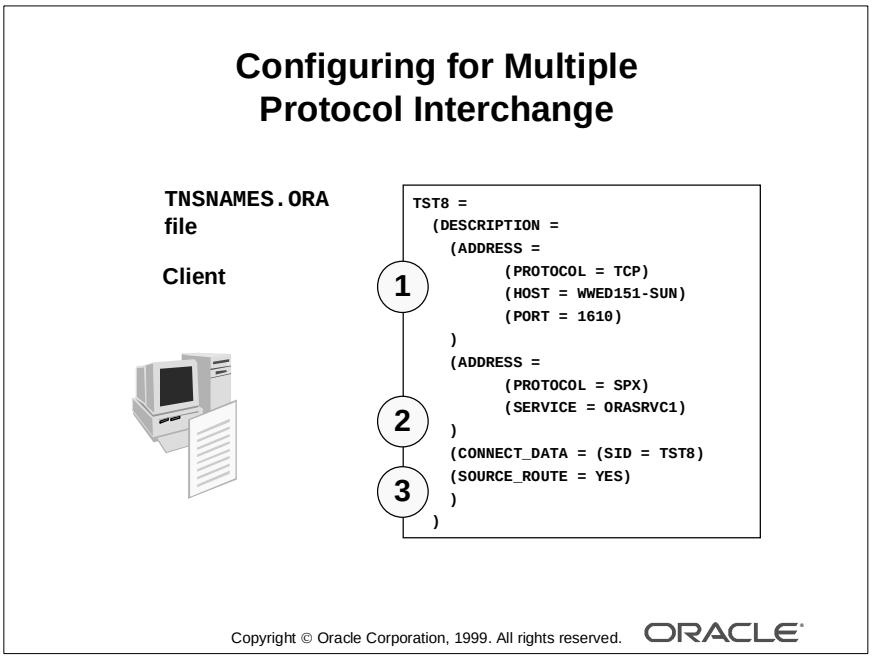
For network access control, the CMAN_RULES section of the `cman.ora` file needs to be configured.

Parameter Type	Description
SRC	Source host name or IP address of session request (client)
DST	Destination host name or IP address (server)
SRV	SID name of the targeted database
ACT	Accept (ACC) or reject (REJ) the incoming requests based on the preceding three parameters

Technical Note

The wildcard is the *x*. In the case of the IP address (d.d.d.d), the individual ds may be replaced with wildcard character *x*.

Multiple rules can be defined within the RULE_LIST. The rules in the first matched rule are applied to the request. When CMAN_RULES exist, the Connection Manager adheres to the principle “that which is not expressly permitted is prohibited.” If the CMAN_RULES are not defined, then everything is permitted.



Multiple Protocol Interchange Configuration

In this example, the client uses:

1. TCP/IP to connect to the CMAN
2. Then the CMAN uses SPX/IX to connect to the server.
3. The SID and SOURCE_ROUTE need to be specified as explained earlier.

This is the same functionality that was provided with the Oracle MultiProtocol Interchange (MPI) of Oracle7.

Note that both the TCP and the SPX protocols need to be installed on the node on which Connection Manager is running.

Optional Settings for Connection Manager

Middle tier



CMAN.ORA

```
cman_profile =
(PARAMETER_LIST =
   (MAXIMUM_RELAYS = 512)
   (LOG_LEVEL = 0)
   (TRACING = YES)
   (RELAY_STATISTICS = NO)
   (SHOW_TNS_INFO = YES)
   (USE_ASYNC_CALL = YES)
   (AUTHENTICATION_LEVEL = 1)
   (MAXIMUM_CONNECT_DATA = 2048)
   (ANSWER_TIMOUT = 5)
)
```

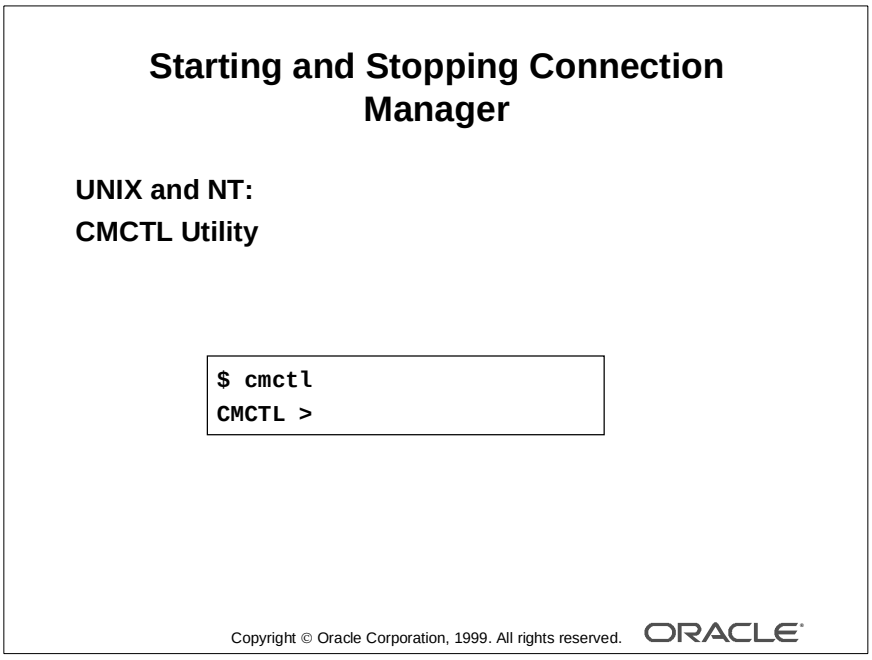
Copyright © Oracle Corporation, 1999. All rights reserved.

CMAN Parameters

Parameter Type	Description
MAXIMUM_RELAYS	Determines the maximum number of concurrent connections allowed. The default is 8. Maximum value allowed is 1,024.
LOG_LEVEL	Determines the level of logging performed by the CMAN. Default is 0, which means no logging is performed Value can range from 0 to 4.
TRACING	Can be specified as YES or NO. YES enables CMAN tracing to a file. Default is NO. **Note:** Oracle Trace must be used to read the trace file.
RELAY_STATISTICS	Can be specified as YES or NO. YES instructs the CMAN to maintain statistics pertaining to relay I/O activities such as: • Number of IN bytes • Number of OUT bytes • Number of IN packets • Number of OUT packets Default is NO.
SHOW_TNS_INFO	Can be specified as YES or NO. YES instructs the CMAN to include TNS events in the log file. Default is NO.

CMAN Parameters (continued)

Parameter Type	Description
USE_ASYNC_CALL	Can be specified as YES or NO. YES instructs the CMAN to use all asynchronous functions while in the answering, accepting, or calling phase of establishing a connection. Default is NO. **Note:** CMAN supports out-of-band breaks.
AUTHENTICATION_LEVEL	Can be specified as 0 or 1. 1 instructs the CMAN to reject connection requests that do not use secure services that are part of the Oracle Advanced Security option. 0 is the default, which means secure services are not required.



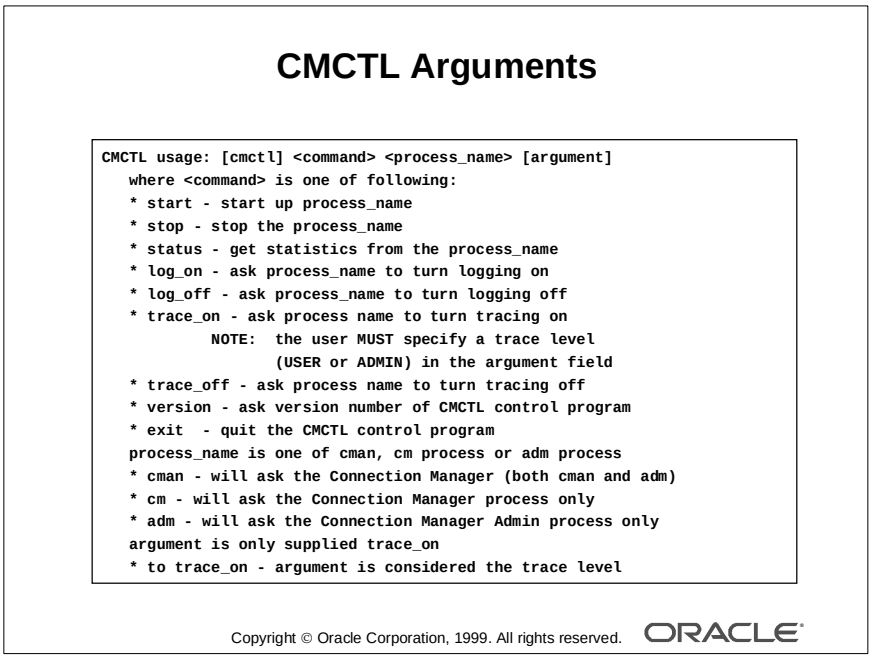
Connection Manager Control Utility (CMCTL)

This utility provides administrative access to CMADMIN and CMGW.

The most used commands are start, stop, status, and version.

Parameter Type	Description
CMCTL START [CMAN | CM | ADM]	This command starts up the Connection Manager, or one of its components. No argument or CMAN means both the gateway and administration process start. CM starts only the gateway process (CMGW). ADM starts only the administration process (CMADMIN).

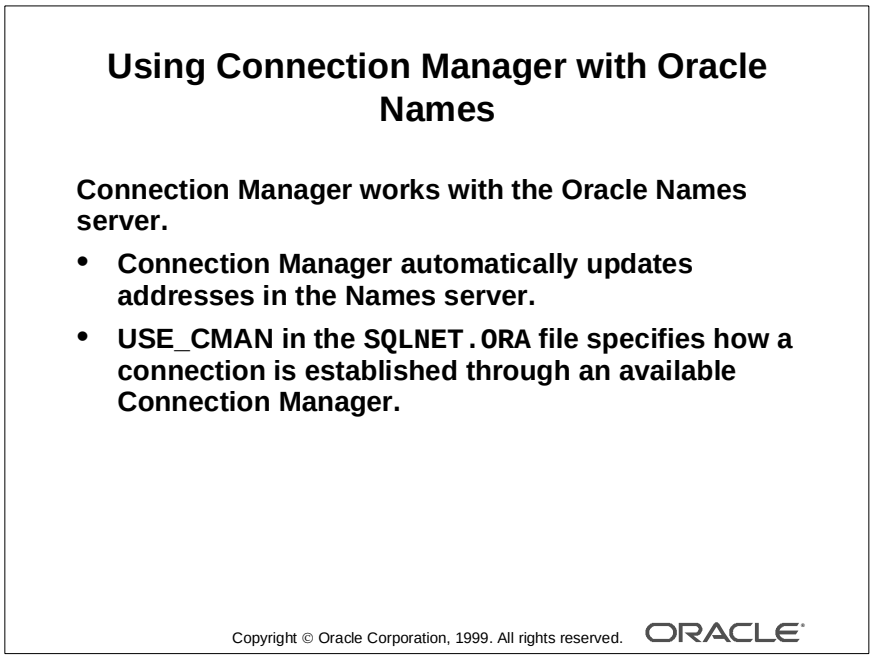
Parameter Type	Description
CMCTL STOP [CM]	This command stops the Connection Manager processes. No argument or CM means both the gateway and administration processes stop. **Note:** The administration process stops automatically when the gateway process terminates. CMAN cannot be stopped if there are connections currently going through it.
CMCTL STATUS [CMAN | CM | ADM]	This command provides the status of the Connection Manager components. No argument, CMAN, or CM provides the status of the gateway process. ADM provides the status of the administration process.
CMCTL VERSION	This command provides the version number of the control utility. The version of CMGW and CMADMIN are provided with the status command.



CMCTL Optional Arguments

In addition to starting and stopping Connection Manager, you can specify additional arguments from the CMCTL utility.

These could also have been configured in the `cman.ora` file.



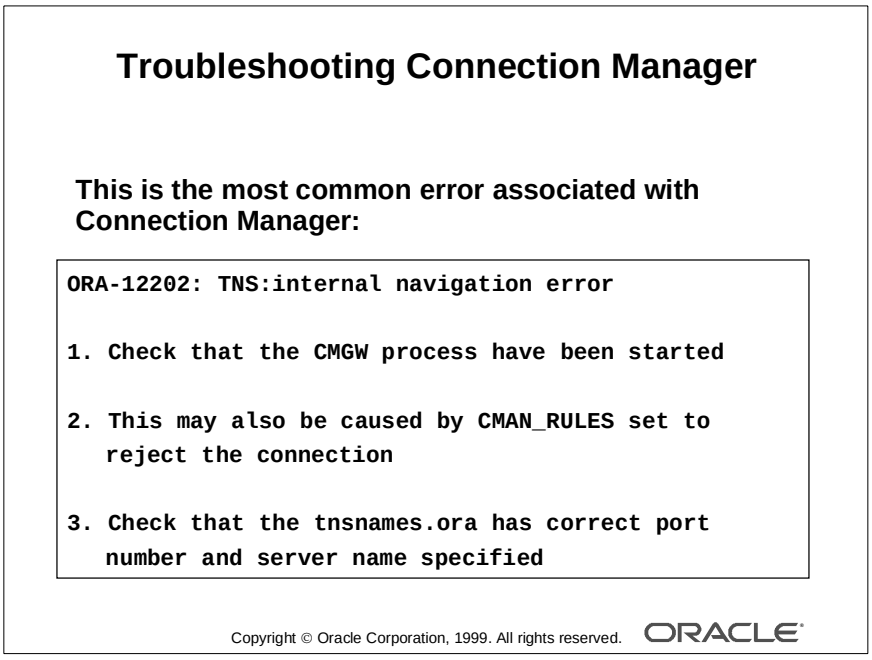
Connection Manager and Names Server

If using Oracle Names server, the Connection Manager automatically updates the addresses in the Names server, inserting the address for the CMAN into the existing address.

Note: If more than one Oracle Connection Manager is used in the connection path, you cannot use Oracle Names to connect clients through it.

In the client `sqlnet.ora` file, the following parameter can be set:

Parameter Type	Description
USE_CMAN	This can be set to TRUE or FALSE. TRUE forces the client to connect to the destination through the CMAN using a randomly picked indirect path (address list with at least one CMAN address). TRUE and no indirect paths in the description, then a randomly picked path is used. FALSE or not defined, then a path is picked at random.

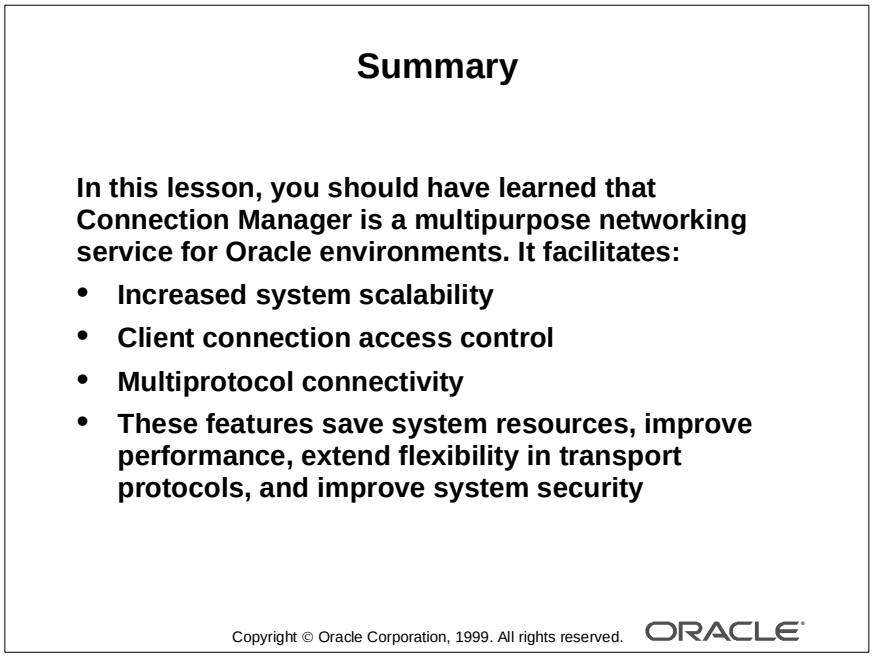


Common Error Messages

The most common error seen with Connection Manager is ORA-12202.

Other errors you may encounter are the ones mentioned in the basic configuration, for example, No Listener, Unable to resolve service name, and similar errors.

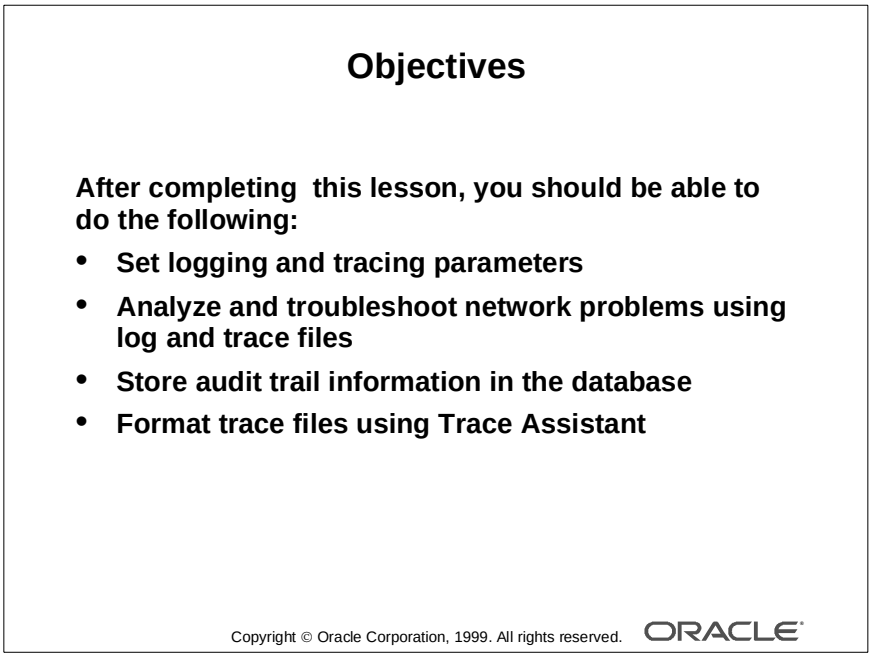
Summary


Summary
In this lesson, you should have learned that Connection Manager is a multipurpose networking service for Oracle environments. It facilitates:
• Increased system scalability
• Client connection access control
• Multiprotocol connectivity
• These features save system resources, improve performance, extend flexibility in transport protocols, and improve system security
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

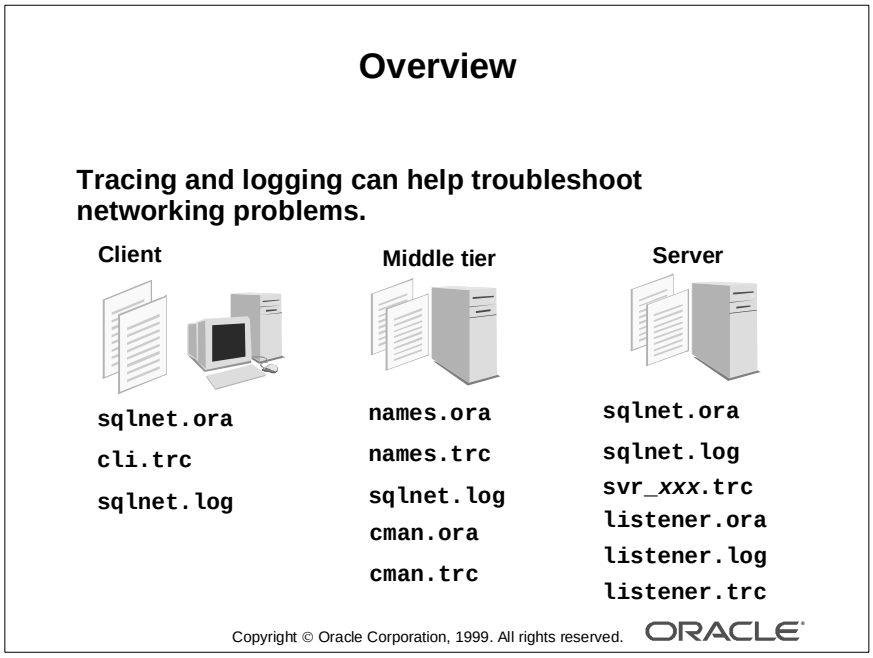
9

Troubleshooting the Network Environment

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Set logging and tracing parameters
• Analyze and troubleshoot network problems using log and trace files
• Store audit trail information in the database
• Format trace files using Trace Assistant
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Overview

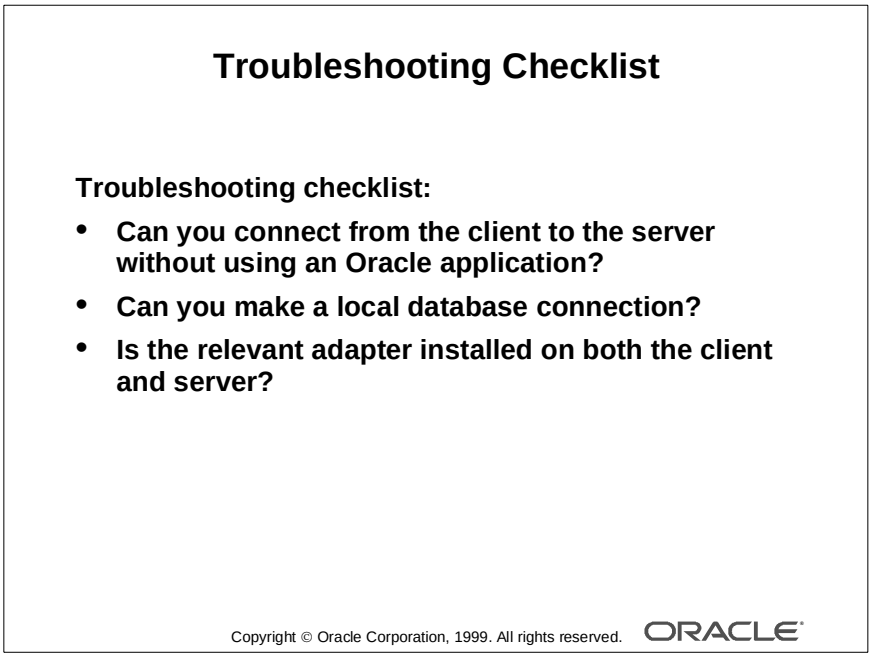


Logging and Tracing Files

Net8 provides methods for understanding and resolving network problems through the use of log and trace files. These files keep track of the interaction between network components as errors occur. Evaluating this information helps you to troubleshoot problems associated with the different networking components.

The output from tracing and logging are stored in ASCII files. The location and how much tracing information can be written to these files are configurable.

Troubleshooting Checklist



Initial Troubleshooting Checklist

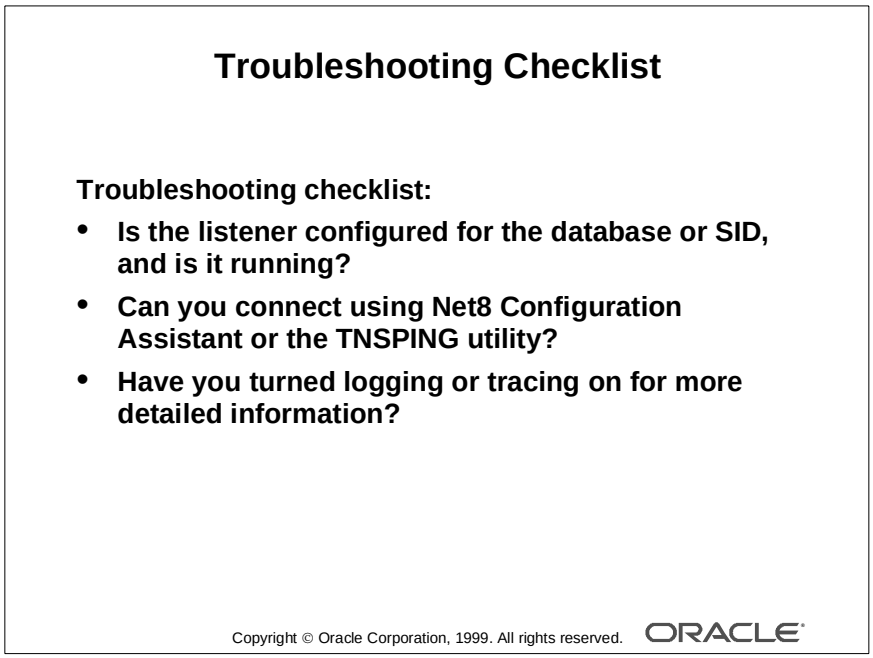
If you experience problems with your network and the error codes returned do not give you the necessary information to correct your problem, then make sure that the basic functions are working.

Verify the following:

- Check that you can establish a basic connection from your client to the server.

  This ensures that the underlying network is operational.
- Check that you can make a local database connection.

  This ensures that there are no database-related problems.
- Check that you have the relevant adapter installed on the client and the server.
  - On UNIX, you find an executable called *adapters*, which you can use to test if any executable has the adapter linked in.
    Example *adapter Oracle*, shows you if the Oracle kernel has your adapter linked in. If it is not linked in or not. If not, you have to relink your executable.
  - On NT, you can verify that you have the right adapter by using the Oracle Installer and checking the installed products.



Initial Troubleshooting Checklist (continued)

- Check that the listener is configured for the correct database SID and that the listener has been started.

  This ensures the server-side connectivity.
- Check Oracle connectivity using Net8 Assistant or the TNSPING utility.

  Use the Net8 Assistant Test Connection feature to check that you can connect to your destination. This should verify if your configuration files are incorrect.

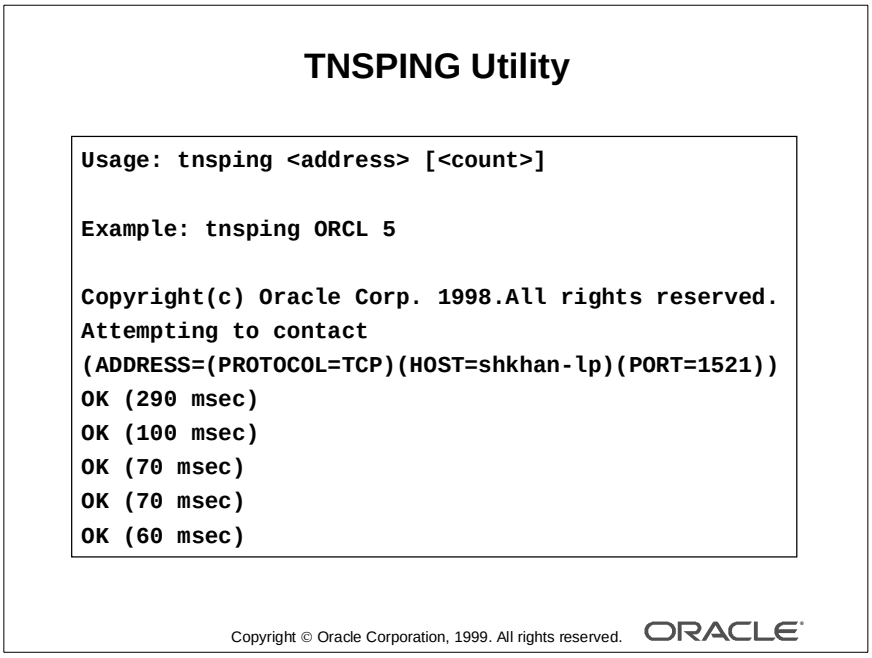
  Use TNSPING, which is a utility that determines whether or not a service (for example, an Oracle database, an Oracle Names server, or any other Oracle service) on a Net8 network can be successfully reached.

  If you still have a problem with your network, then use logging or tracing to get additional information.

You can enable tracing for all major components of Net8. Later in this lesson, you look at how logging and tracing can be enabled for the client side.

This is in most cases the best place to start.

TNSPING Utility



TNSPING Utility

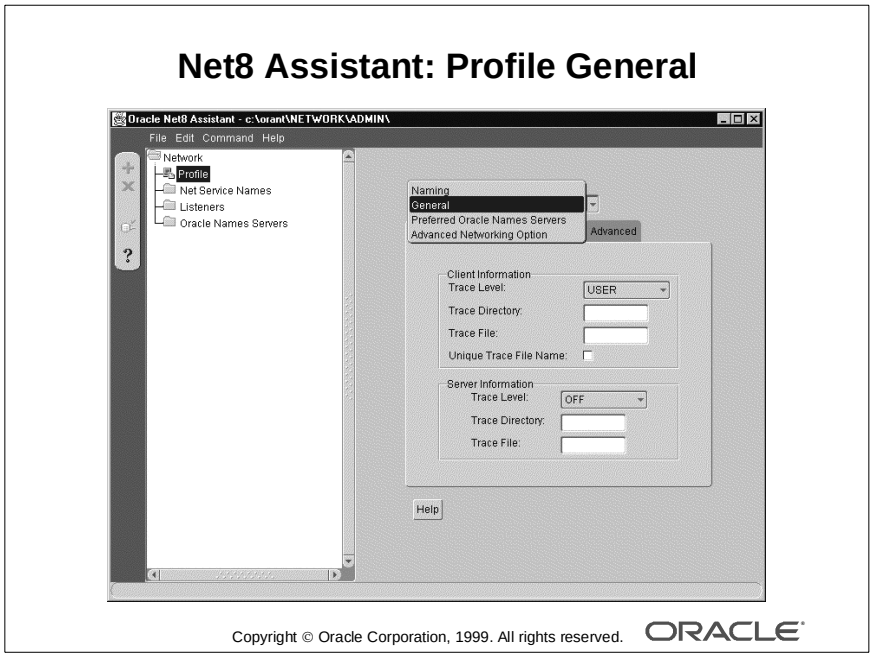
TNSPING is a utility that determines whether or not a service such as an Oracle database or a Names server on a Net8 network can be successfully reached.

A successful connect from a client to a server (or from a server to another server) using TNSPING displays an estimated round-trip time (in milliseconds) to reach the Net8 service.

If the connection fails, TNSPING displays a message describing the error that occurred. You can see the network error that is occurring without the overhead of using a database connection.

TNSPING differs from the way other Oracle application connect. It does not require a username and password to check the underlying connectivity.

Net8 Assistant: Logging and Tracing



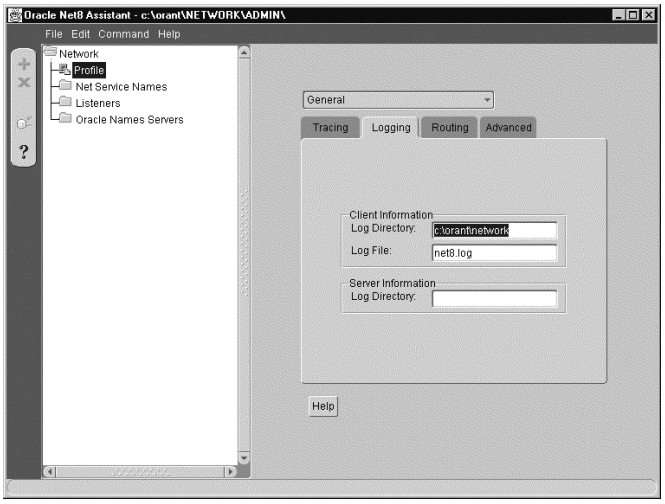
Enabling Logging and Tracing

To turn tracing on for the client side, you have two options. You can use the Oracle Net8 Configuration Assistant, which saves your preferences to a profile. A profile is stored and implemented through a configuration file called `sqlnet.ora`.

The other option is to edit the profile yourself. This approach is not recommended, because it increases the possibility of configuration errors.

To configure tracing and logging, start Net8 Assistant and select Profile—>General.

Net8 Assistant: Profile Logging



Copyright © Oracle Corporation, 1999. All rights reserved.

Net8 Logging

Logging refers to the process by which network components note and append error-specific information to a log file.

Each Net8 component produces its own log file to describe the state of the software at various communication layers as an error occurs. To ensure that all errors are recorded, logging cannot be disabled on clients or Names servers.

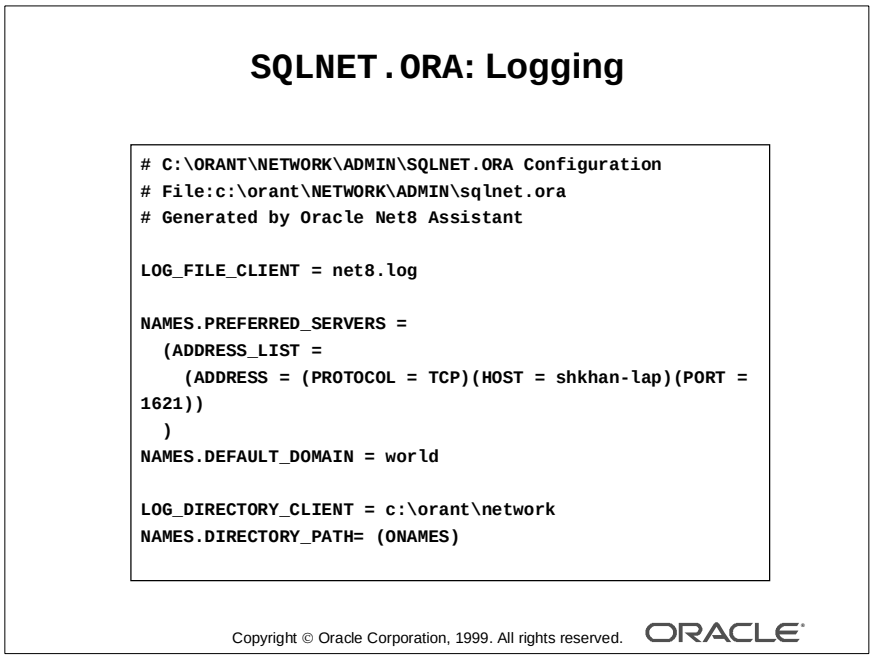
Furthermore, only an administrator may replace or erase log files. The log file for the listener also includes audit trail information about every client connection request, as well as most listener control commands.

Configuring Client Logging

To configure logging on the client using the Net8 Configuration Assistant, proceed as follows:

1. From the Oracle Net8 Configuration Assistant, click the Profile icon in the tree directory.
2. Select General from the pull-down menu.
3. Click the Logging tab.
4. Enter any valid directory name in the Log Directory field in the Client Information region to specify the directory to which log files are to be written on either the client or the server.
5. Enter any valid filename in the Log File field in the Client Information region to specify the name of the log file on the client.
6. Save your configuration.

Note: The Server Information field in the Logging tab panel is appropriate only for logging networking operations on a server.



Client Profile Logging Parameters

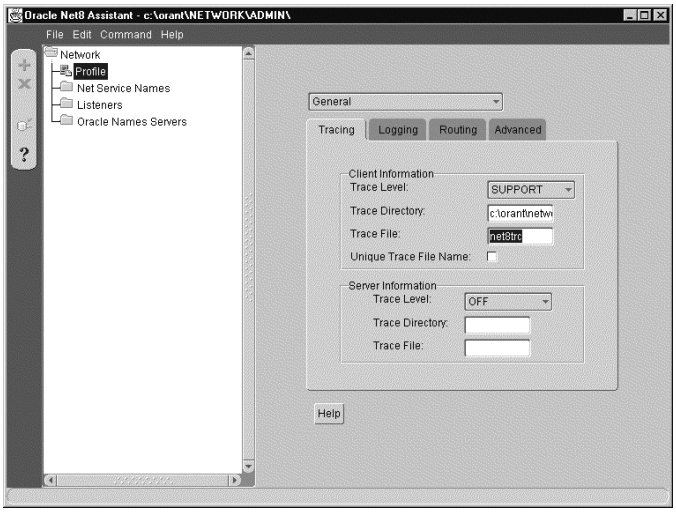
The `sqlnet.ora` file generated holds the information you put in the profile.

The parameters you can specify for logging are:

LOG_DIRECTORY_CLIENT	Controls the directory in which the log file is written.
Default Value:	The current directory from which the executable is started.
Example:	log_directory_client=/oracle/network/log

LOG_FILE_CLIENT	Controls the log output filename for an Oracle client.
Default Value:	SQLNET.LOG
Example:	log_file_client=client

Net8 Assistant: Profile Tracing



Copyright © Oracle Corporation, 1999. All rights reserved.

Net8 Tracing

The tracing facility produces a detailed sequence of statements that describe network events as they are executed.

By tracing an operation, you can obtain more information on the internal operations of the components of Net8 than is provided in a log file. This information is output to files that can be evaluated to identify the events that led to an error.

Tracing Precautions

The tracing facility uses a large amount of disk space and can have a significant impact on system performance. Therefore, you should use tracing only when necessary.

Use the Net8 Configuration Assistant to enable tracing on a client. Tracing is a feature through which you can obtain information about the internal operations of Net8 that can be useful in troubleshooting problems in your networking environment.

This feature is disabled by default.

Configuring Client Tracing

To configure tracing on the client using the Net8 Configuration Assistant, proceed as follows:

1. From the Net8 Assistant, click the Profile icon in the directory tree.
2. Select General from the pull-down menu.
3. Click the Tracing tab.

Configuring Client Tracing (continued)

4. Select a tracing level from the pull-down menu located in the Client Information region. You can choose from one of the following values to specify the level of tracing on a client or server:

   OFF: Tracing disabled

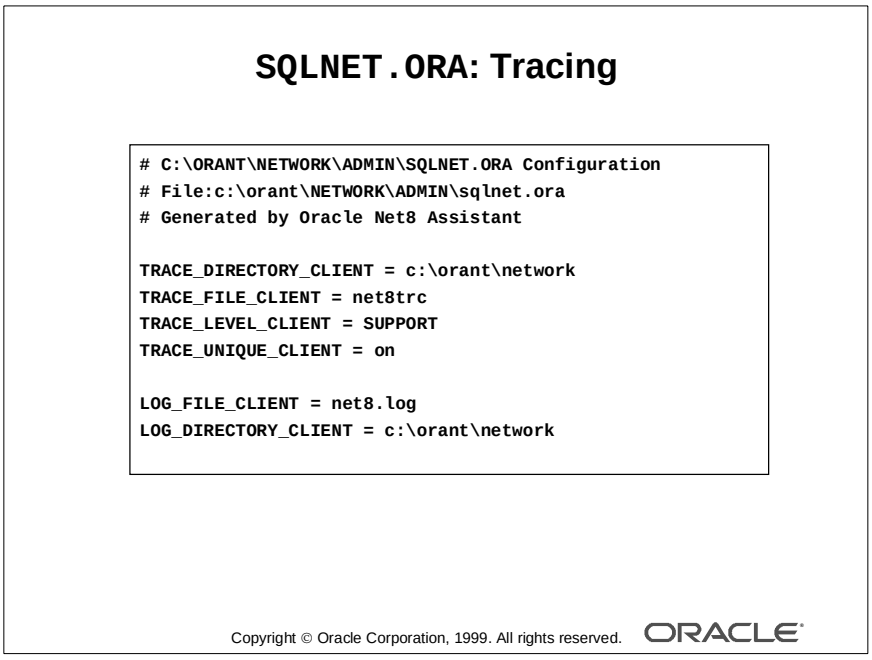
   USER: Tracing information applicable for users

   ADMIN: Tracing information applicable for database administrators

   SUPPORT: Tracing information applicable for customer support staff

5. Enter any valid directory name in the Trace Directory field in the Client Information region to specify the directory to which trace files are to be written on a client or server.
6. Enter any valid filename in the Trace File field in the Client Information region to specify the name of the trace file on the client or server.
7. Select the Unique File Trace Name check box if you want a unique identifier appended to each new trace file created. Otherwise, new trace data is appended to the same file with each new session.
8. Save your configuration.

Note: The Server Information field in the Tracing tab panel is appropriate only for tracing networking operations on a server.



Client Profile Tracing Parameters

The `sqlnet.ora` file generated holds the information you put in the profile.

The parameters you can specify for tracing are:

TRACE_DIRECTORY_CLIENT	Controls the directory in which the trace file is written.
Default Value:	$ORACLE_HOME/network/trace
Example:	trace_directory_client=/oracle/traces

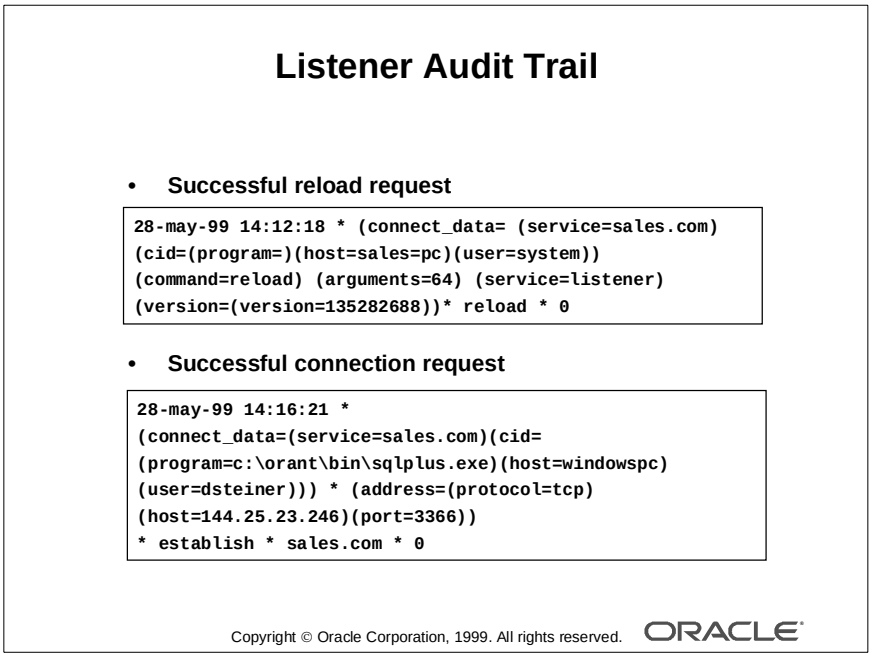
TRACE_FILE_CLIENT	Controls the name of the client trace file.
Default Value:	SQLNET.TRC
Example:	trace_file_client=cli

Client Profile Tracing Parameters (continued)

TRACE_LEVEL_CLIENT	Turns tracing on or off to a certain specified level.
Default Value:	OFF: No trace output USER: User trace information ADMIN: Administration trace information SUPPORT: Worldwide Customer Support tracing information
Example:	trace_level_client=user

TRACE_UNIQUE_CLIENT	Used to make each client trace file have a unique name to prevent each trace file from being overwritten with the next occurrence of the client. The PID is attached to the end of the filename.
Default Value:	OFF
Example:	trace_unique_client=on

Listener Audit Trail



Audit Trail

The listener log file contains audit trail information you can use to gather and analyze network usage and statistical information whenever one of the following occurs:

- A client connection request
- A start, stop, status, reload, or service command issued by the listener control utility

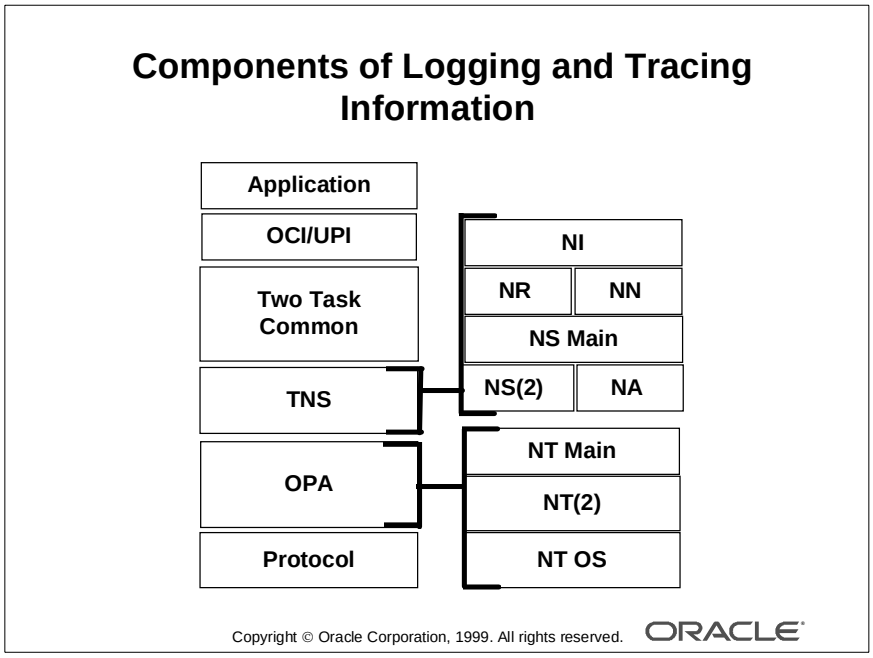
In a client connection request, the user ID is recorded as well as the platform, protocol, and software used to make the connection.

Audit trail information can be used to view trends and user activity by first storing it in a table and then collating it into a report format. To import the data into a table, use an import utility such as SQL*Loader.

Note: The audit trail feature is switched on by default, and there is no way to turn it off.

For more information on SQL*Loader, refer to the *Oracle8i Utilities Guide.*

Logging and Tracing Components



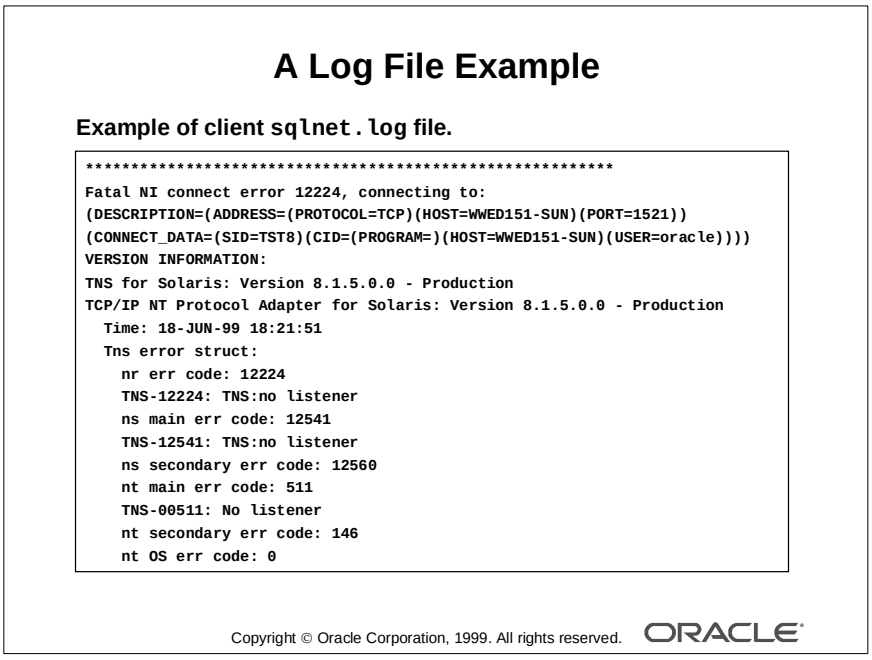
Net8 Tracing Layer Aliases

Log files provide information contained in an error stack. An error stack refers to the information that is produced by each layer in an Oracle communications stack as the result of a network error.

The example shows the networking layer codes as they might appear in an error stack.

Trace files contain more detailed information of the routines used within the different layers. The routines are prefixed with the layer aliases mentioned.

- NI Net8 Interface Layer
- NR Network Routing
- NN Network Naming (Oracle Names)
- NS Network Session (main and secondary layers)
- NA Native Services includes Network Authentication (NAU) and Network Encryption (NAE)
- NT Network Transport (main, secondary, and operating system layers)



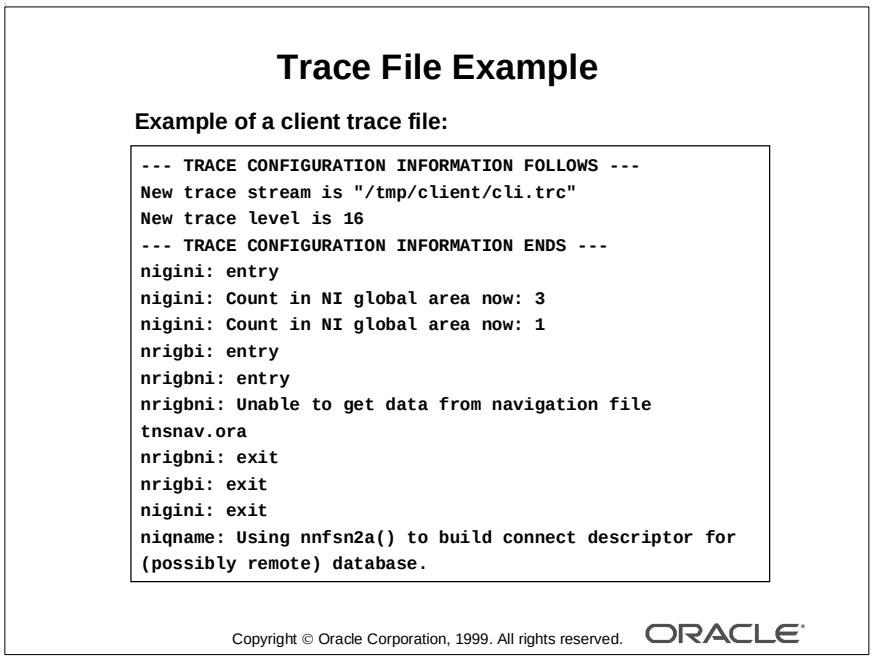
Log File Example

In the example in the slide, a connection failed because the listener was not started.

The error generated, plus additional information, is written to the `sqlnet.log` file.

To use a log file to diagnose a network error:

1. Review the log file for the most recent error number you received from the application. Note that this is almost always the last entry in the log file.
2. Starting from the bottom of the file, locate the first non-zero entry in the error report. This is usually the actual cause.
3. If that error does not provide the desired information, review the next error in the stack until you locate the correct error information.
4. If the cause of the error is still not clear, turn on tracing and reexecute the statement that produced the error message.



Trace File Example

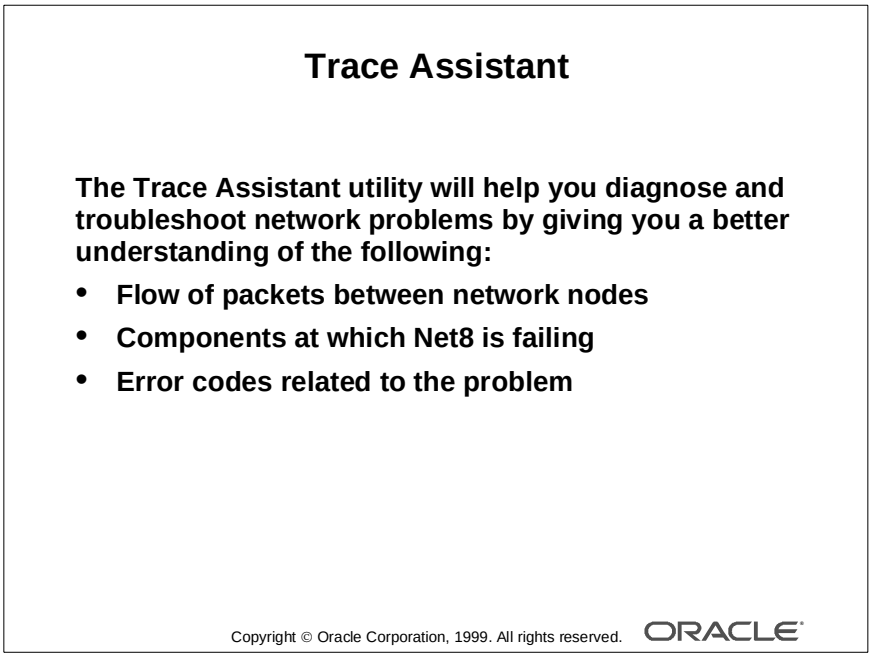
You should normally use tracing when a network error cannot be resolved through the log files.

When turning on tracing, you should enable it just for the duration of the process of reproducing an error situation. Tracing is generated for every session and is an overhead, so do remember to disable tracing whenever it is not needed.

The following characteristics apply to the information generated by tracing from the networking layers:

- Each layer may or may not be involved in network communication.This depends on the function carried out. For instance a connect through a listener not involving a Names server and a connect involving a Names server will use different routines or paths through the layers.
- Each layer internally has multiple routines, each postfixed with the layer name. For example, *nigini*, where the *ni* is the Network Interface, and *gini* is part of the function name.
- Each layer can call routines from other layers one or more times during network communication.

Trace Assistant



Trace Assistant

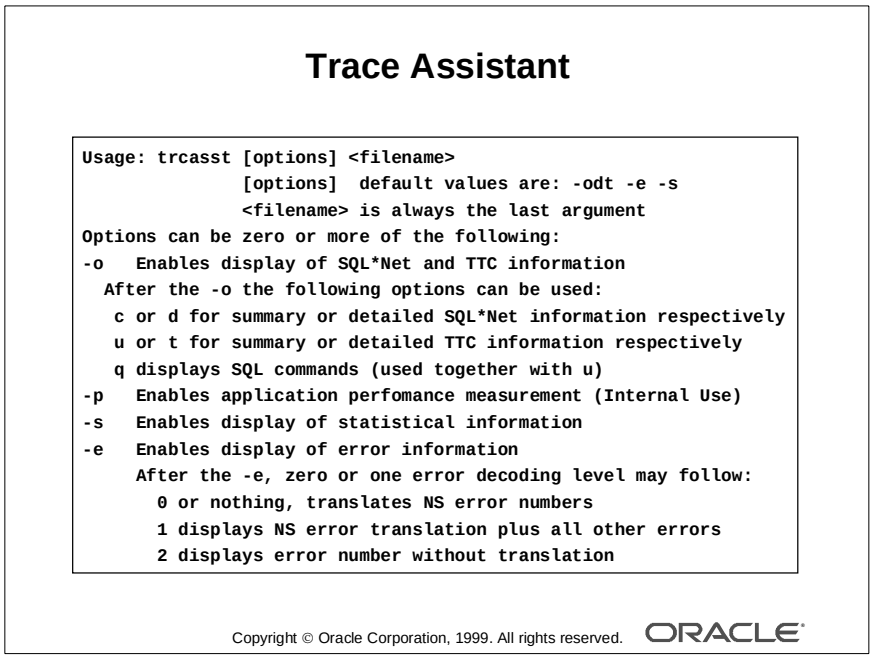
Evaluating trace files either manually or by using the Trace Assistant tool helps you to diagnose and troubleshoot network problems

Net8 performs its functions by sending and receiving data packets. By specifying a tracing level of SUPPORT, you can view the actual contents of the Net8 packet in your trace file. The order of the packet types sent and received helps you to determine how your connection was established.

Trace Assistant provides the mechanisms to make the information stored in trace files more useful to you and offers an easy and fast way to understand and take advantage of this information.

Trace Assistant analyzes the events that have occurred on the network at not only the Net layer but also at the TTC (Two Task Common) layer. By doing so, Trace Assistant can provide useful statistics and data about the performance of an Oracle application across the network. This information is valuable in identifying potential performance bottlenecks.

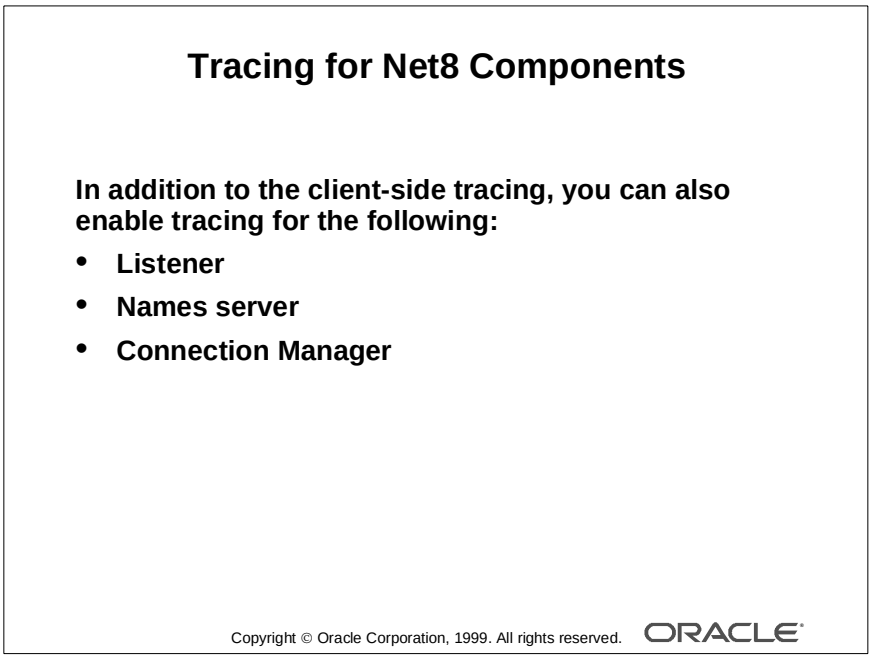
The amount and type of information provided by Trace Assistant is selected by the user, through the setting of command line options.



Trace Assistant Options

Trace Assistant can be run against an existing trace file with a number of options. The output is summarized and in a more readable format than in a regular trace file. An example could be a sample of statistics to see how many packages have been transferred.

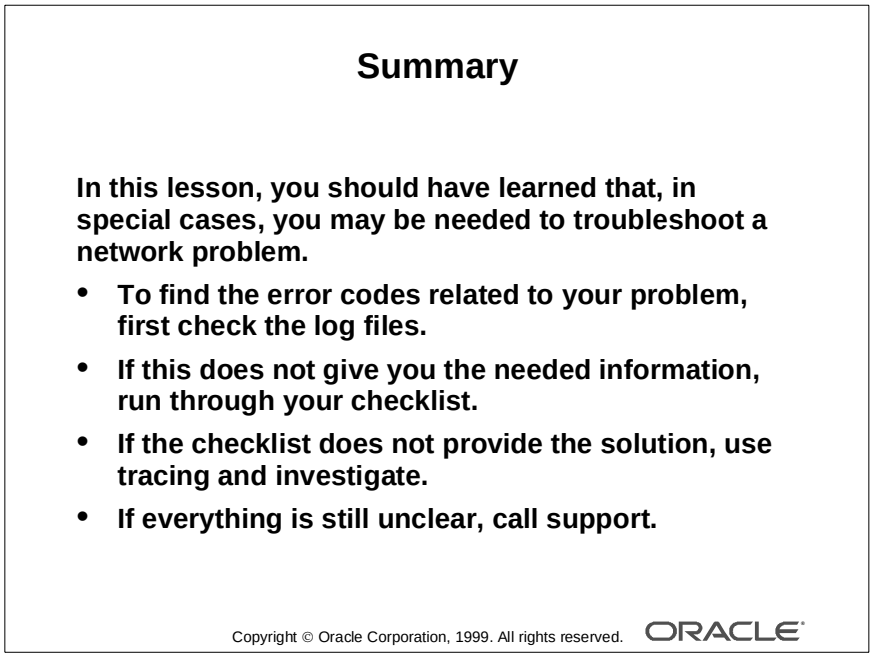
```
trcasst -s cli.trc
Trace File Statistics:
------------------------------------------------------------------------------------------
SQL*Net:
        Total Calls:        73 sent         109 received,     53 upi
        Total Bytes:        8082 sent,      8471 received,
        Average Bytes:      110 sent,       811 received,
        Maximum Bytes:      504 sent,       2048 received,
GRAND TOTAL                 sent: 73        received:   109
PACKETS
```



Tracing for Other Net8 Components

In addition to tracing and logging parameters specified for the client side, it is possible to use tracing for all other components of Net8. All parameter settings can be found in *Oracle8i Server Networking and Security Guide, Release 8.1.5*.

Summary


Summary
In this lesson, you should have learned that, in special cases, you may be needed to troubleshoot a network problem.
• To find the error codes related to your problem, first check the log files.
• If this does not give you the needed information, run through your checklist.
• If the checklist does not provide the solution, use tracing and investigate.
• If everything is still unclear, call support.
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Frequently Asked Questions

The following discussion points are taken from the most frequently asked questions regarding topics in this lesson. This information is retrieved and analyzed in cooperation with Oracle Worldwide Support.

Problem

How is Net8 tracing different from SQL*Net 2.*x* tracing?

Solution

Net8 tracing includes all the previous SQL*Net 2.*x* options and adds Oracle Trace capability. You can manage your trace information through an Oracle Enterprise Manager console in an Oracle Trace repository.

Problem

What are the `*.cdf` and `*.dat` files?

Solution

The `*.cdf` and `*.dat` files are created by Oracle Trace. To read these, you must use the *trcfmt* utility. It will extract and format the data inside binary files (`.cdf` and `.dat` extensions) into text files (`.txt` extension). To use this tool, type the following at any command line prompt:

```
trcfmt collection.cdf
```

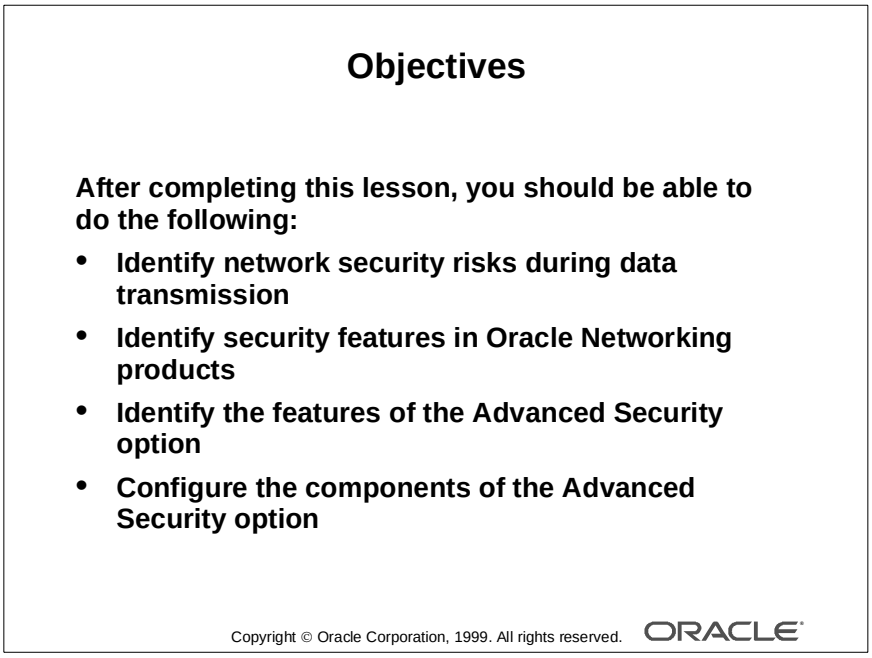
Note: "collection" is the name, with the path included if "collection" is not in the directory in which the `.cdf` and `.dat` files exist. If you have collected several processes within a single `.cdf` or `.dat` file, they are extracted to files with the name of `process_id.txt`.

Once you have formatted the data to text, you can view the data through any text editor.

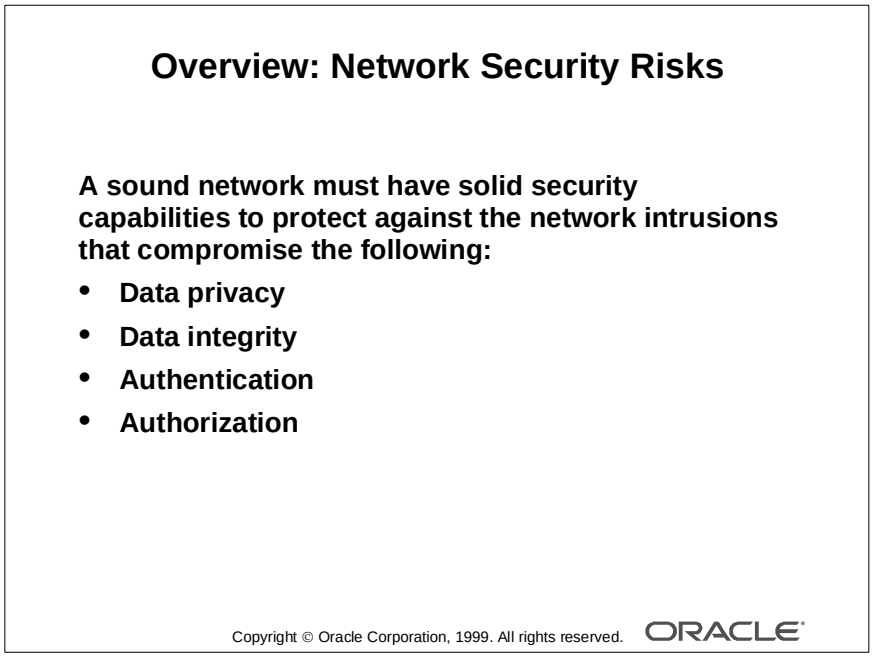
10

Security in the Network Environment

Objectives


Objectives
After completing this lesson, you should be able to do the following:
• Identify network security risks during data transmission
• Identify security features in Oracle Networking products
• Identify the features of the Advanced Security option
• Configure the components of the Advanced Security option
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

Overview: Network Security Risks

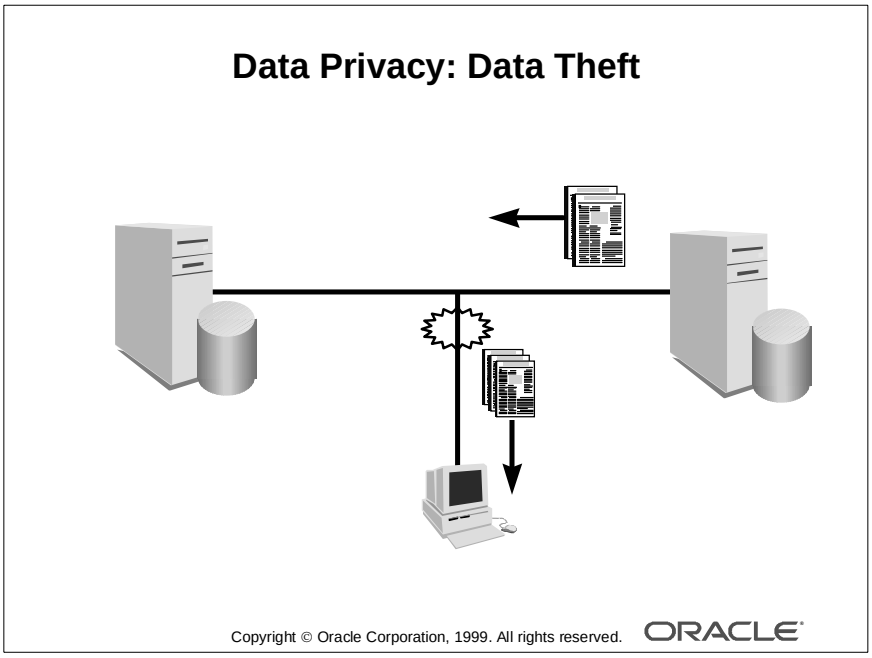


Challenges in a Distributed Environment

The principle challenges in a networked environment include maintaining the following:

- Data privacy: Ensuring that data is not disclosed or stolen during transmission
- Data integrity: Ensuring that data is not modified or disrupted during transmission
- Authentication: Confidence that users', hosts', and clients' identities are correctly known
- Authorization: Permitting a user, a program, or a process to access an object or set of objects

Data Privacy: Data Theft



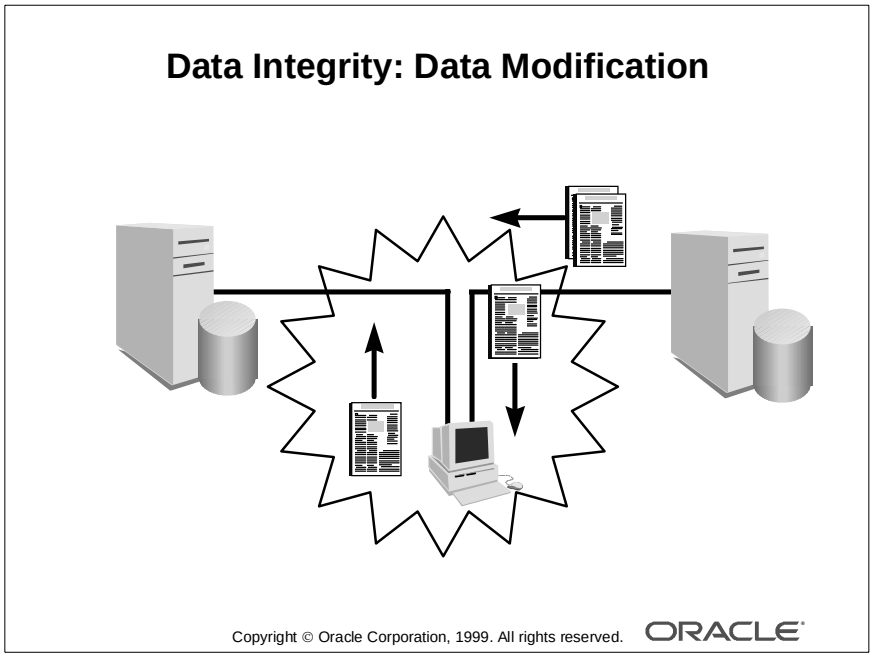
Data Theft

In most networks, it is easy to tap into the network cable with a workstation or PC and capture all of the traffic flowing across the wire. This applies to almost all types of networks, including both wide area and local area networks.

Legitimately connected users are as much a threat as attackers who connect illicitly in wiring closets. Readily available software (from both commercial sources and bulletin boards) makes this type of theft easy.

Both data and passwords are at risk; Oracle7 release 7.1 and later encrypt passwords at login time (but not during password change operations).

Data Integrity: Data Modification

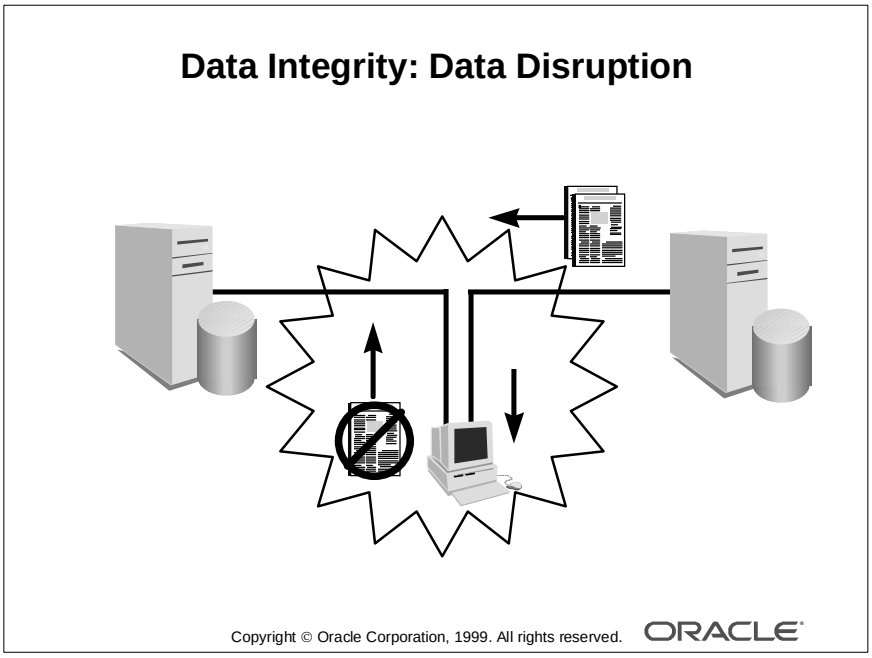


Data Modification

Almost as easily as an attacker can steal data, the attacker can “cut” the network (either logically or physically) and put a workstation between two communicating parties (the client and the server). Many common network protocols (such as TCP/IP) are vulnerable to this type of attack, which uses software that “tricks” the network routing software into sending all packets through the attacker’s workstation.

Once this is done, the integrity of the data flowing between the two unsuspecting machines can be compromised in a number of ways. Data packets can be modified to contain different data (for instance, a bank deposit could be modified to be $2,000 instead of $1,000).

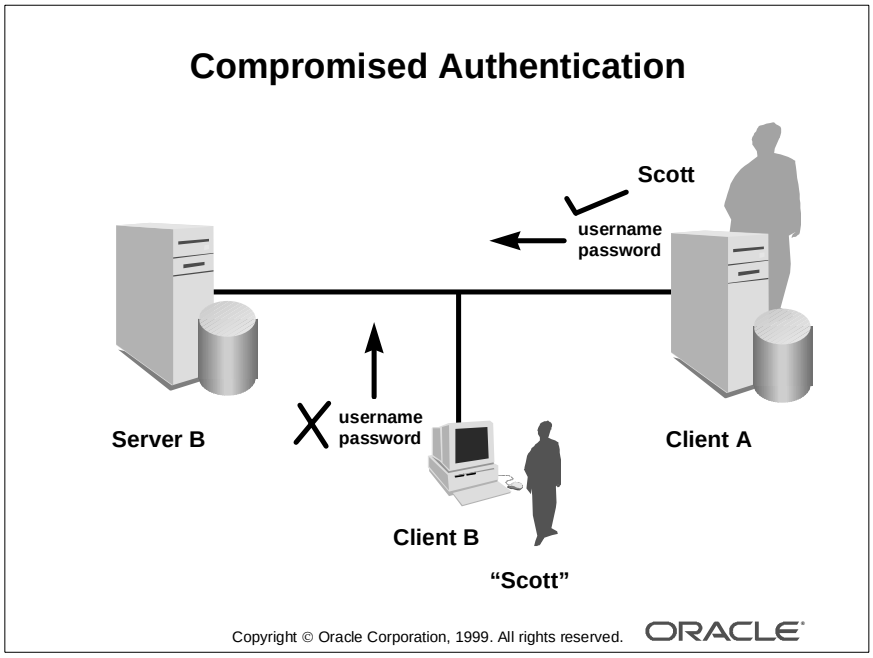
Data Integrity: Data Disruption



Data Disruption

Just as dangerous as modifying data packets is disrupting whole packets. Commands can be sent several times (for instance, a bank deposit transaction for $1,000 could be sent to the server several times). The commands can be deleted from the data stream (for instance, a bank withdrawal transaction could be prevented from reaching the server). As such, computerized systems are increasingly becoming the target of industrial sabotage.

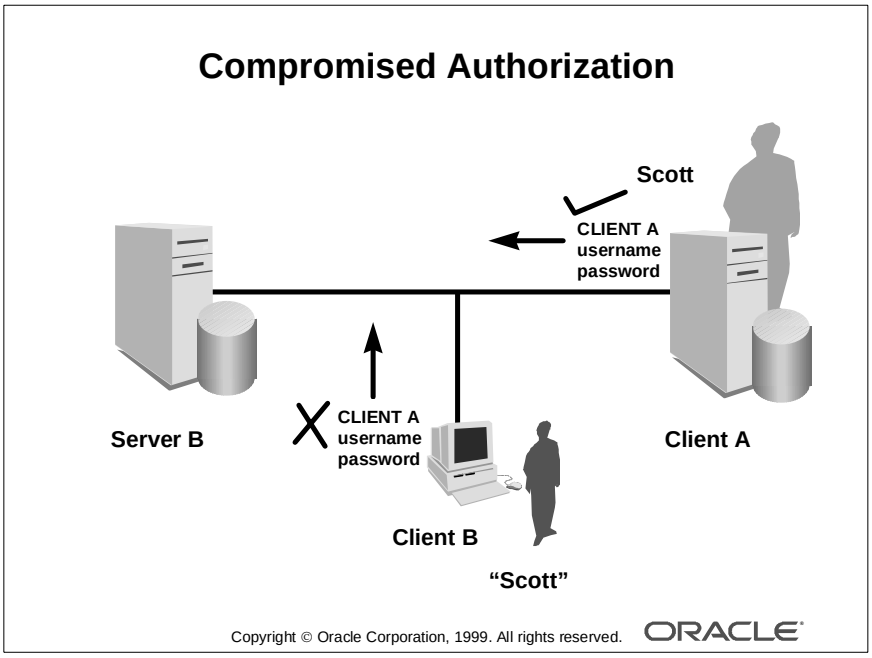
Compromised Authentication



Compromised Authentication

Establishing user identity is also of primary concern in distributed environments; otherwise, there can be little confidence in limiting privileges by user. For example, unless you have confidence in user authentication mechanisms, how can you be sure that user Scott connecting to Server B from Client B really is user Scott?

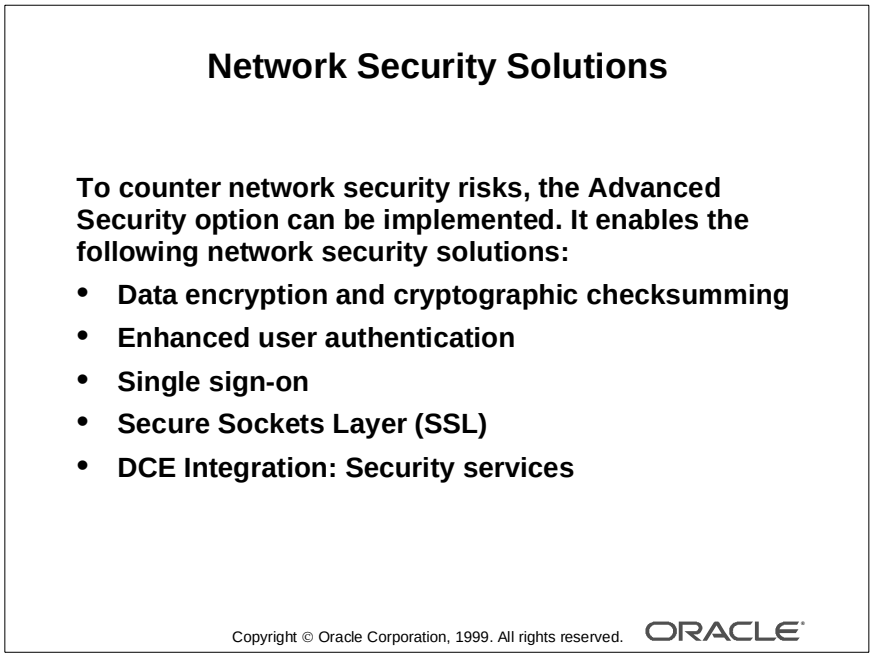
Compromised Authorization



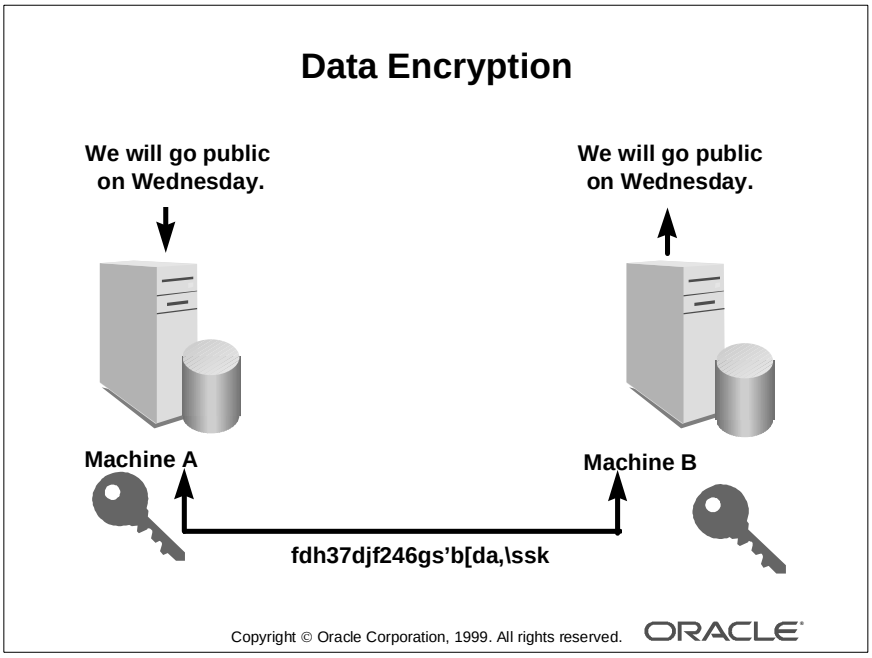
Compromised Authorization

You also need to have confidence in the way clients and servers are made known to one another over the network, so that you have assurance not only that user Scott is who he says he is, but that Client B and Client A are also what they claim to be.

Network Security Solutions



Data Encryption



Data Encryption

Encryption is a technique that scrambles data using a key. Without the key, the data cannot be unscrambled. Special hardware is used to encrypt data.

Two algorithms are used for encryption. They are as follows:

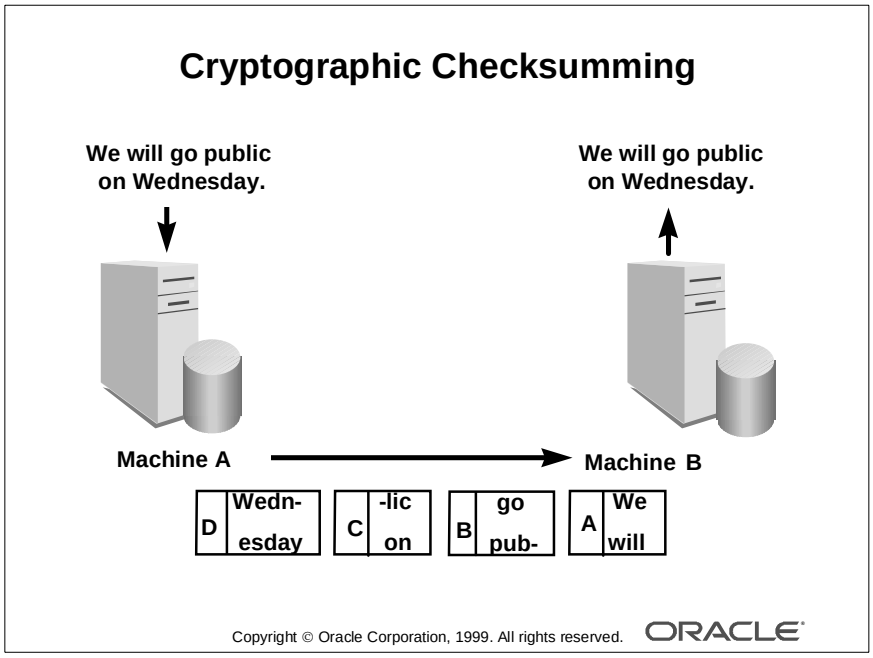
- DES: Data Encryption Standard
- RSA RC4: RC4 algorithm developed by RSA Data Security, Inc.

Key Lengths The longer the length of the key, the stronger the encryption. The stronger the encryption, the harder it is to break the code.

Available key lengths are as follows:

- DES 56 bits
- DES 40 bits
- RC4 128 bits
- RC4 56 bits
- RC4 40 bits

Cryptographic Checksumming



Checksumming

Function of Sequencing

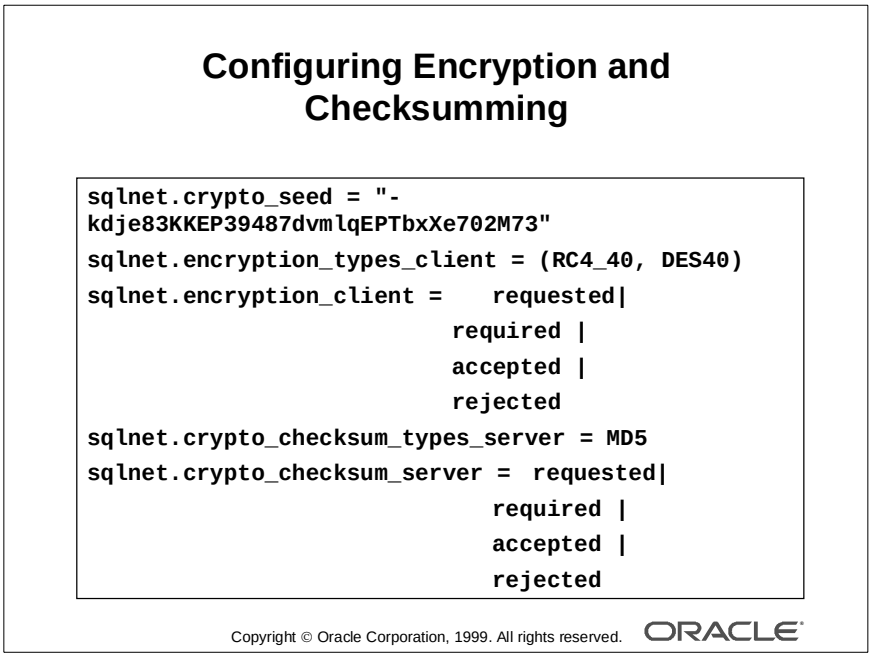
- Labels each packet A, B, C before it transmits.
- When the packet arrives, the server checks to see if the packet is in order.

Function of the MD5 Algorithm The message digest algorithm is available from RSA. It performs the following tasks:

- Makes a hash calculation on the contents of each packet
- Records the value at the end of the packet

The server makes the same hash calculation to see if the value has changed.

Configuring Encryption and Checksumming



Encryption and Checksumming Parameters

You can negotiate whether to turn on encryption or checksumming by specifying four of the Oracle Advanced Security option configuration parameters.

Note: The encryption seed value can be up to 70 characters. The greater the number of characters used and variation of the characters, the greater the security. DES40 is used in the example in the slide.

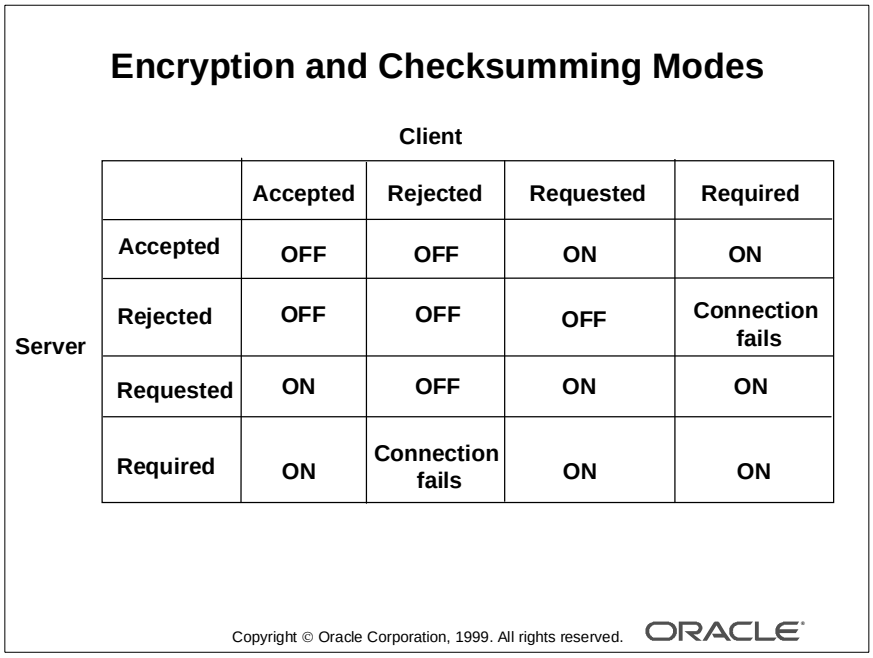
DES 56 and RC4 128 can also be used.

In the example in the slide, the machine is configured as a client using encryption and a server using checksumming.

Default Encryption and Checksumming Values

- 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, the Oracle Advanced Security option defaults the value to ACCEPTED.
- If no encryption or checksumming algorithm is specified on the server encryption, client encryption, server checksum, or client checksum pages, the server side of the connection uses the first algorithm in its own list of installed algorithms that also appears in the client's list of installed algorithms.
- Encryption and checksumming function independently of each other; encryption can be activated while checksumming is off and vice versa.

Encryption and Checksumming Modes



Encryption and Checksumming Modes

Based on a combination of client and server configuration parameters, the table shows whether or not encryption or checksumming is turned on. A description of each of the modes is defined below:

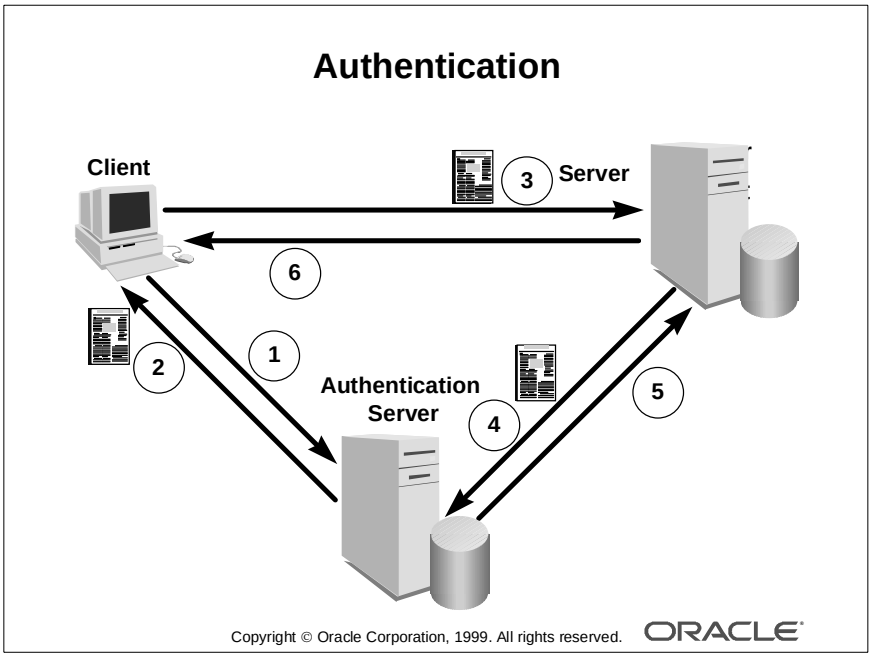
- Accepted: Service becomes active if the partner states that the service is requested or required.
- Rejected: Service is not activated at all, and the connection fails if the partner states that the service is required.
- Requested: Service becomes active if the partner states that the service is accepted, requested, or required.
- Required: Service becomes active only if the partner accepts or requires the service. The connection fails if the partner states that the service is rejected.

Algorithm Negotiation

In any network connection, it is possible that both ends (client and server) may support more than one encryption algorithm and more than one cryptographic checksumming algorithm. When each connection is made, the server decides which algorithm to use, if any, based on which algorithms are available on each end of the connection and what preferences have been specified in the Net8 configuration files.

When the server is trying to find a match between the algorithms it has made available and the algorithms the client has made available, it picks the first algorithm in its own list that also appears in the client's list. If one side of the connection does not specify a list of algorithms, all the algorithms that are installed on that side are acceptable.

Authentication



Network Authentication Services

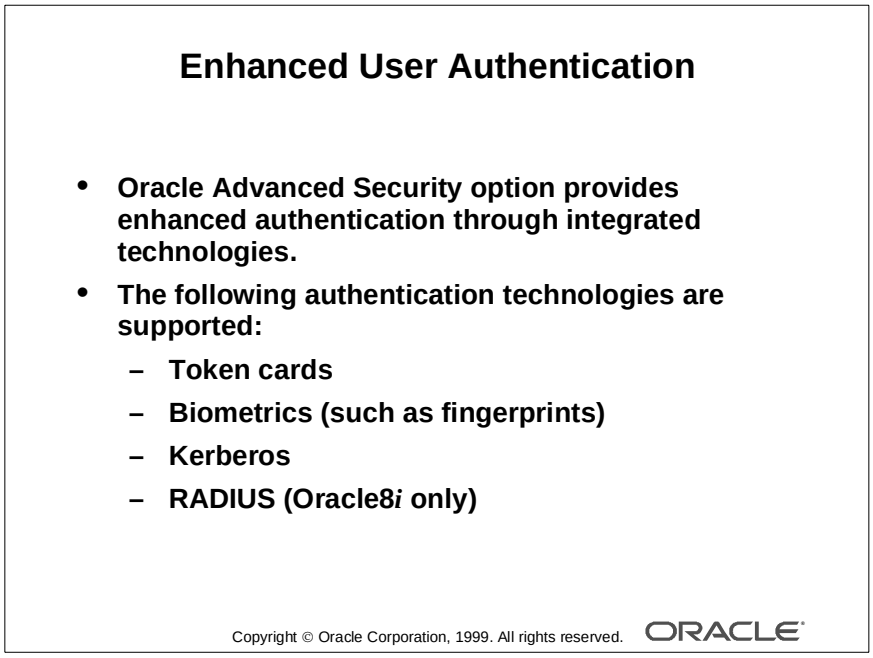
Confidence in the identity of users and hosts can be achieved by using a centralized, secure authentication service rather than relying on hosts and users identifying themselves to one another directly.

Logical Representation of Authentication Events

Within a typical authentication request in a client-server connection, the following events occur:

1. The client requests authentication from the server.
2. The authentication server passes a ticket or credentials back to the client.
3. The client takes these credentials and passes them to the server while asking for a service, such as a connection to a database.
4. To verify the credentials, the server sends the credentials back to the authentication server.
5. If the authentication server accepts the credentials, it notifies the server.
6. The server provides the requested service; otherwise, the request is denied.

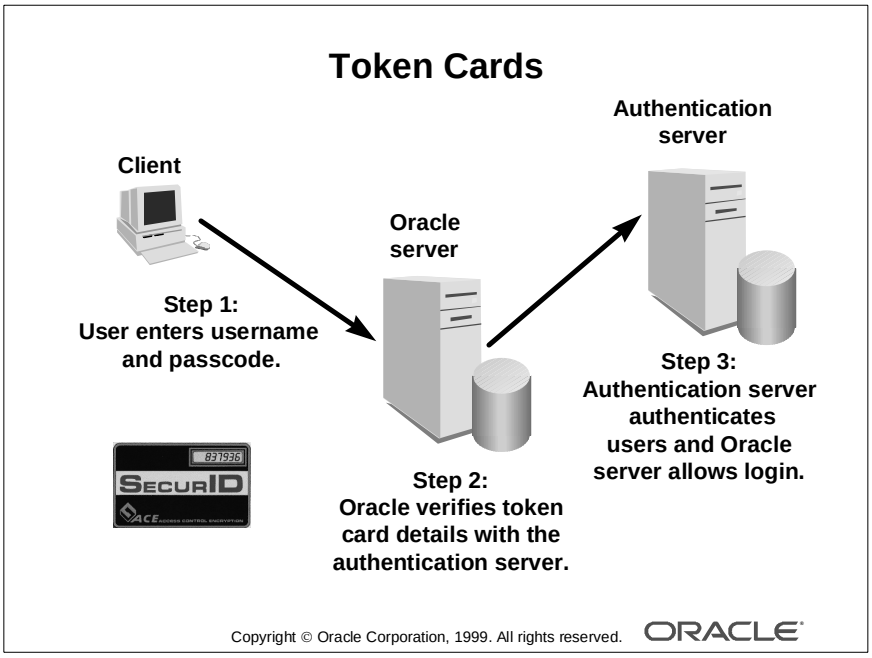
Enhanced User Authentication



Authentication Adapters

Support for authentication services is provided through authentication adapters, which are very much like the existing Net8 protocol adapters. Authentication adapters integrate below the Net8 interface so that existing applications can take advantage of new authentication systems transparently, without any changes to the application.

Token Cards



Benefits of Token Card Authentication

- Ease of use for users, who need only remember a personal identification number (PIN) instead of multiple passwords
- Ease of password management (a card rather than multiple passwords)
- Enhanced password security (To masquerade as a user, a malefactor would have to have the card as well as the PIN required to operate it.)
- Enhanced accountability through a stronger authentication mechanism

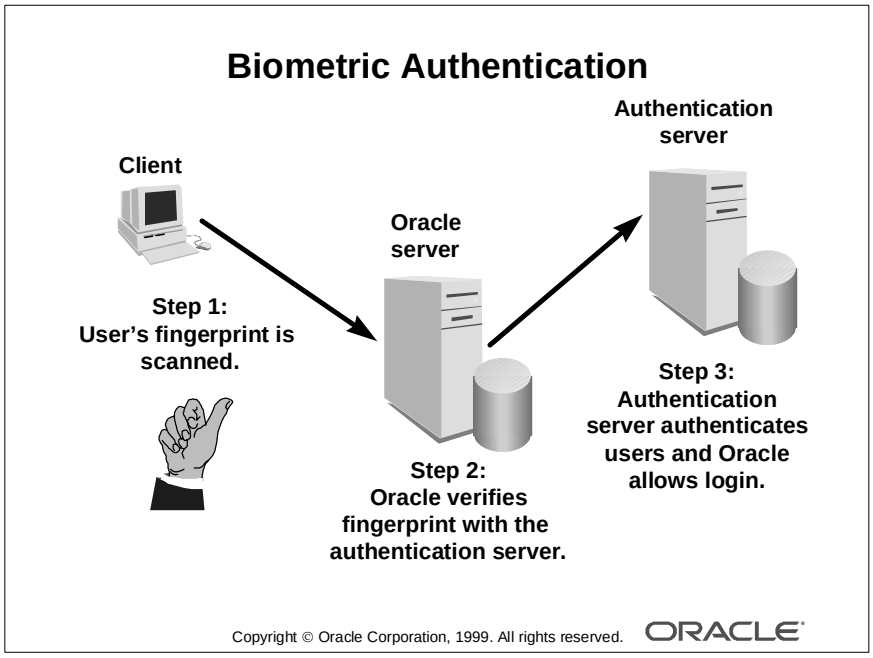
Supported Token Cards

The Oracle Advanced Security option supports the Security Dynamics' SecurID card. SecurID provides two-factor user identification.

- Factor one is something the user knows, a PIN.
- Factor two is something the user possesses, the SecurID card.

Single-use access codes change automatically every 60 seconds, and no two cards ever display the same code at the same time.

Biometric Authentication



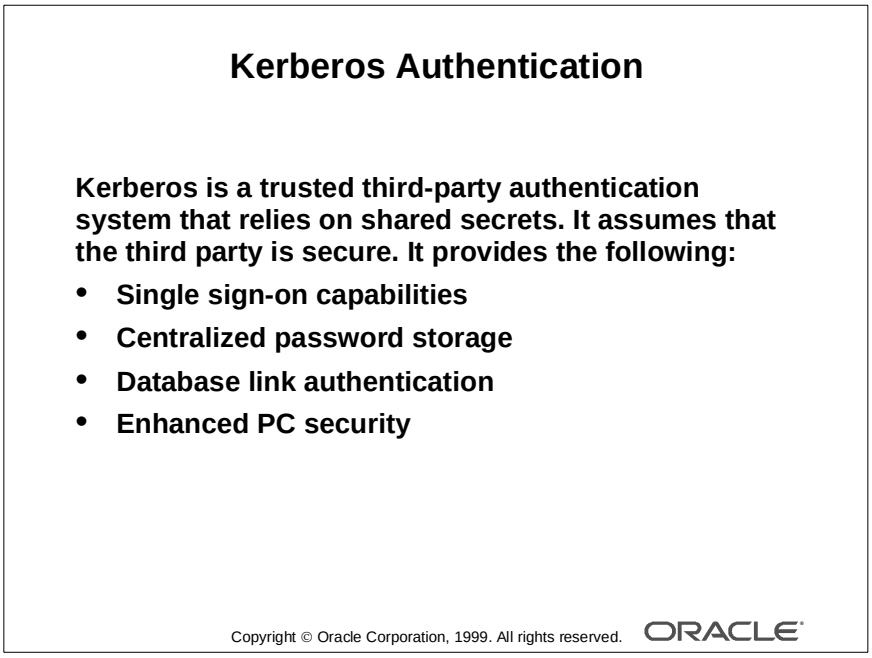
Biometric Authentication

The Oracle Advanced Security option provides support for the Oracle Biometric Authentication Adapter. The authentication adapters are used on both the clients and the database servers to communicate biometric authentication data between the authentication server and the clients.

Supported Biometric Authentication Device Because a user's fingerprint is unique, only the "true" user can match the stored fingerprint scan. Oracle supports the Identix Touchsafe II Biometric device.

- The fingerprint of each user is enrolled into the security system.
- To log in, users specify a service and place a finger on the Touchsafe II fingerprint reader.
- The fingerprint is scanned and compared to the previously stored fingerprint.
- If the scan matches, the user is authenticated.

Kerberos Authentication

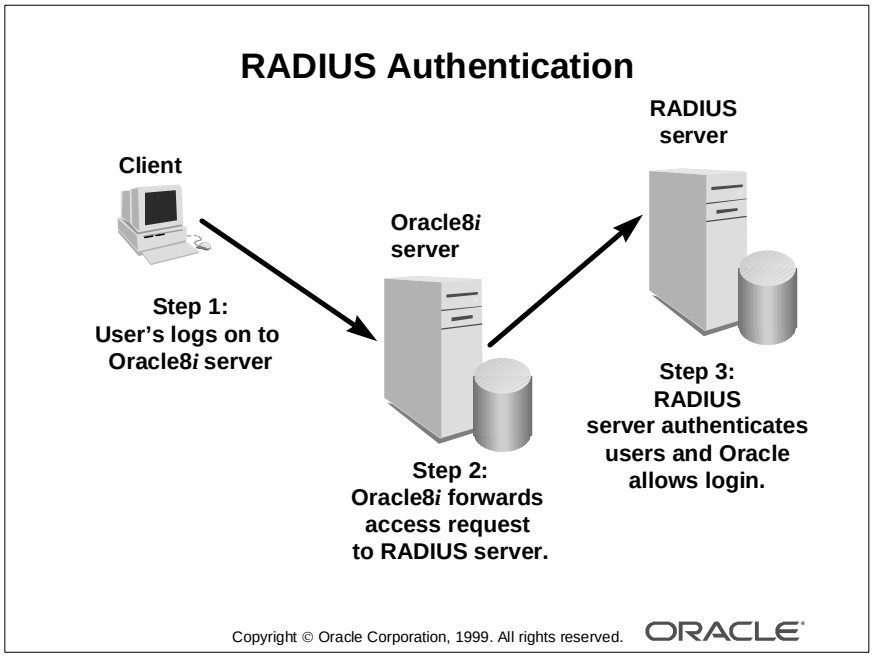


Kerberos Adapters

Two adapters for the Kerberos authentication service are provided by Oracle:

- Kerberos Authentication Adapter
- CyberSAFE Challenger Authentication Adapter

RADIUS Authentication



RADIUS Authentication

Remote Authentication Dial-In User Service (RADIUS) is a flexible, lightweight protocol that provides the followings services:

- Authentication
- Authorization
- Accounting
- Centralized user information

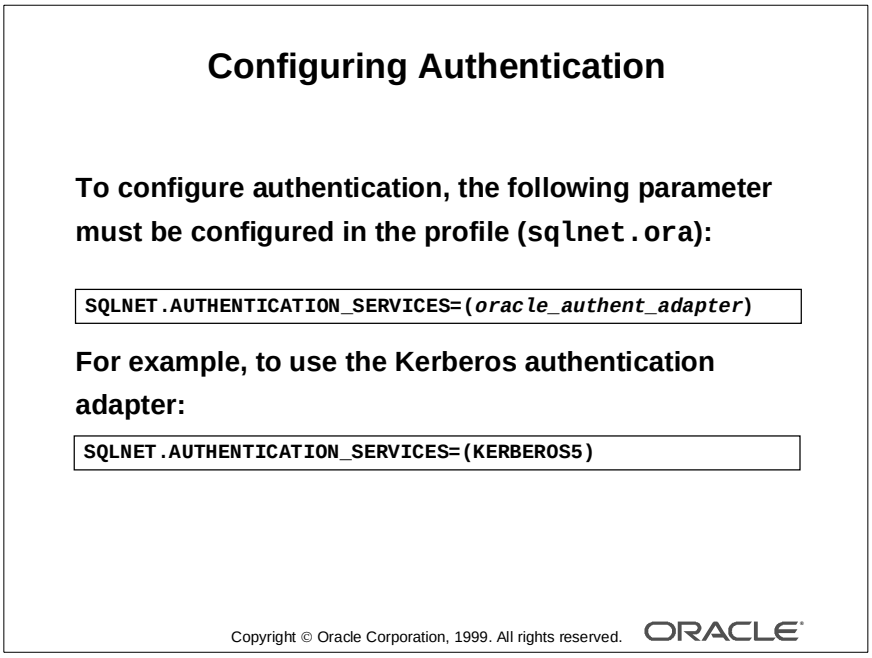
Note: RADIUS is currently supported only with Oracle8*i*.

Supported RADIUS services

Various RADIUS server vendors are supported, as follows:

- Lucent
- ActivCard
- Funk
- Vasco
- Bay Networks
- Secure Computing

Configuring Authentication



Configuration Recommendations

- Set REMOTE_OS_AUTHENT to FALSE.

  When configuring the Oracle authentication adapters, it is strongly recommended that you add the following parameter to the initialization file (`init.ora`) used for the database instance:

  ```
  REMOTE_OS_AUTHENT= FALSE
  ```

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

  Note: Setting REMOTE_OS_AUTHENT to TRUE can create a security hole, because someone using a nonsecure protocol (for example, TCP) can perform an operating system–authorized login (formerly referred to as an OPS$ login).

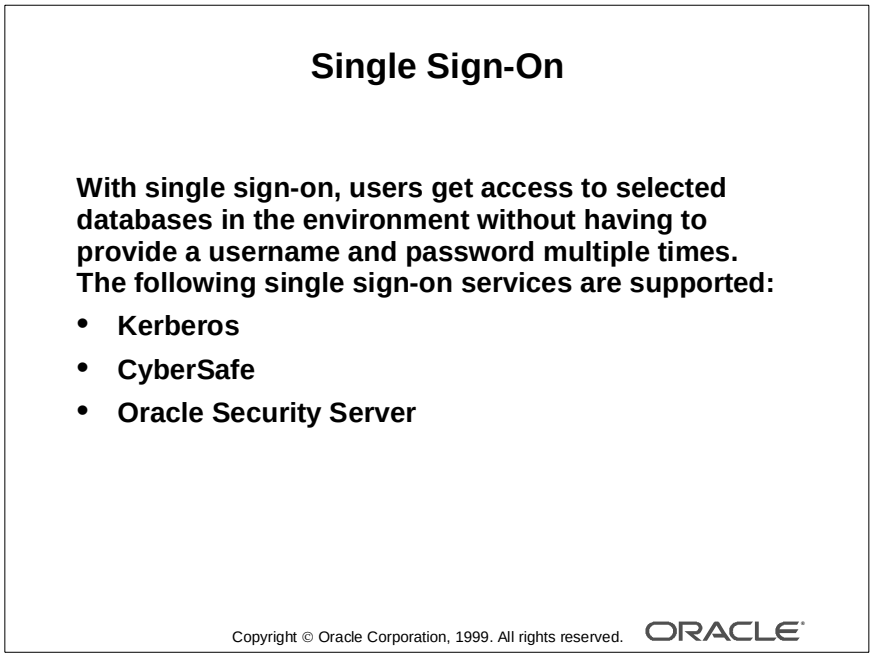
  If the SQLNET.AUTHENTICATION_SERVICES parameter is set to NONE in the `sqlnet.ora` file on the server or client, the database attempts to use the username and password provided to log the user on. But, if REMOTE_OS_AUTHENT is set to FALSE, the connection fails.

- Set OS_AUTHENT_PREFIX to Null.

  Authentication service–based user names can be long, and Oracle usernames are limited to 30 characters. So, it is strongly recommended that you enter a null value for the OS_AUTHENT_PREFIX parameter in the initialization file (`init.ora`) used for the database instance:

  ```
  OS_AUTHENT_PREFIX=""
  ```

Single Sign-On



Single Sign-On

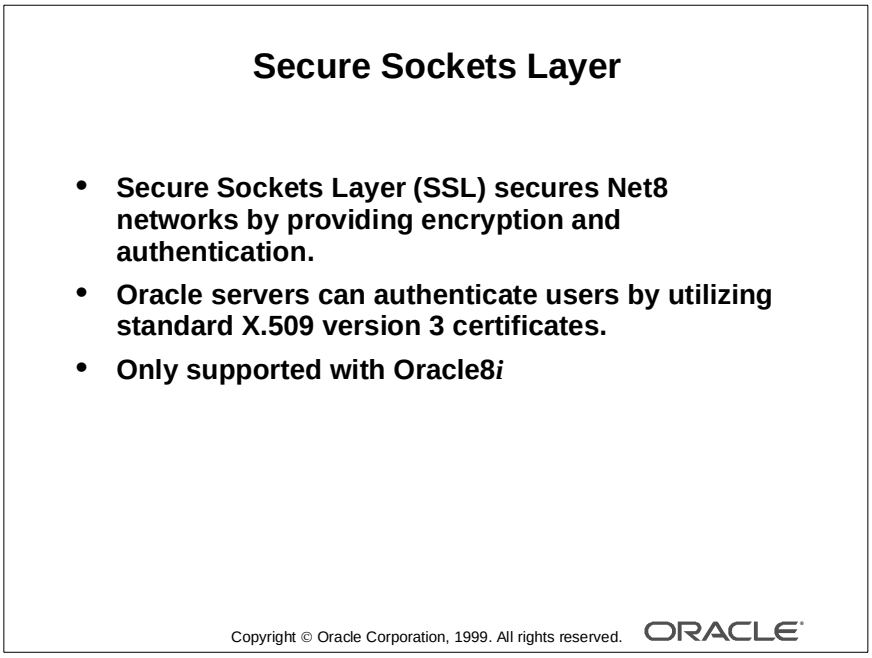
Network authentication services can provide the benefit of single sign-on for users.

Users generally respond to multiple accounts in one of two ways:

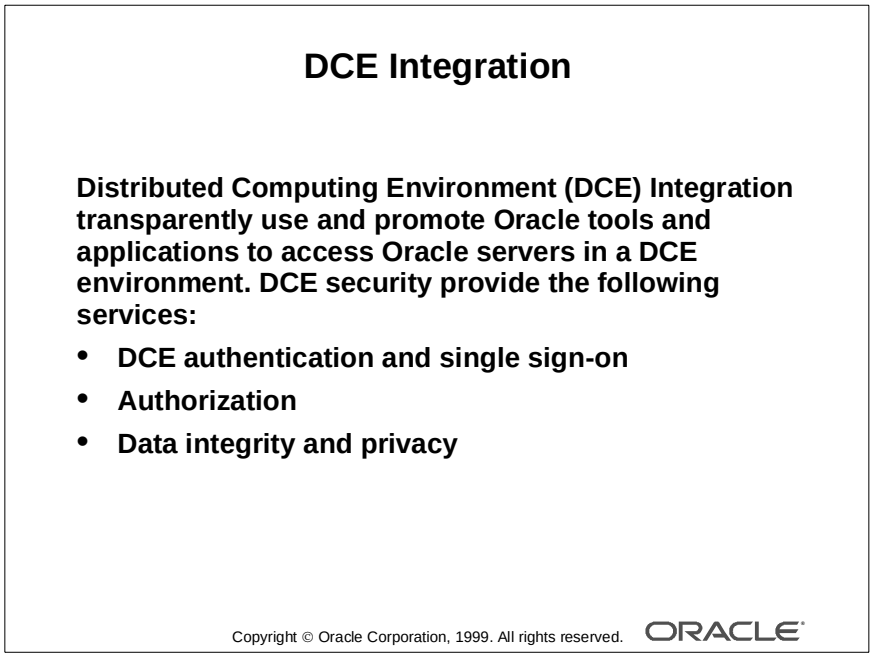
- If they can choose their own passwords, they may standardize them so that they are the same on all machines (which results in a potentially large exposure in the event of a compromised password).
- If they use passwords with slight variations (which may be easily guessed from knowing one password), they may just write them down or forget them, either of which severely compromises password secrecy and service availability.

Providing a single sign-on, so that users can access multiple accounts and applications with a single password, eliminates the need for multiple passwords for users and simplifies management of user accounts and passwords for system administrators.

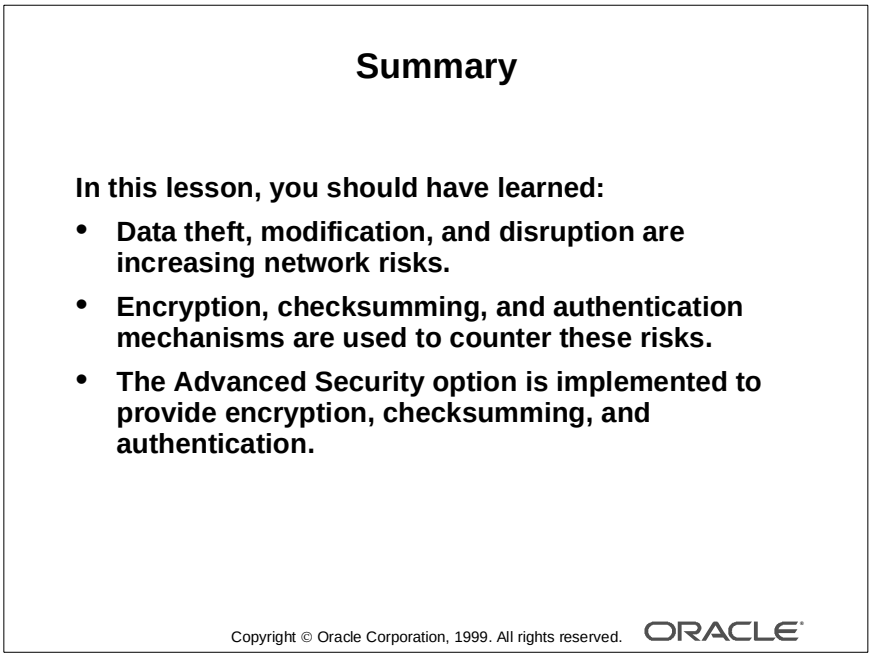
Secure Sockets Layer



DCE Integration



Summary


Summary
In this lesson, you should have learned:
• Data theft, modification, and disruption are increasing network risks.
• Encryption, checksumming, and authentication mechanisms are used to counter these risks.
• The Advanced Security option is implemented to provide encryption, checksumming, and authentication.
Copyright © Oracle Corporation, 1999. All rights reserved.
ORACLE

A

Practices

Practice 3

Before beginning this practice, your instructor should provide you with the server name on which you will configure your listener and a port number on which your listener will be listening. Throughout this practice, replace all the occurrences of *nn* with the number assigned to you by the instructor.

1. Create a listener lsnr*nn* using Net8 Assistant on the client PC. The listener must be configured for the server provided by the instructor; this server contains an Oracle database U*nn*. The listener must also be configured for the TCP/IP protocol only and must listen for incoming connections on the port provided by the instructor. Do not configure the listener to use prespawned dedicated servers.

   Note: For this classroom setup, the listener configuration file must be created on the client PC using Net8 Assistant and, in later steps, transferred via FTP or similar file transfer application on the server.
2. View the contents of the `listener.ora` file to verify the configuration details.
3. Transfer the listener configuration file from the client to the server using FTP or any similar file transfer utility. The listener configuration file must be transferred from the client to the directory on the server that will be searched for Net8 configuration files.
4. Manually add a parameter to the `listener.ora` file so that the listener creates a log file called `lsnr`*`nn`*`.log` in the `$TNS_ADMIN/log` directory. If the `log` directory does not exist in the `$TNS_ADMIN` directory, create it.

   Note: This step is necessary for the classroom setup. If an entry for the listener log file is not inserted into your `listener.ora` file, your listener will attempt to write to the default listener log directory.
5. Using the Listener Control utility, start your listener.
6. View the contents of the listener log file.
7. Start up your database instance.

Practice 4

If you are unsure of the name of your client, please ask the instructor how to obtain the name.

1. Using the local naming method in Net8 Assistant, configure the client side. The protocol used is TCP/IP. The port number for the listener you should connect to is the port number provided for the listener in the previous practice. SID is U*nn*.

   Note: Use the Oracle8*i* Release 8.0 Database SID method to connect instead of the Oracle8*i* Release 8.1 Service Name method.
2. Test that the service is operational.
3. Investigate the contents of the `tnsnames.ora` file.
4. Connect to the server as `system/manager` using SQL*Plus and verify that you are connected to the correct instance by querying the V$INSTANCE view. When everything is working, have the group or person next to you edit your `tnsnames.ora` file in order to provoke an error either by specifying a wrong port to connect to or a wrong node name, database, or service name.
   When they are done, try to correct the error by regenerating the `tnsnames.ora` file by hand.
5. Using Net8 Assistant, configure your client to make connections to the listener on the server using the new service name instead of the instance name.

   Hint: Use Oracle8*i* release 8.1–compatible identification.
6. Connect to the server as `system/manager` using SQL*Plus and verify that you are connected to the correct instance as was done in step 4.
7. Shut down the database instance and the listener, and configure your database instance (U*nn*) for automatic instance registration.

   Hint: Look at the parameters SERVICE_NAMES and INSTANCE_NAME in the `init`*`SID`*`.ora` file.
8. Check that the instance has been registered and which listener has the handle.
9. Restart the database and have it register with your working listener `lsnr`*`nn`*.
   Hint: Look at the parameters LOCAL_LISTENER in the `init`*`SID`*`.ora` file.
   Create a new `tnsnames.ora` manually do not use the Net8 Assistant.

Practice 6

1. Create a Names server on your local PC node and view the contents of the Names configuration file. Do not configure the Names server with a region database. Use 1575 as the port on which the Names server will accept incoming name resolution requests. Make sure the DB_DOMAIN parameter is set to NULL.

   Note: In this class, the Names server is created on your local PC node; therefore, your node will act as a client as well as a Names server. Ideally, a Names server should reside on a dedicated middle-tier node on the network.

   Bring up the DOS box and enter `hostname` to determine the name of your client PC.

2. Start up the Names server, and add a service name for the database U*nn* to the Names server. Use the attributes for the database service name defined in your existing `tnsnames.ora` file.

   Note: Do not use Oracle8*i* release 8.0–compatible identification to connect

3. Using Net8 Assistant, configure your client profile to use Oracle Names, and assign the previously created Names server as the preferred Names server for your client profile.
4. Connect to U*nn* using SQL*Plus and verify if the Names server is operational by using ONAMES as the only naming method.

5. Using the Windows Control panel, shut down the Names server. After the Names server is shut down, perform the following functions using the Names Control utility:
   a. View the status of the Names server by using the Names Control Utility.
   b. What is the error message, and why does it occur?
   c. Set the server to the Names server that you previously configured.
   d. What is the error message, and why does it occur?
   e. Start up the Names server.
6. If you do not plan to attempt the optional practice, configure your profile as before, so that it uses the TNSNAMES naming method rather than ONAMES.

Optional Practice

7. Ensure that the Names server is running. Restart the Net8 Assistant utility, then reconfigure your Names server to use a region database. The region database should now reside on the U*nn* database on the server, while the Names server still resides on the client. Use the SYSTEM user to run the Names server initialization script (Typically you would use a separate user for this purpose).

   Note: Do not use Oracle8*i* release 8.0–compatible identification when connecting.

8. Add a service name for U*nn* to the newly created region database on U*nn* itself.
   Note: Use a different service name with the same connection attributes.
9. Attempt to connect to U*nn* using SQL*Plus after the service name is added.

10 Query the NAME_P and ZD_VALUE1_P columns of the ONRS_REGION table to view the newly created values.

11 Configure your profile as before, so that it uses the TNSNAMES naming method rather than ONAMES, and ensure that you can connect using the previously used TNSNAMES naming method.

Practice 7

Start a Telnet session connecting to the server where your database resides.

1. Configure and start up the multithreaded server for your database so that you have one dispatcher listening for TCP/IP connections and one shared server to serve requests.

   Specify the maximum dispatchers as two and maximum shared servers as six.
2. To verify that a dispatcher is associated with your listener, use the LSNRCTL utility and issue the command: lsnrctl SERVICES *your_listener_name*.
3. Before making a connection, query the view V$CIRCUIT from SQL*Plus connecting as `system/manager` in your Telnet session to see if it contains data. This view has an entry for each connection session currently using the MTS.
4. Make a connection using SQL*Plus, connecting as `system/manager` from your client to the server, and check V$CIRCUIT view again.

   After you have verified the connection, exit SQL*Plus.
5. Do the following:
   a. Query the V$SHARED_SERVER view to see how many shared servers have been started.
   b. Query the V$DISPATCHER view to see how many dispatchers have been started.
6. Make two connections using SQL*Plus, connecting as `system/manager` from your client to the server using MTS.
   a. Has the number of shared servers increased?
   b. Why or why not?
7. How could the number of shared servers increase?
8. Do the following:
   a. Add one more dispatchers to handle TCP requests.
   b. Verify that an additional dispatcher has been added.
   c. Decide whether you can add a third dispatcher.
9. Shut down your database and:
   a. Configure connection pooling so that the pooling takes effect when a third connection is established.
   b. Verify that the connection pooling feature is working.

Practice 8

Because of setup restrictions in the classrooms, you will be simulating the middle tier by using your local machine. In a real-world environment, it is recommended to use a multitier model.

Before starting this practice, make sure that you have no connections established to the server side.

Start a Telnet session and connect to your server. Log in and check the usage of sockets/ports by issuing the command `netstat -a | grep` *your_client_name* `| grep ESTABLISHED` where *your_client_name* is the host name of your PC.

You should see something like:

```
netstat -a | grep wwedf-162 | grep ESTABLISHED
wwedf-162.us.oracle.com.v7dist33 8533 0 8760 0 ESTABLISHED
```

The output line will show the established connections from your client using the TCP protocol.

This indicates the socket used by your Telnet session. If you get more than one line, you have more than one connection established to the server.

In this lab, you will look at the socket usage when using the multiplexing feature of Connection Manager.

1. On your local machine configure and start Connection Manager.
2. Edit your `tnsnames.ora` file, and add a service name entry that will use Connection Manager to connect to the database on your server.
3. Test your service name by establishing two connections to the server using SQL*Plus.
   a. From your Telnet session, check how many sockets you are now using.
   b. What explains the number of sockets used?
4. Exit your SQL*Plus sessions and verify that you just have one socket where a connection is established.
5. From your Telnet session, shut down your database on the server and configure for multiplexing.
6. Start up your database again and make two SQL*Plus connections from your client.
   a. Check how many sockets you use now.
   b. What explains the number of sockets used?
   c. Close all the SQL*Plus sessions.
7. Do the following:
   a. Configure the Connection Manager so that your PC will not be restricted from accessing any destination.
   b. Describe what happens when you try to connect.
8. Stop the Connection Manager, shut down the database server, and disable the multithreaded server.

Practice 9

Before starting this practice, the instructor will provide you with a trace directory and a log directory name.

1. Enable logging for your client. Specify the log directory and verify that the `sqlnet.ora` file is correct.

   You can either use Net8 Assistant or edit your `sqlnet.ora` file by hand.
2. Make a connection to the server using SQL*Plus, connecting as `system/manager`. Will a log file be generated?
3. Shut down the listener on the server side, then try to connect, and finally investigate the log file.
4. Start your listener again.
5. Enable tracing at SUPPORT level and make sure that the trace files have unique names. You can do this either by using Net8 Assistant or by editing the `sqlnet.ora` file.
6. Make a connection, exit the connection, and investigate the trace file.
7. Change directories to the assigned trace directory, and try running the Trace Assistant utility using the -o option, send output to the file `tst.txt`, and then investigate the file.

B

Practice Solutions

Practice 3 Solutions

Before beginning this practice, your instructor should provide you with the server name on which you will configure your listener and a port number on which your listener will be listening. Throughout this practice, replace all the occurrences of *nn* with the number assigned to you by the instructor.

1. Create a listener lsnr*nn* using Net8 Assistant on the client PC. The listener must be configured for the server provided by the instructor; this server contains an Oracle database U*nn*. The listener must also be configured for the TCP/IP protocol only and must listen for incoming connections on the port provided by the instructor. Do not configure the listener to use prespawned dedicated servers.

   Note: For this classroom setup, the listener configuration file must be created on the client PC using Net8 Assistant and, in later steps, transferred via FTP or similar file transfer application on the server.

   a. From the Windows NT Start menu on the client PC, select Programs—>Oracle - *Oracle Home*—>Network Administration—>Net8 Assistant.

   b. Click the Listeners folder in Net8 Assistant.

   c. Select Create from the Edit menu item, or click the "+" icon.

   d. Enter a name for your listener (lsnr*nn*) in the Choose Listener Name dialog box that appears and click OK.

      The name of the newly created listener will appear below the Listeners folder in the Net8 Assistant.

   e. Click the new listener name and select Listening Locations from the pull-down menu on the right side of the screen in Net8 Assistant, if not already selected.

   f. Click the Add Address button.

      A tab for the address details of the listener will appear.

   g. Select TCP/IP as the protocol, if not already selected.

   h. Enter the name of the server in the Host field, and the port number assigned for your listener in the Port field (the Port number is provided by your instructor). Leave the protocol stack support as the default value Net8 Clients.

   i. Select Database Services from the pull-down menu on the right side of the screen in Net8 Assistant.

   j. Click the Add Database button.

      A tab for the database on behalf of which the listener will listen for incoming connections will appear.

   k. Enter a name for the global database in the Global Database Name field (the Global Database Name is provided by your instructor).

   l. Enter the directory, defined as your `$ORACLE_HOME` directory on the server, in the Oracle Home Directory field (Issue the *env* command from the UNIX prompt to get the home directory).

   m. Enter your database system identifier (SID) (U*nn*) in the SID field.

   n. Save your configuration by selecting Save Network Configuration from the File menu item in Net8 Assistant.

2 View the contents of the `listener.ora` file to verify the configuration details.

```
# C:\ORANT\NETWORK\ADMIN\LISTENER.ORA Configuration
# File:c:\orant\NETWORK\ADMIN\listener.ora
# Generated by Oracle Net8 Assistant

LSNRnn =
  (DESCRIPTION =
(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=<port>))
    (PROTOCOL_STACK =
      (PRESENTATION = TTC)
      (SESSION = NS)
    )
  )

SID_LIST_LSNRnn =
  (SID_LIST =
    (SID_DESC =
      (GLOBAL_DBNAME = Unn.world)
      (ORACLE_HOME = <your_oracle_home_directory>)
      (SID_NAME = Unn)
    )
  )
```

3 Transfer the listener configuration file from the client to the server using FTP or any similar file transfer utility. The listener configuration file must be transferred from the client to the directory on the server that will be searched for Net8 configuration files.

a Telnet and log on to the server.

b Type the *env* command from the UNIX prompt on the server to view the environmental variables defined in your shell.

c Make a note of the directory path defined in the `$TNS_ADMIN` environment variable.

d Transfer the `listener.ora` file from the client into this directory on the server.

4 Manually add a parameter to the `listener.ora` file so that the listener creates a log file called `lsnr`*nn*`.log` in the `$TNS_ADMIN/log` directory. If the `log` directory does not exist in the `$TNS_ADMIN` directory, create it.

Note: This step is necessary for the classroom setup. If an entry for the listener log file is not inserted into your `listener.ora` file, your listener will attempt to write to the default listener log directory.

a Telnet and log on to the server.

b Change the directory to the `$TNS_ADMIN` directory.

```
$ cd $TNS_ADMIN
```

c If the log directory does not exist in this directory, create it with the following command:

```
$ mkdir log
```

d Edit the `listener.ora` file in the `$TNS_ADMIN` directory and add the following entry to it:

```
log_file_lsnrnn = <your_tns_admin_directory>/log/lsnrnn.log
```

5 Using the Listener Control utility, start your listener.

a Enter the following command after establishing a Telnet session to the U*nn* account:

```
$ lsnrctl start lsnrnn
Starting /oracle/product/bin/tnslsnr: please wait...
TNSLSNR for Solaris: Version 8.1.5.0.0 - Production
System parameter file is <tns_admin>/listener.ora
Log messages written to <tns_admin>/log/lsnrnn.log
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=<
port>))
(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=<
port>))
(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))
STATUS of the LISTENER
------------------------
Alias                     lsnrnn
Version                   TNSLSNR for Solaris: Version 8.1.5.0.0
- Production
Start Date                15-JUL-99 13:17:48
Uptime                    0 days 0 hr. 0 min. 0 sec
```

```
Trace Level             off
Security                OFF
SNMP                    OFF
Listener Parameter File   <tns_admin>/listener.ora
Listener Log File         <tns_admin>/log/lsnrnn.log
Services Summary...
  Unn has 1 service handler(s)
The command completed successfully
```

6 View the contents of the listener log file.

a View the contents of the lsnr*nn*.log file in the `$TNS_ADMIN/log` directory.

7 Start up your database instance.

a Enter the following command to start the server:

```
$ sqlplus "/ as sysdba"
SQL*Plus: Release 8.1.5.0.0 - Production on Mon Aug 2 16:06:53
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> startup pfile=$HOME/LABS/initUnn.ora
ORACLE instance started.
Total System Global Area    6315408 bytes
Fixed Size                    64912 bytes
Variable Size               5308416 bytes
Database Buffers             409600 bytes
Redo Buffers                 532480 bytes
Database mounted.
Database opened.
```

Practice 4 Solutions

If you are unsure of the name of your client, please ask the instructor how to obtain the name.

1. Using the local naming method in Net8 Assistant, configure the client side. The protocol used is TCP/IP. The port number for the listener you should connect to is the port number provided for the listener in the previous practice. SID is U*nn*.

   Note: Use the Oracle8*i* Release 8.0 Database SID method to connect instead of the Oracle8*i* Release 8.1 Service Name method.

   a. From the Windows NT menu bar, choose Start—>Programs—>Oracle - *Oracle Home*—>Network Administration—>Net8 Assistant.

   b. From the Net8 Assistant menu, choose Net Service Names, and then select Edit—>Create, or click the "+" icon.

   c. In the Net Service Name Wizard window, enter your alias name for the address that you want to connect to.

   d. Select the network protocol (TCP/IP) and click Next.

   e. Enter the host name (the name of the server that holds your database) and the port number you have been assigned and click Next.

   f. Select Oracle8*i* Release 8.0 or Previous, and enter the system identifier (SID) of the database to which you want to connect in the Service Name field, and click Next.

   g. Select Finish.

   h. You will now return to the Net8 Assistant. Choose File —>Save Network Configuration.

2. Test that the service is operational.

   a. Test by using the test facility from Net8 Assistant or by connecting to the server side using SQL*Plus from the client side.
   Example:

   ```
   c:\> sqlplus system/manager@Unn
   where Unn is your service name
   ```

3. Investigate the contents of the `tnsnames.ora` file.

```
# C:\ORANT\NETWORK\ADMIN\TNSNAMES.ORA Configuration
# File:c:\orant\NETWORK\ADMIN\tnsnames.ora
# Generated by Oracle Net8 Assistant
Unn =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = <server_name>)(PORT =
<port>))
    )
    (CONNECT_DATA =
      (SID = Unn)
    )
  )
```

4 Connect to the server as `system/manager` using SQL*Plus and verify that you are connected to the correct instance by querying the V$INSTANCE view. When everything is working, have the group or person next to you edit your `tnsnames.ora` file in order to provoke an error either by specifying a wrong port to connect to or a wrong node name, database, or service name.
When they are done, try to correct the error by regenerating the `tnsnames.ora` file by hand.

```
c:\> sqlplus system/manager@Unn
SQL*Plus: Release 8.1.5.0.0 - Production on Thu Jul 15 18:34:46
1999
(c) Copyright 1999 Oracle Corporation. All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Objects options
PL/SQL Release 8.1.5.0.0 - Production

SQL> select instance_name, version, startup_time, status
  2  from v$instance;
INSTANCE_NAME    VERSION           STARTUP_T STATUS
---------------- ----------------- --------- -------
DB01            8.1.5.0.0         02-AUG-99 OPEN
```

5 Using Net8 Assistant, configure your client to make connections to the listener on the server using the new service name instead of the instance name.

Hint: Use Oracle8*i* release 8.1–compatible identification.

a From the Windows NT menu bar, choose Start—>Programs—>Oracle - *Oracle Home*—>Network Administration—>Net8 Assistant.

b From the Net8 Assistant menu, double-click Net Service Names and choose the service name you just created.

c In the right window, deselect the Use Oracle8 8.0 Compatible Identification check box.

The Service Name field will be highlighted.

d Enter the database name with the domain name as follows:

U*nn*

e Save the network configuration.

Browse the contents of the `tnsnames.ora` file.

```
# C:\ORANT\NETWORK\ADMIN\TNSNAMES.ORA Configuration
# File:c:\orant\NETWORK\ADMIN\tnsnames.ora
# Generated by Oracle Net8 Assistant
Unn =
  (DESCRIPTION =
    (ADDRESS_LIST =
      (ADDRESS = (PROTOCOL = TCP)(HOST = <server_name>)(PORT =
<port>))
    )
```

```
    (CONNECT_DATA =
      (SERVICE_NAME = Unn)
    )
  )
```

6 Connect to the server as `system/manager` using SQL*Plus and verify that you are connected to the correct instance as was done in step 4.

7 Shut down the database instance and the listener, and configure your database instance (U*nn*) for automatic instance registration.

Hint: Look at the parameters SERVICE_NAMES and INSTANCE_NAME in the `init`*`SID`*`.ora` file.

a Enter the following command to shut down the server:

```
$ sqlplus "/ as sysdba"
SQL*Plus: Release 8.1.5.0.0 - Production on Mon Aug 2 16:06:53
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

b Enter the following command to stop the listener:

```
$ lsnrctl stop lsnrnn
```

c Edit the `init`*`SID`*`.ora` file and find the service name and instance name parameter.

d Enter the following parameters if they have not already been entered by the installation:

```
INSTANCE_NAMES=Unn
SERVICE_NAMES=Unn
```

e Enter the following command to start the server:

```
$ sqlplus "/ as sysdba"
SQL*Plus: Release 8.1.5.0.0 - Production on Mon Aug 2 16:06:53
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> startup pfile=$HOME/LABS/initUnn.ora
ORACLE instance started.
```

```
Total System Global Area    6315408 bytes
Fixed Size                    64912 bytes
Variable Size               5308416 bytes
Database Buffers             409600 bytes
Redo Buffers                 532480 bytes
Database mounted.
Database opened.
```

8 Check that the instance has been registered and which listener has the handle.

a Enter the following command to check the default listener:

```
$ lsnrctl services
LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 12-AUG-
99 13:26:45
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connecting to
(DESCRIPTION=(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS))
(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=1521)))
Services Summary...
Unn has 1 service handler(s)
    DEDICATED SERVER established:0 refused:0
      LOCAL SERVER
Unn has 1 service handler(s)
    DEDICATED SERVER established:0 refused:0
      LOCAL SERVER
The command completed successfully
```

b The default listener listening on port 1521 has the handle to the U*nn* database.

9 Restart the database and have it register with your working listener `lsnr`*nn*. Hint: Look at the parameters LOCAL_LISTENER in the `init`*SID*`.ora` file. Create a new `tnsnames.ora` manually. Do not use the Net8 Assistant.

a Enter the following command to shut down the server:

```
$ sqlplus "/ as sysdba"
SQL*Plus: Release 8.1.5.0.0 - Production on Mon Aug 2 16:06:53
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

b Edit your `listener.ora` file to make sure it does not contain the SID_LIST_LSNR*nn* parameter.

c Enter the following to stop your listener:

```
$ lsnrctl stop lsnrnn
```

d Enter the following command to start your listener:

```
$ lsnrctl start lsnrnn
```

e Enter the following parameter in your `initSID.ora` file:

```
LOCAL_LISTENER=<listener_name>
```

f Create a new `tnsnames.ora` file with the following content:

```
listener_name=
(description=
   (address=
      (protocol=tcp)
      (host=<server_name>)
      (port=<port>)))
```

g Enter the following command to start the server:

```
$ sqlplus "/ as sysdba"
SQL*Plus: Release 8.1.5.0.0 - Production on Mon Aug 2 16:06:53
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> startup pfile=$HOME/LABS/initUnn.ora
ORACLE instance started.
Total System Global Area    6315408 bytes
Fixed Size                    64912 bytes
Variable Size               5308416 bytes
Database Buffers             409600 bytes
Redo Buffers                 532480 bytes
Database mounted.
Database opened.
```

h Check to make sure your instance has registered with your working listener and not the default listener on port 1521.

```
$ lsnrctl status lsnrnn
LSNRCTL for Solaris: Version 8.1.5.0.0 ... 12-AUG-99 14:20:36
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=wwed165-
sun)(PORT=1534))
(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))
```

```
STATUS of the LISTENER
------------------------
Alias                     lsnrnn
Version                   TNSLSNR for Solaris: Version 8.1.5.0.0
- Production
Start Date                12-AUG-99 14:19:05
Uptime                    0 days 0 hr. 1 min. 30 sec
Trace Level               off
Security                  OFF
SNMP                      OFF
Listener Parameter File   /oracle/net8i/listener.ora
Listener Log File         /oracle/net8i/log/lsnrnn.log
Services Summary...
  Unn has 1 service handler(s)
The command completed successfully
```

Practice 6 Solutions

1 Create a Names server on your local PC node and view the contents of the Names configuration file. Do not configure the Names server with a region database. Use 1575 as the port on which the Names server will accept incoming name resolution requests. Make sure the DB_DOMAIN parameter is set to NULL.

Note: In this class, the Names server is created on your local PC node; therefore, your node will act as a client as well as a Names server. Ideally, a Names server should reside on a dedicated middle-tier node on the network.

Bring up the DOS box and enter `hostname` to determine the name of your client PC.

a Click the Oracle Names Servers icon in Net8 Assistant.

b Click the OK button when the message dialog appears.

c Select Create from the Edit menu item, or click the "+" icon.

d This will start up the Names Wizard. Follow the instructions of the Names Wizard closely.

e Click the Next button to continue reading the wizard instructions.

f Enter a name for the Names server <*your_names_server_name*> and click the Next button.

g Enter the host name of your *client* machine, and a port number (1575) assigned for the Names server.
Note: Enter `hostname` at the DOS prompt to determine the name of the client PC.

h Click Next to follow the wizard instructions.

i Select the Don't Use a Region Database option button and click Next.

j Select the Names Server Is First in Its Region option button and click Next.

k Click the Yes option button for the Names server being in the root region and click Next.

l Click the Finish button.

m Click OK when the dialog box appears indicating a successful Names server creation.

n Browse the contents of the `names.ora` file located in the `$ORACLE_HOME\network\admin` directory on the client:

```
# C:\ORANT\NETWORK\ADMIN\NAMES.ORA Configuration
# File:c:\orant\NETWORK\ADMIN\names.ora
# Generated by Oracle Net8 Assistant
NAMES.SERVER_NAME = <names_server_name>
NAMES.ADDRESSES =
  (ADDRESS = (PROTOCOL = TCP)(HOST = <client_machine>)(PORT =
1575))
```

o Save the network configuration.

p Select Discover Oracle Names Servers from the Command menu item. This will take some time. Click Cancel when the dialog box appears indicating that no Names server was discovered.

q Click OK at the dialog box.

2 Start up the Names server, and add a service name for the database U*nn* to the Names server. Use the attributes for the database service name defined in your existing `tnsnames.ora` file.

Note: Do not use Oracle8*i* release 8.0–compatible identification to connect

a Select Manage Server from the pull-down menu.

b Select the Start button if it has not already been selected.

c Click the Apply button.

d This should start the Names server process. Click OK at the dialog box.

e Select Manage Data from the pull-down menu.

f Click the Add button.

g Enter a service name to this Names server by entering the host name of the service and the port number the listener is on. Get this information from your existing `tnsnames.ora` file.

h Verify that the SID, Protocol, and Host Name values are correct.

i Deselect the Use Oracle8*i* Release 8.0 Compatible Identification check box.

j Enter the service name (U*nn*).

k Click the Execute button.

The bottom left corner should display a status of Operation complete.

3 Using Net8 Assistant, configure your client profile to use Oracle Names, and assign the previously created Names server as the preferred Names server for your client profile.

a Start up Net8 Assistant.

b Click the Profile folder icon.

c Select Naming from the pull-down menu, if not already selected.

d Click the Methods tab.

e Add ONAMES to the Selected Methods, if it has not already been selected.

f Select Preferred Oracle Names Servers from the pull-down menu.

g Click the New button.

h Select TCP as the protocol.

i Enter the host name of the preferred Names server (Your client machine’s host name in this case).

j Enter the port number at which the Names server is listening for incoming requests. You can get this from the existing `names.ora` file.

k Save the network configuration from the File menu item.

4. Connect to U*nn* using SQL*Plus and verify if the Names server is operational by using ONAMES as the only naming method.
   a. Start up SQL*Plus and connect to U*nn* using the existing `tnsnames.ora` file.
   b. Remove all other naming methods except ONAMES.
   c. Save the network configuration.
   d. Start up SQL*Plus and connect to U*nn* again.
5. Using the Windows Control panel, shut down the Names server. After the Names server is shut down, perform the following functions using the Names Control utility:

   Double-click the Services icon from the Windows Control Panel, and stop the Oracle<*Oracle Home*>Names<*names_server_name*> service.

   a. View the status of the Names server by using the Names Control Utility.

      At the DOS prompt, enter `namesctl` to use the Names Control utility, and enter the following to view the status of the Names server:

      ```
      NAMECTL> status
      ```

   b. What is the error message, and why does it occur?

   The following error message occurs:

      NNL-00005: no server has been set. Use the “SET SERVER” command first

      This error occurs because the Names Control utility does not set the Names server. It has to be set manually using the SET SERVER command.

   c. Set the server to the Names server that you previously configured.

      Enter the following to set the server to the current Names server:

      ```
      NAMECTL> set server
      (address=(protocol=tcp)(host=<server>)(port=<port>))
      ```

      Note: This address can be taken from the `names.ora` file.

   d. What is the error message, and why does it occur?

      ```
      NNC-00003: error opening stream
      (address=(protocol=tcp)(host=<server>)(port=<port>))
      ```

      This error occurs because the Names server has not been started. The SET SERVER command and the STATUS command both require that the Names server be running.

   e. Start up the Names server.

      Issue the following command to start up the Names server:

      ```
      NAMECTL> start
      (address=(protocol=tcp)(host=<server>)(port=<port>))
      ```

      or

      ```
      NAMECTL> start
      ```

6 If you do not plan to attempt the optional practice, configure your profile as before, so that it uses the TNSNAMES naming method rather than ONAMES.
   a Click the Profile icon and select Naming from the pull-down menu.
   b Click the Methods tab and select TNSNAMES only.
   c Save the network configuration.

Optional Practice

7 Ensure that the Names server is running. Restart the Net8 Assistant utility, then reconfigure your Names server to use a region database. The region database should now reside on the U*nn* database on the server, while the Names server still resides on the client. Use the SYSTEM user to run the Names server initialization script (Typically you would use a separate user for this purpose).

   Note: Do not use Oracle8*i* Release 8.0–compatible identification when connecting.

   a Use SQL*Plus to connect to the U*nn* server as SYSTEM, using the previously configured Names server.
   b Run the Names server initialization file `$ORACLE_HOME\network\names\namesini.sql` on U*nn*.

      ```
      SQL> @$ORACLE_HOME\network\names\namesini.sql
      ```

   c Restart the Net8 Assistant utility.
   d Double-click the Oracle Names Servers icon.
   e Click the previously configured Names server listed under the Oracle Names Servers icon.
   f From Net8 Assistant, select Configure Server from the pull-down menu for the current Names server.
   g Click the Database tab.
   h Select the Region Database option button.
   i Enter the host name and port number of the U*nn* server, which will maintain the region database.

      Note: The port number should be that at which the listener of the server is listening at for incoming connections, usually port 1521.

   j Deselect the Use Oracle8*i* Release 8.0–Compatible Identification check box.
   k Enter the U*nn* as the Service Name of the database that will store the region information.
   l Enter the username and password into the new region database. (Use the SYSTEM username and password provided by your instructor.)
   m Save the network configuration.
   n Restart the Names server.

8 Add a service name for U*nn* to the newly created region database on U*nn* itself.
Note: Use a different service name with the same connection attributes.

- **a** Select Manage Data from the pull-down menu.
- **b** Click the Add button.
- **c** Enter a different service name with the same host name and port of the U*nn* database in the appropriate fields.
- **d** Verify that the Service Name, Protocol, and Server Type values are correct.
- **e** Click Execute.

9 Attempt to connect to U*nn* using SQL*Plus after the service name is added.

Start up SQL*Plus and connect to U*nn*.

10 Query the NAME_P and ZD_VALUE1_P columns of the ONRS_REGION table to view the newly created values.

From the SQL prompt, enter the following SQL statement to list the connection strings stored in the region database:

```
SQL> col name_p format a15;
SQL> col zd_value1_p format a40;
SQL> select name_p, zd_value1_p from onrs_region;
NAME_P       ZD_VALUE1_P
------------ ----------------------------------------
             (DATA_LIST=(FLAGS=0x11)(DATA=(TYPE=ns.sm
             d.)(NAME=name_svr01.)))

name_svr01   (DATA_LIST=(FLAGS=0x9)(DATA=(TYPE=a.smd.
             )(ADDRESS = (PROTOCOL = TCP)(HOST = hras
             muss-lap)(PORT = 1575))))

name_svr01   (DATA_LIST=(FLAGS=0x9)(DATA=(TYPE=tos.np
             d.omd.)(CTEXT=ORACLE_NAMESERVER)))

name_svr01   (DATA_LIST=(FLAGS=0x9)(DATA=(TYPE=host.n
             m.omd.)(TEXT=hrasmuss-lap)))

hannes_data
hannes_data  (DATA_LIST=(FLAGS=0x1)(DATA=(TYPE=a.smd.
             )(DESCRIPTION=(SOURCE_ROUTE=OFF)(ADDRESS
             _LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=wwed1
             65-sun)(PORT=1534)))(CONNECT_DATA=(SERVI
             CE_NAME=DB01)(SRVR=dedicated)))))
```

11 Configure your profile as before, so that it uses the TNSNAMES naming method rather than ONAMES, and ensure that you can connect using the previously used TNSNAMES naming method.

a Click the Profile icon and select Naming from the pull-down menu.

b Click the Methods tab and select TNSNAMES only.

c Save the network configuration.

d Connect to the U*nn* server using SQL*Plus.

Practice 7 Solutions

Start a Telnet session connecting to the server where your database resides.

1. Configure and start up the multithreaded server for your database so that you have one dispatcher listening for TCP/IP connections and one shared server to serve requests.

   Specify the maximum dispatchers as two and maximum shared servers as six.

   a. Modify the `initU*nn*.ora` to have the following entries:

   ```
   # Multithreaded Server Environment
   mts_dispatchers='(PRO=TCP)(DIS=1)'
   mts_servers = 1
   mts_max_dispatchers = 2
   mts_max_servers = 6
   ```

   b. Start up the database:

   ```
   $ sqlplus "/ as sysdba"
   SQL*Plus: Release 8.1.5.0.0 - Production on Mon Aug 2 16:06:53
   1999
   (c) Copyright 1998 Oracle Corporation.  All rights reserved.
   Connected to:
   Oracle8i Enterprise Edition Release 8.1.5.0.0 - Production
   With the Partitioning and Java options
   PL/SQL Release 8.1.5.0.0 - Production
   SQL> startup pfile=$HOME/LABS/initUnn.ora
   ORACLE instance started.
   Total System Global Area    6315408 bytes
   Fixed Size                    64912 bytes
   Variable Size               5308416 bytes
   Database Buffers             409600 bytes
   Redo Buffers                 532480 bytes
   Database mounted.
   Database opened.
   ```

2. To verify that a dispatcher is associated with your listener use the LSNRCTL utility, and issue the command: lsnrctl SERVICES *your_listener_name*.

   ```
   $ lsnrctl services lsnrnn
   LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 16-JUL-99
   17:24:35
   (c) Copyright 1998 Oracle Corporation.  All rights reserved.
   ```

```
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=<por
t>))
(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))
Services Summary...
Unn has 3 service handler(s)
    DEDICATED SERVER established:8 refused:0
      LOCAL SERVER
    DEDICATED SERVER established:0 refused:0
      LOCAL SERVER
    DISPATCHER established:0 refused:0 current:0 max:254
state:ready
      D000 <machine: <server_name>, pid: 2983>
      (ADDRESS=(PROTOCOL=tcp)(HOST=<server_name>)(PORT=53064))
The command completed successfully
```

3 Before making a connection, query the view V$CIRCUIT from SQL*Plus connecting as `system/manager` in your Telnet session to see if it contains data. This view has an entry for each connection session currently using the MTS.

```
$ sqlplus system/manager
SQL*Plus: Release 8.1.5.0.0 - Production on Fri Jul 16 17:28:02
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
Oracle8i Enterprise Edition Release 8.1.5.0.0 -Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> select circuit, status from v$circuit;
no rows selected
```

4 Make a connection using SQL*Plus, connecting as system/manager from your client to the server, and check V$CIRCUIT view again.

After you have verified the connection, exit SQL*Plus.

a Connect from SQL*Plus.

```
c:\> sqlplus system/manager@Unn
```

b Check that you have a connection using MTS.

```
SQL> select circuit, status from v$circuit;
   CIRCUIT  STATUS
   -------- ----------------
   801E58D0 NORMAL
```

The entry in V$CIRCUIT indicates that you have a connection using MTS.

5 Do the following:

a Query the V$SHARED_SERVER view to see how many shared servers have been started. You will see that just one shared server has been started.

```
SQL> select name, status, circuit from v$shared_server;

NAME STATUS          CIRCUIT
---- --------------- --------
S000 EXEC            801E58D0
```

b Query the V$DISPATCHER view to see how many dispatchers have been started. You will see that one dispatcher for TCP has been started.

```
SQL> select name, status from v$dispatcher;
NAME STATUS
---- ----------------
D000 WAIT
```

6 Make two connections using SQL*Plus, connecting as `system/manager` from your client to the server using MTS.

a Has the number of shared servers increased?

```
SQL> select name, status, circuit from v$shared_server;
NAME STATUS          CIRCUIT
---- --------------- --------
S000 EXEC            801E58D0
The number of shared servers has not increased.
```

b Why or why not?

The number of shared servers has not increased because they are shared among several connections established through the dispatcher.

7 How could the number of shared servers increase?

The only way that the number of shared servers will increase is if multiple requests must be handled at the same time and there are not enough shared servers to handle the requests.

Shared servers can be allocated dynamically up to the number specified by MTS_MAX_SERVERS.

Exit your SQL*Plus connections.

8 Do the following:

a Add one more dispatchers to handle TCP requests.

```
SQL> alter system set mts_dispatchers='(PRO=TCP)(DIS=2)';
System altered.
```

b Verify that an additional dispatcher has been added.

```
SQL> select name, status from v$dispatcher;

NAME STATUS
---- ----------------
D000 WAIT
D001 WAIT
```

c Decide whether you can add a third dispatcher.

No, because the `initU`*nn*`.ora` parameter MTS_MAX_DISPATCHERS is set to 2.

9 Shut down your database

a Configure connection pooling so that the pooling takes effect when a third connection is established.

In your `initU`*nn*`.ora` file specify the following:

```
mts_dispatchers=
"(PRO=TCP)(DIS=1)(POO=ON)(CON=1)(SESS=3)(TIC=6)"
mts_servers = 1
```

Note that SESS is set to 3 because of a bug that computes CON to CON+1. If SESS is set to 2 and CON also is equal to 2, then the third connection would be established via a dedicated server.

b Verify that the connection pooling feature is working.

Start up the database and start three SQL*Plus connections from your client to your server. You will see that the third connection will hang for up to 60 seconds before being logged on.

Practice 8 Solutions

Because of setup restrictions in the classrooms, you will be simulating the middle tier by using your local machine. In a real-world environment, it is recommended to use a multitier model.

Before starting this practice, make sure that you have no connections established to the server side.

Start a Telnet session and connect to your server. Log in and check the usage of sockets/ports by issuing the command `netstat -a | grep` *your_client_name* `| grep ESTABLISHED` where *your_client_name* is the host name of your PC.

You should see something like:

```
netstat -a | grep wwedf-162 | grep ESTABLISHED
wwedf-162.us.oracle.com.v7dist33 8533 0 8760 0 ESTABLISHED
```

The output line will show the established connections from your client using the TCP protocol.

This indicates the socket used by your Telnet session. If you get more than one line, you have more than one connection established to the server.

In this lab, you will look at the socket usage when using the multiplexing feature of Connection Manager.

1 On your local machine configure and start Connection Manager.

a `cman.ora` file:

```
# Connection Manager config file cman.ora
# cman's listening addresses
cman = (ADDRESS_LIST=
(ADDRESS=(PROTOCOL=tcp)(HOST=<client_name>)(PORT=1610)))
```

b From a DOS prompt:

```
c:\> cmctl
CMCTL for 32-bit Windows: Version 8.1.5.0.0 - Production on 16-
JUL-99 11:39:48
(c) Copyright 1998 Oracle Corporation. All rights reserved.
CMCTL> start cm
Service Oracle<Oracle Home>Cman start pending.
Service Oracle<Oracle Home>Cman started.
CMAN Status:
(STATUS=(VERSION=8.1.5.0.0)(STARTED=08-JUL-99
12:13:04)(STATE=running))
CMCTL> exit
```

2 Edit your `tnsnames.ora` file, and add a service name entry that will use Connection Manager to connect to the database on your server.

`tnsnames.ora` file:

```
Unn =
(DESCRIPTION=(address_list=
(ADDRESS=
(PROTOCOL=tcp)(PORT=1610)(HOST=<client_name>))
(ADDRESS=
(PROTOCOL=tcp)(PORT=<port>)(HOST=<server_name>)))
(CONNECT_DATA=(SERVICE_NAME=Unn))
(SOURCE_ROUTE=no)
)
```

3 Test your service name by establishing two connections to the server using SQL*Plus.

a From your Telnet session, check how many sockets you are now using.

```
/dbaclass7/dba20> netstat -a | grep <your_client> | grep
ESTABLISHED
<your_client>.DBA09_LI  7793      0  8760      0 ESTABLISHED
<your_client>.DBA20_LI <your_client>.1850  7549     0  8760
0 ESTABLISHED
<your_client>.DBA20_LI <your_client>.1853  7549     0  8760
0 ESTABLISHED
```

b What explains the number of sockets used?
There are now three sockets in use: one for the Telnet session and one for each of the SQL*Plus network sessions. Note that you have not yet configured for the multiplexing on the server side.

4 Exit your SQL*Plus sessions and verify that you just have one socket where a connection is established.

```
/dbaclass7/dba20> netstat -a | grep wwedf-162 | grep ESTABLISHED
wwedf-162.us.oracle.com.DBA09_LI  7793      0  8760      0
ESTABLISHED
```

5 From your Telnet session, shut down your database on the server and configure for multiplexing.

```
$ sqlplus system/manager
SQL*Plus: Release 8.1.5.0.0 - Production on Fri Jul 16 17:28:02
1999
(c) Copyright 1998 Oracle Corporation.  All rights reserved.
Connected to:
```

```
Oracle8i Enterprise Edition Release 8.1.5.0.0 -Production
With the Partitioning and Java options
PL/SQL Release 8.1.5.0.0 - Production
SQL> shutdown immediate;
Database closed.
Database dismounted.
ORACLE instance shut down.
```

Modify the `initU`*`nn`*`.ora` file to have the following entries:

```
# Multithreaded Server Environment
mts_dispatchers='(PRO=TCP)(MUL=ON)(DIS=1)'
mts_servers = 1
```

6 Start up your database again and make two SQL*Plus connections from your client.

a Check how many sockets you use now.

```
/dbaclass7/dba20> netstat -a | grep <client_name> | grep
ESTABLISHED
<client_name>.us.oracle.com.DBA09_LI  7793     0  8760     0
ESTABLISHED
<server_name>.DBA20_LI <client_name>.us.oracle.com.1850 7549
0  8760     0 ESTABLISHED
```

b What explains the number of sockets used?
There are now just two sockets in use: one for the Telnet session and one for the two SQL*Plus network sessions. There are two sockets in use because of the multiplexing feature that now is enabled.

c Close all the SQL*Plus sessions.

7 Do the following:

a Configure the Connection Manager so that your PC will not be restricted from accessing any destination.

Add the entry in your `cman.ora` configuration file.

```
cman_rules = (rule_list=
(rule=(src=<client_name>)(dst=x)(srv=x)(act=REJ)))
Stop and start the Connection Manager again.
c:\> cmctl
CMCTL for 32-bit Windows: Version 8.1.5.0.0 - Production on 16-
JUL-99 11:39:48
(c) Copyright 1998 Oracle Corporation. All rights reserved.
CMCTL> stop cm
CMCTL> start cm
Service Oracle<Oracle Home>Cman start pending.
Service Oracle<Oracle Home>Cman started.
CMAN Status:
(STATUS=(VERSION=8.1.5.0.0)(STARTED=16-JUL-99
12:13:04)(STATE=running))
```

```
CMCTL> exit
```

b Describe what happens when you try to connect.

```
SQL*Plus: Release 8.1.5.0.0 - Production on Fri Jul 16 11:11:35
1998
(c) Copyright 1999 Oracle Corporation. All rights reserved.
ERROR:
ORA-12204: TNS:received data refused from an application
```

8 Stop the Connection Manager, shut down the database server, and disable the multithreaded server.

Practice 9 Solutions

Before starting this practice, the instructor will provide you with a trace directory and a log directory name.

1. Enable logging for your client. Specify the log directory and verify that the `sqlnet.ora` file is correct.

   You can either use Net8 Assistant or edit your `sqlnet.ora` file by hand.

   `sqlnet.ora`:

   ```
   # C:\ORANT8\NETWORK\ADMIN\SQLNET.ORA Configuration
   #  File:C:\ORANT8\NETWORK\ADMIN\sqlnet.ora
   # Generated by Oracle Net8 Assistant
   LOG_FILE_CLIENT = net
   LOG_DIRECTORY_CLIENT = c:\orant\network\log
   SQLNET.EXPIRE_TIME = 0
   NAMES.DIRECTORY_PATH = (TNSNAMES)
   ```

   Note: If the Net8 Assistant is used to create the file above, you must simply use `net` as the file name, and Net8 Assistant will append `.log` to the end of the file.

2. Make a connection to the server using SQL*Plus, connecting as `system/manager`. Will a log file be generated?

   No log file is generated, because no errors occurred.

3. Shut down the listener on the server side, then try to connect, and finally investigate the log file.

   a. $ **lsnrctl stop lsnr02**

   ```
   LSNRCTL for Solaris: Version 8.1.5.0.0 - Production on 16-JUL-
   99 19:37:07
   (c) Copyright 1998 Oracle Corporation.  All rights reserved.
   Connecting to
   (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=wwed165-
   sun)(PORT=1521))
   (PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))
   The command completed successfully
   ```

   b. c:> **sqlplus system/manager@U*nn***

   ```
   SQL*Plus: Release 8.1.5.0.0 - Production on Fri Jul 16 20:06:15
   1999
   (c) Copyright 1999 Oracle Corporation.  All rights reserved.
   ERROR:
   ORA-12541: TNS:no listener
   ```

c From `net.log`:

```
************************************************************************
   Fatal NI connect error 12541, connecting to:

   (DESCRIPTION=(ADDRESS_LIST=(ADDRESS=(PROTOCOL=TCP)(HOST=wwed165
   -
   sun)(PORT=1610)))(CONNECT_DATA=(SERVICE_NAME=PROD.world)(CID=(P
   ROGRAM=C:\orant\BIN\SQLPLUSW.EXE)(HOST=SHKHAN-
   LAP)(USER=shkhan))))

     VERSION INFORMATION:
      TNS for 32-bit Windows: Version 8.1.5.0.0 - Production
      Windows NT TCP/IP NT Protocol Adapter for 32-bit Windows:
   Version 8.1.5.0.0 - Production
     Time: 16-JUL-99 20:15:07
     Tracing not turned on.
     Tns error struct:
       nr err code: 0
       ns main err code: 12541
       TNS-12541: TNS:no listener
       ns secondary err code: 12560
       nt main err code: 511
       TNS-00511: No listener
       nt secondary err code: 61
       nt OS err code: 0
```

4 Start your listener again.

```
$ lsnrctl start lsrnn
Starting /oracle/product/bin/tnslsnr: please wait...
TNSLSNR for Solaris: Version 8.1.5.0.0 - Production
System parameter file is /oracle/product/network/admin/
listener.ora
Log messages written to /oracle/product/network/log/lsnr02.log
Listening on:
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=<por
t>))
(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))

Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=<server_name>)(PORT=<por
t>))
(PROTOCOL_STACK=(PRESENTATION=TTC)(SESSION=NS)))
```

```
STATUS of the LISTENER
------------------------
Alias                     lsnrnn
Version                   TNSLSNR for Solaris: Version 8.1.5.0.0 -
Production
Start Date                16-JUL-99 19:47:06
Uptime                    0 days 0 hr. 0 min. 0 sec
Trace Level               off
Security                  OFF
SNMP                      OFF
Listener Parameter File   /oracle/product/network/admin/
listener.ora
Listener Log File         /oracle/product/network/log/lsnr02.log
Services Summary...
  Unn    has 1 service handler(s)
The command completed successfully
```

5 Enable tracing at SUPPORT level and make sure that the trace files have unique names. You can do this either by using Net8 Assistant or by editing the `sqlnet.ora` file.

Investigate the `sqlnet.ora` file:

```
TRACE_DIRECTORY_CLIENT = c:\orant\network\trace
TRACE_FILE_CLIENT = net
TRACE_LEVEL_CLIENT = SUPPORT
SQLNET.EXPIRE_TIME = 0
NAMES.DIRECTORY_PATH= (TNSNAMES)
```

6 Make a connection, exit the connection, and investigate the trace file.

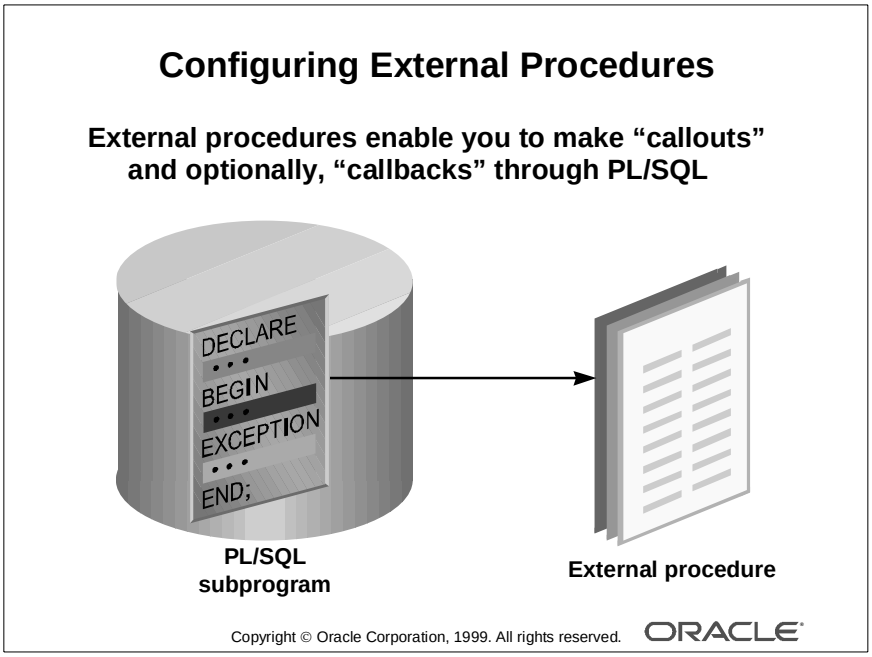
7 Change directories to the assigned trace directory, and try running the Trace Assistant utility using the -o option, send output to the file `tst.txt`, and then investigate the file.

```
C:\ORANT\NETWORK\trace>trcasst -o net.trc > tst.txt
C:\ORANT8\NET80\ADMIN\trace>edit tst.txt
```

C

Configuring External Procedures

Configuring External Procedures



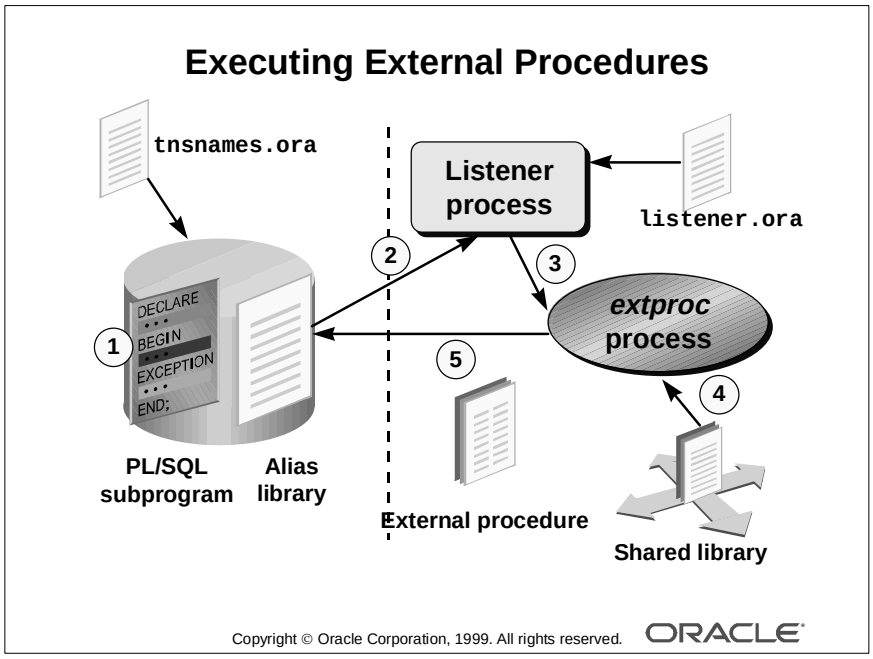
External Procedures

External procedures are configured in the `listener.ora` and `tnsnames.ora` configuration files. An external procedure is a third-generation-language routine stored in a shared library, or a Java class method stored in a lib unit. This makes the strengths and capabilities of those languages available to you. The routine must be called from Java or C, but can be written in any language. The external procedure is registered with PL/SQL so that it can be called from a PL/SQL program.

Typically, external procedures are used to:

- Interface with:
  - Pre-developed application code
  - Third-party routines
  - Embedded systems
- Solve scientific and engineering problems
- Analyze data
- Control real-time devices and processes

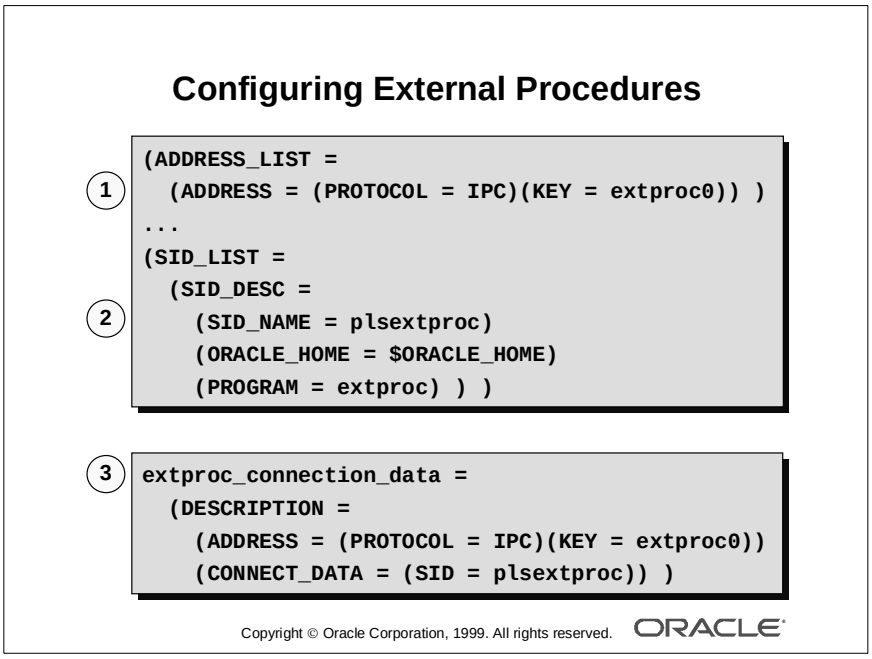
For example, you can use external procedures to send instructions to a robot, solve partial differential equations, process signals, analyze time series, or create animation on a video display.



Executing External Procedures

When the external procedure is invoked, PL/SQL loads the library dynamically and calls the external procedure as if it were a PL/SQL subprogram. A server process executes the PL/SQL subprogram that is registered as an external procedure.

1. The server process looks up the alias library to get the operating system filename.
   - The alias library is created in the Oracle database with the CREATE LIBRARY command.
   - The user must be granted the execute privilege on the library to call the external procedure.
2. The server process passes the request to the listener. The `tnsnames.ora` file includes the information needed by the server to make the connection to the external process.
3. The listener process spawns the extproc process. The `listener.ora` file includes the information needed by the listener to start the process. The extproc process remains active for that user until the user logs off. Because the external procedure runs in a separate address space, it cannot cause damage to the database. For example, it cannot write into the Oracle address space.
4. The extproc process loads the shared library.
5. The extproc process executes the external procedure, which participates fully in the current transaction. The external procedure can call back to the database to do SQL operations and can return results to the PL/SQL subprogram.



External Procedure Configuration

By default, the IPC connection to external procedures is configured automatically during installation. For environments where the external procedure information does not exist, edit the `listener.ora` and `tnsnames.ora` files, as follows:

1. Configure either a TCP/IP or IPC listener address in the `listener.ora` file.
2. Add a system identifier (SID) name of PLSEXTPROC and a program name of EXTPROC in the server's `listener.ora` file.
3. Add a net service name description entry for EXTPROC in the `tnsnames.ora` file, using the following guidelines:
   - The entry must be identified as `extproc_connection_data`.
   - The KEY parameter in the `tnsnames.ora` and `listener.ora` files must match.
   - In the CONNECT_DATA section, the SID parameter is used rather than SERVICE_NAME.
   - The SID parameter in the `tnsnames.ora` must match the SID_NAME parameter coded in the `listener.ora` file.