Oracle® Database

Net Services Administrator's Guide

11*g* Release 2 (11.2)

E10836-01

August 2009

ORACLE®

Oracle Database Net Services Administrator's Guide, 11*g* Release 2 (11.2)

E10836-01

Contributors: Robert Achacoso, Lance Ashdown, Matt Cassady, Abhishek Dadhich, Santanu Datta, Steve Ding, Caroline Johnston, Feroz Khan, Bhaskar Mathur, Scot McKinley, Ed Miner, Sweta Mogra, Srinivas Pamu, Kant Patel, Murali Purayathu, Karthik Rajan, Saravanakumar Ramasubramanian, Kevin Reardon, Sudeep Reguna, James Spiller, Richard Strohm, Norman Woo

Contents

2 Identifying and Accessing the Database

3 Managing Network Address Information

4 Understanding the Communication Layers

5 Understanding Oracle Net Architecture

Part II Configuration and Administration of Oracle Net Services

6 Quick Start to Oracle Net Services

7 Managing Oracle Net Services

8 Configuring Naming Methods

9 Configuring and Administering Oracle Net Listener

10 Configuring Oracle Connection Manager

11 Configuring Dispatchers

12 Configuring Profiles

13 Enabling Advanced Features of Oracle Net Services

14 Optimizing Performance

Part III Testing and Troubleshooting Oracle Net Services

15 Testing Connections

16 Troubleshooting Oracle Net Services

List of Tables

List of Examples

List of Figures

Preface

Oracle Database Net Services Administrator's Guide describes how to use Oracle Net Services. This guide describes the Oracle Net Services product and its components, as well as Oracle Net Services administrative and deployment topics. This preface contains the following topics:

Audience

Oracle Database Net Services Administrator's Guide is intended for the following readers:

- Network administrators
- Directory server administrators
- Database administrators
- Decision makers

This guide is especially targeted for network administrators who are responsible for ensuring connectivity. For network administrators, Oracle recommends:

- For a conceptual understanding of Oracle Net Services, read all of Part I, "Understanding Oracle Net Services"
- For essential configuration instructions, read all of Part II, "Configuration and Administration of Oracle Net Services"
- For troubleshooting, read Part III, "Testing and Troubleshooting Oracle Net Services"

For directory administrators, Oracle recommends:

- For understanding how Oracle Net Services uses a directory server, read Chapter 3, "Managing Network Address Information" in Part I
- For instructions about configuring naming information in a directory server, and exporting existing naming data to a directory server, read Chapter 8, "Configuring Naming Methods" in Part II

For database administrators, Oracle recommends:

- For a general understanding of networking, read Chapter 1, "Introducing Oracle Net Services" and Chapter 6, "Quick Start to Oracle Net Services"
- For an overview of communication layers, read Chapter 4, "Understanding the Communication Layers"
- For understanding how to configure Oracle database server features that require listener and shared server configuration, read Chapter 9, "Configuring and Administering Oracle Net Listener", Chapter 11, "Configuring Dispatchers", and Chapter 14, "Optimizing Performance"

For decision makers, Oracle recommends

- For an understanding of how Oracle Net Services fits into the overall network architecture and for explaining the basics of Oracle Net Services, read Chapter 1, "Introducing Oracle Net Services", Chapter 3, "Managing Network Address Information", and Chapter 6, "Quick Start to Oracle Net Services"

Oracle recommends that all readers look over Part I, to ensure that they have the background required to benefit from the rest of the guide.

Documentation Accessibility

Our goal is to make Oracle products, services, and supporting documentation accessible to all users, including users that are disabled. To that end, our documentation includes features that make information available to users of assistive technology. This documentation is available in HTML format, and contains markup to facilitate access by the disabled community. Accessibility standards will continue to evolve over time, and Oracle is actively engaged with other market-leading technology vendors to address technical obstacles so that our documentation can be accessible to all of our customers. For more information, visit the Oracle Accessibility Program Web site at `http://www.oracle.com/accessibility/`.

Accessibility of Code Examples in Documentation

Screen readers may not always correctly read the code examples in this document. The conventions for writing code require that closing braces should appear on an otherwise empty line; however, some screen readers may not always read a line of text that consists solely of a bracket or brace.

Accessibility of Links to External Web Sites in Documentation

This documentation may contain links to Web sites of other companies or organizations that Oracle does not own or control. Oracle neither evaluates nor makes any representations regarding the accessibility of these Web sites.

Deaf/Hard of Hearing Access to Oracle Support Services

To reach Oracle Support Services, use a telecommunications relay service (TRS) to call Oracle Support at 1.800.223.1711. An Oracle Support Services engineer will handle technical issues and provide customer support according to the Oracle service request process. Information about TRS is available at `http://www.fcc.gov/cgb/consumerfacts/trs.html`, and a list of phone numbers is available at `http://www.fcc.gov/cgb/dro/trsphonebk.html`.

Related Documentation

For additional information, see the following Oracle resources:

- *Oracle Database Net Services Reference*
- Oracle Database 11*g* documentation set

Many books in the documentation set use the sample schemas of the seed database, which is installed by default when you install Oracle Database. Refer to *Oracle Database Sample Schemas* for information about how these schemas were created and how you can use them yourself.

Printed documentation is available for sale in the Oracle Store at

`http://oraclestore.oracle.com`

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

`http://www.oracle.com/technology/membership`

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

`http://www.oracle.com/technology/documentation`

For additional information about the **OSI**, see:

`http://www.ietf.org`

Oracle error message documentation is only available in HTML. If you only have access to the Oracle Documentation CD, then you can browse the error messages by range. After you find the specific range, use your browser's "find in page" feature to locate the specific message. When connected to the Internet, you can search for a specific error message using the error message search feature of the Oracle online documentation.

Conventions

The examples for directories in the book are for Linux. Unless otherwise noted, Microsoft Windows directory paths are the same except that they use a backslash (\) instead of the forward slash (/).

The following text conventions are used in this document:

Convention	Meaning
boldface	Boldface type indicates graphical user interface elements associated with an action, or terms defined in text or the glossary.
italic	Italic type indicates book titles, emphasis, or placeholder variables for which you supply particular values.
`monospace`	Monospace type indicates commands within a paragraph, URLs, code in examples, text that appears on the screen, or text that you enter.

What's New in Oracle Net Services?

This preface describes the new networking features of Oracle Database 11*g* and provides pointers to additional information.

Oracle Database 11*g* Release 2 New Features in Oracle Net Services

The new features for Oracle Net Services in Oracle Database 11*g* Release 2 include:

- Internet Protocol Version 6 (IPv6) Addressing and Connectivity

  This feature includes the following enhancements:

  - IPv6 support for a single-instance database environment
  - Session layer abstraction to support listening across all IPv4 and IPv6 interfaces

    See Also:

    - "TCP/IP Protocol" on page 4-6 to learn about IPv6
    - "About IPv6 Addresses in Connect Descriptors" on page 2-6
    - "Configuring Oracle Connection Manager as a Bridge for IPv4 and IPv6" on page 10-7

- Oracle Restart

  Oracle Restart enhances the availability of Oracle databases in a single-instance environment by restarting the database, the listener, and other Oracle components after a hardware or software failure or whenever your database host computer restarts. The components are started in the proper order, taking into consideration the dependencies among components.

  See Also: "About Oracle Restart" on page 5-3

- Support for `TRANSPORT_CONNECT_TIMEOUT` and `CONNECT_TIMEOUT` at the description level in the connect string. These timeouts apply to each IP address that resolves to a host name.

  - `TRANSPORT_CONNECT_TIMEOUT` specifies the time, in seconds, for a client to establish a TCP connection to the database server. The default value is 60 seconds.
  - `CONNECT_TIMEOUT` specifies the time, in seconds, for a client to establish an Oracle Net connection to the database instance. This parameter overrides the `SQLNET.OUTBOUND_CONNECT_TIMEOUT` parameter.

- CIDR and wildcard support for valid node checking.

  The valid node checking list can include CIDR notation for IPv4 and IPv6 addresses. Wildcard format (*) is supported for IPv4 addresses.

Oracle Database 11*g* Release 1 New Features in Oracle Net Services

The new features for Oracle Net Services in Oracle Database 11*g* Release 1 include:

- Enhanced Network Administration Security
  - Non-Anonymous LDAP Access for Net Naming

    Database administrators can now restrict access to a service by associating an access control list (ACL) with it. This feature can be used with installations that require an extremely high level of security.

    See "Configuring the Directory Naming Method" on page 8-12 and "Configuring Database Access Control" on page 12-4.
- Performance Improvements
  - New Oracle Net fastpath for the common usage scenarios significantly improves Oracle Net performance and is enabled by default in Oracle Database 11*g*. Users do not need to perform any configuration for this feature.
  - Efficient network support for bulk data transfers, such as SecureFiles LOBs. This feature eases the notion of session data unit (SDU) and optimizes large data transfers over an Oracle Net connection by using new paradigms. See "Configuring Session Data Unit" on page 14-1.
  - PHP Scalability

    This feature adds Oracle Net support for efficient event-dispatch mechanisms on platforms that support them. This is internally enabled for PHP usage scenarios and users do not need to perform any configuration for enabling this feature.
- Fast Reconnects for High Availability (HA)
  - This feature adds a mechanism for efficient detection of terminated nodes and connect time failover. Configurable timeouts have been implemented at various levels.
- Support for Database Resident Connection Pooling
  - This feature enables you to share connections or sessions between multiple middle-tier processes

    See Also:

    Oracle Database Concepts for additional information about database resident connection pooling

    "About Database Resident Connection Pooling" on page 2-11
- Enhancements to the Easy Connect Naming Method

  For TCP/IP environments, you can simplify client configuration by using the Easy Connect naming method. The Easy Connect naming method simplifies network management by allowing clients to connect to Oracle Database services without first configuring net service names. Instead, clients make connections with the host name and optional port and service name of the database. See "Using the Easy Connect Naming Method" on page 8-1.

Part I

Understanding Oracle Net Services

Part I provides an overview of Oracle Net Services concepts, products, and tools.

This part contains the following chapters:

1

Introducing Oracle Net Services

This chapter describes the basic elements of Oracle Net Services architecture and the Oracle Net foundation layer.

This chapter contains the following topics:

- About Oracle Net Services
- Oracle Net Services Components

About Oracle Net Services

Oracle Net Services provides enterprise-wide connectivity solutions in distributed, heterogeneous computing environments. Oracle Net Services eases the complexities of network configuration and management, maximizes performance, and improves network diagnostic capabilities.

This section introduces the basic networking concepts involved in a typical network configuration. This section contains the following topics:

- Understanding Connectivity
- Understanding Manageability
- Understanding Shared Server Architecture
- Understanding Performance
- Understanding Network Security

Understanding Connectivity

Oracle Net, a component of Oracle Net Services, enables a network session from a client application to an Oracle Database server. When a network session is established, Oracle Net acts as the data courier for both the client application and the database. It is responsible for establishing and maintaining the connection between the client application and database, as well as exchanging messages between them. Oracle Net is able to perform these jobs because it is located on each computer in the network.

This section contains the following connectivity topics:

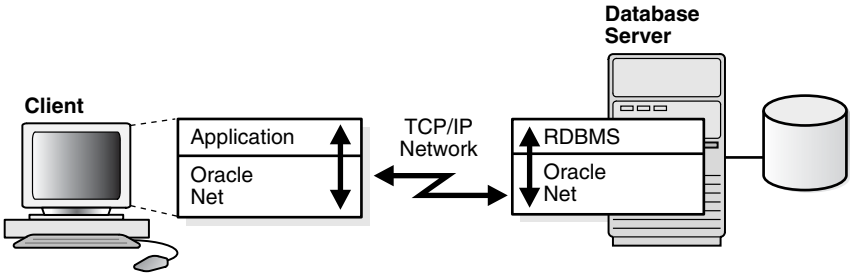
- Client/Server Application Connections
- Web Client Application Connections

Client/Server Application Connections

Oracle Net enables connections from traditional client/server applications to Oracle Database servers. Figure 1–1 shows how Oracle Net enables a network connection

between a client and a database server. Oracle Net is a software component that resides on both the client and the database server. Oracle Net is layered on top of network **Oracle protocol support**, rules that determine how applications access the network and how data is subdivided into packets for transmission across the network. In Figure 1–1, Oracle Net communicates with **TCP/IP** to enable computer-level connectivity and data transfer between the client and the database.

Figure 1–1 Client/Server Application Connection



Specifically, Oracle Net is comprised of the **Oracle Net foundation layer**, which establishes and maintains connections, and **Oracle protocol support**, which maps the foundation layer technology to industry-standard protocols.

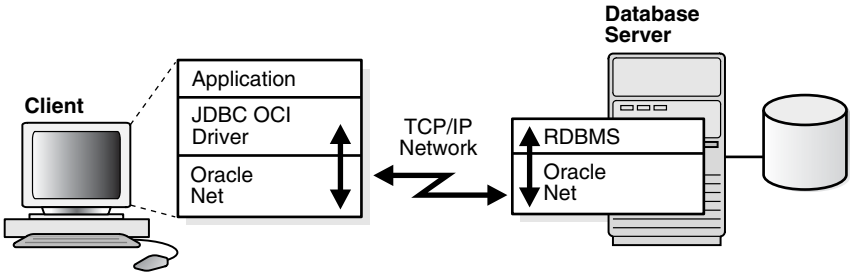
Java Client Application Connections Java client applications access an Oracle Database through a **Java Database Connectivity (JDBC) Driver**, a standard Java interface for connecting from Java to a relational database. Oracle offers the following drivers:

- **JDBC OCI Driver** for client side use with an Oracle client installation
- **JDBC Thin Driver**, a pure Java driver for client side use without an Oracle installation, particularly with applets

These drivers use Oracle Net to enable connectivity between a client application and an Oracle Database.

Figure 1–2 shows a Java client application using a JDBC OCI driver and an Oracle Database server. The Java client application makes calls to the JDBC OCI driver, which translates the JDBC calls directly into the Oracle Net layer. The client then uses Oracle Net to communicate with the Oracle Database that is also configured with Oracle Net.

Figure 1–2 Java Application Connection



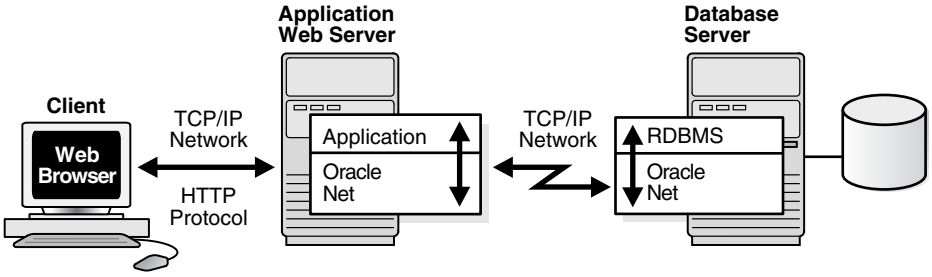
See Also: *Oracle Database JDBC Developer's Guide and Reference*.

Web Client Application Connections

Internet connections from client Web browsers to an Oracle Database server are similar to client/server applications, except that the connection request goes to an application Web server.

Figure 1–3 shows the basic architecture for Web client connections, including a client Web browser, an application Web server, and an Oracle Database server. The browser on the client communicates with **HTTP protocol** to the Web server to make a connection request. The Web server sends the request to an application where it is processed. The application then uses Oracle Net to communicate with the Oracle Database server that also is configured with Oracle Net.

Figure 1–3 Web Client Connections through Application Web Server



The basic components have the following characteristics:

- HyperText Transport Protocol (HTTP)

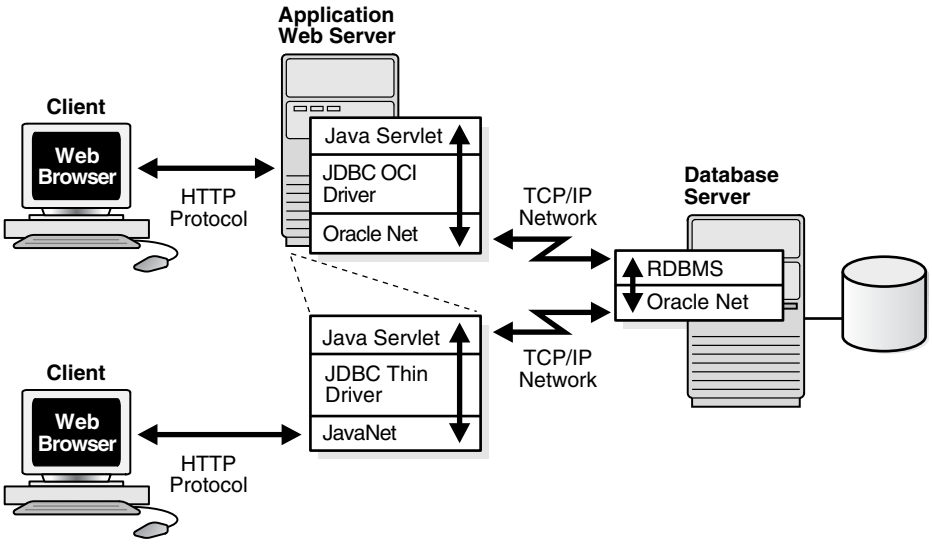
  HTTP provides the language that enables Web browsers and application Web servers to communicate.

- Application Web Server

  An application Web server manages data for a Web site, controls access to that data, and responds to requests from Web browsers. The application on the Web server communicates with the database and performs the job requested by the Web server.

Web Client Connections Through Java Application Web Server An application Web server can host Java applications and servlets, as shown in Figure 1–4. Web browsers make a connection request by communicating through HTTP to an application Web server. The application Web server sends the request to an application or a servlet, which uses a JDBC OCI or a JDBC Thin driver to process the request. The driver then uses Oracle Net to communicate with the Oracle Database server that also is configured with Oracle Net.

Figure 1–4 Web Client Connections Through Java Application Web Server

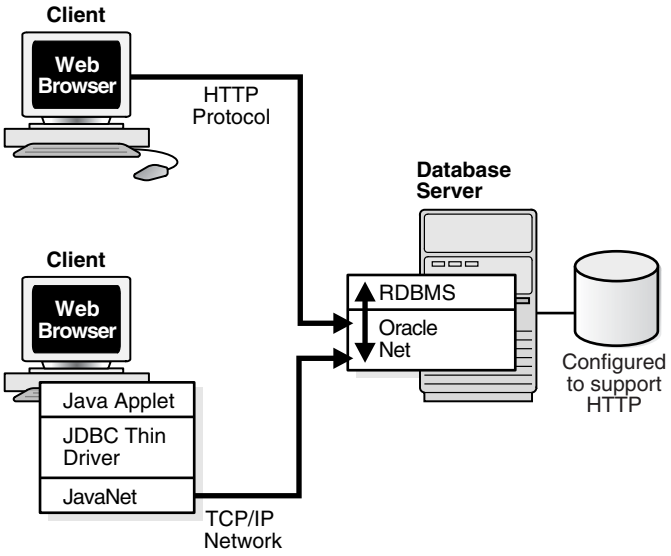


Web Client Connections Without an Application Web Server Web clients that do not require an application Web server to access applications can access the Oracle Database directly, for example, by using a Java applet. In addition to regular connections, the database can be configured to accept HTTP protocol, **FTP protocol**, or **WebDAV protocol** connections. These protocols are used for connections to **Oracle XML DB** in the Oracle Database instance.

Figure 1–5 shows two different Web clients. The first Web client makes an HTTP connection to the database. The second Web client uses a Web browser with a JDBC Thin driver, which in turn uses a Java version of Oracle Net called JavaNet to communicate with the Oracle Database server that is configured with Oracle Net.

See Also: *Oracle XML DB Developer's Guide*

Figure 1–5 Web Client Connection Scenarios



Understanding Manageability

Oracle Net Services offers several manageability features that enable you to easily configure and manage networking components. These features are described in the following topics:

- Location Transparency
- Centralized Configuration and Management
- Quick Installation and Configuration

Location Transparency

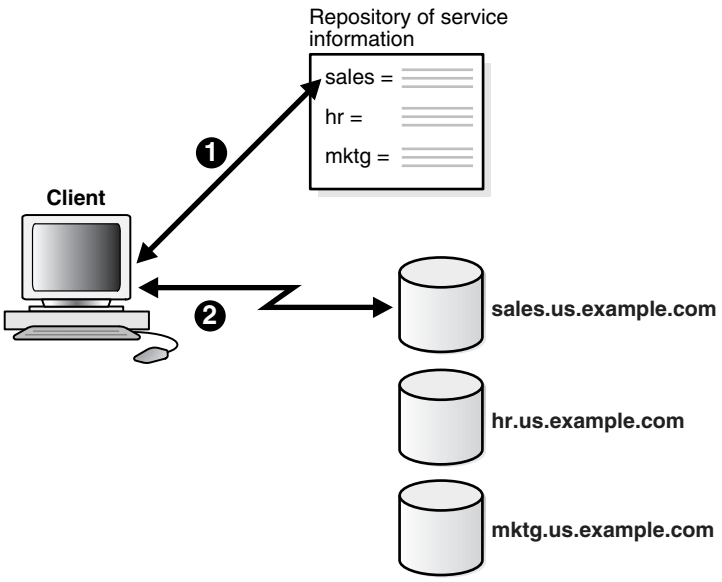
Each database is represented by one or more services. A **service** is identified by a **service name**, for example, `sales.us.example.com`. A client uses a service name to identify the database it must access. The information about the database service and its location in the network is transparent to the client because the information needed for a connection is stored in a repository.

The repository is represented by one or more **naming method**s. A naming method is a resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service. Oracle Net Services offers several types of naming methods that support localized configuration on each client, or centralized configuration that can be accessed by all clients in the network. GUIs enable you to manage data stored in the naming methods.

For example, in Figure 1–6, a company has three databases that clients can access. Each database has a distinct service name, such as `sales.us.example.com`, `hr.us.example.com`, and `mktg.us.example.com`.

1. The client uses the repository to find the information it needs for `sales.us.example.com`.
2. After the client has the information it needs, it connects to the database.

Figure 1–6 Service Information Repository



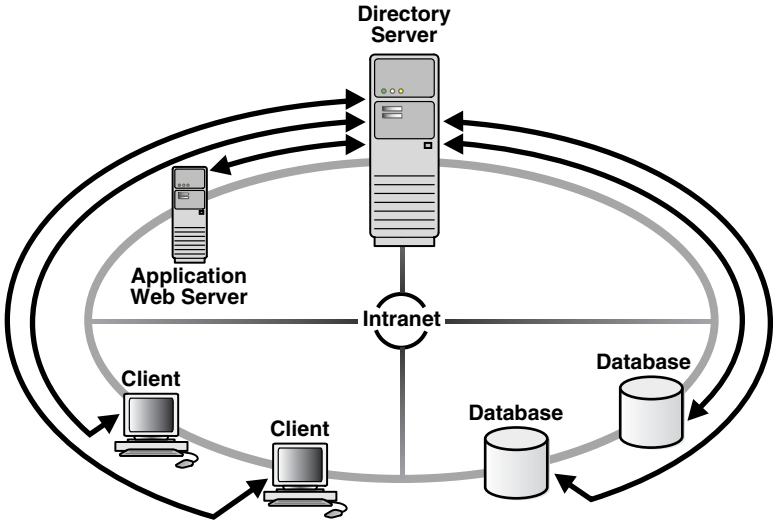
Centralized Configuration and Management

To manage large networking environments, administrators have to be able to easily access a centralized repository to specify and modify the network configuration. For this reason, the Oracle Net Services configuration can be stored in a LDAP-compliant directory server.

Support of LDAP-compliant directory servers provides a centralized vehicle for managing and configuring a distributed Oracle network. The directory can act as a central repository for all information about database network components, user and corporate policies, and user authentication and security, thus replacing client-side and server-side localized configuration files.

All computers on the network can refer to the directory for information. Figure 1–7 shows clients, other servers (such as application Web servers) and Oracle Database servers connecting to a centralized directory server.

Figure 1–7 Centralized Storage of Network Configuration with a Directory Server



See Also: "Using a Directory Server for Centralized Management" on page 3-2 for an in-depth overview of directory server concepts

Quick Installation and Configuration

Networking elements for the Oracle Database server and clients are preconfigured for most environments. The Easy Connect naming method is enabled by default and does not require a repository. Clients connect simply using the hostname of the database. As a result, clients and servers are ready to connect out-of-the-box using Easy Connect, giving users the benefits of distributed computing.

Understanding Shared Server Architecture

The Oracle Database **shared server** architecture increases the scalability of applications and the number of clients that can be simultaneously connected to the database. The shared server architecture also enables existing applications to scale up without making any changes to the application itself.

When using a shared server, clients do not communicate directly with a database **server process**, a database process that handles a client's requests on behalf of a database. Instead, client requests are routed to one or more **dispatcher**s. The dispatchers place the client requests in a common queue. An idle **shared server** from

the shared pool of server processes picks up and processes a request from the queue. This means a small pool of server processes can serve a large number of clients.

Figure 1–8 and Figure 1–9 show the basic difference between the shared server connection model and the traditional **dedicated server** connection model. In the shared server model, a dispatcher can support multiple client connections concurrently. In the dedicated server model, there is one server process for each client. Each time a connection request is received, a server process is started and dedicated to that connection until completed. This causes a processing delay.

Figure 1–8 Shared Server Architecture

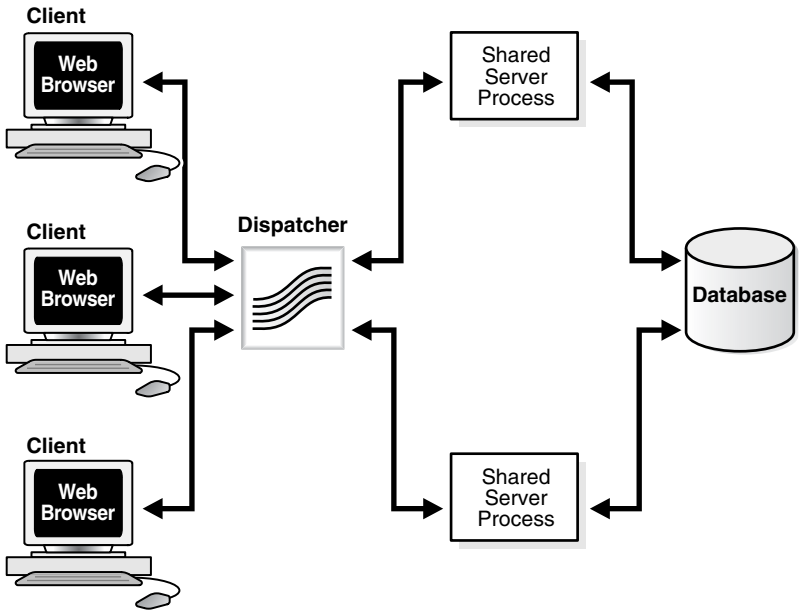
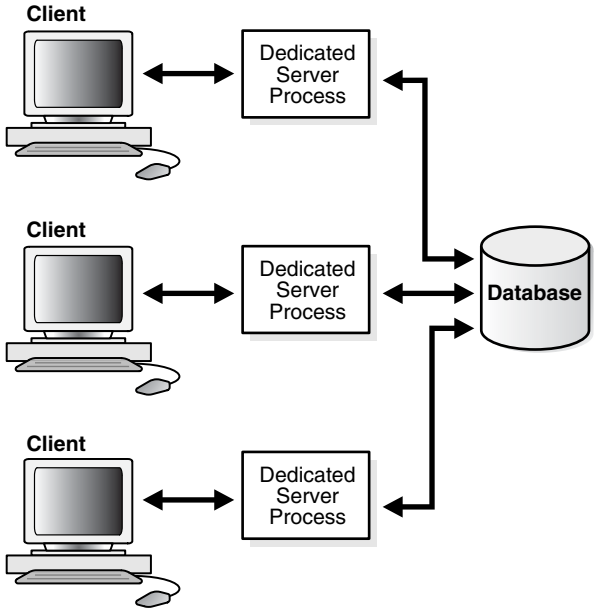


Figure 1–9 Dedicated Server Architecture



A shared server is ideal for configurations with a large number of connections because it reduces the server memory requirements. A shared server is well suited for both Internet and intranet environments.

Utilization of server resources can be further enhanced with Oracle Net Services features that are configurable through a shared server. These features are discussed in the following sections:

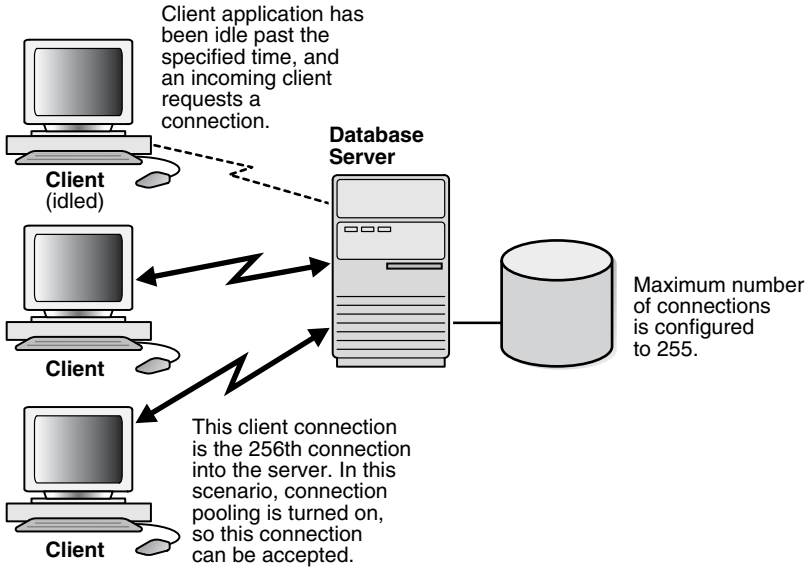
- Connection Pooling
- Session Multiplexing

Connection Pooling

When a large number of clients run interactive Web applications, many of these sessions may be idle at a given time. The **connection pooling** feature enables the database server to timeout an idle session and use the connection to service an active session. The idle logical session remains open, and the physical connection is automatically reestablished when the next request comes from that session. Therefore, Web applications can allow larger numbers of concurrent users to be accommodated with existing hardware.

Figure 1–10 shows how connection pooling works. In this example, the Oracle Database server has been configured with 255 connections. One of the clients has been idle past a specified amount of time. Connection pooling makes this connection available to an incoming client connection, which is connection number 256. When the idle client has more work to do, the connection is reestablished for that client with another client's idle connection.

Figure 1–10 Connection Pooling



Session Multiplexing

Oracle Connection Manager, an Oracle Net Services component, enables multiple client network sessions to be multiplexed, or funneled, through a single network connection to a database.

The **session multiplexing** feature reduces the demand on resources needed to maintain multiple network sessions between two processes by enabling the server to use fewer network connection endpoints for incoming requests. In this way, the total

number of network sessions that a server can handle is increased. One Oracle Connection Manager with multiple gateways enables thousands of concurrent users to connect to a server.

Figure 1–11 shows how session multiplexing can be used in a Web architecture. When Oracle Connection Manager is run on the same computer as an application Web server, the application Web server can route multiple client sessions through Oracle Connection Manager to ensure that those sessions have continuous access to an Oracle Database server. This functionality is especially useful for Web applications where session availability and response time are major concerns.

Figure 1–11 Session Multiplexing

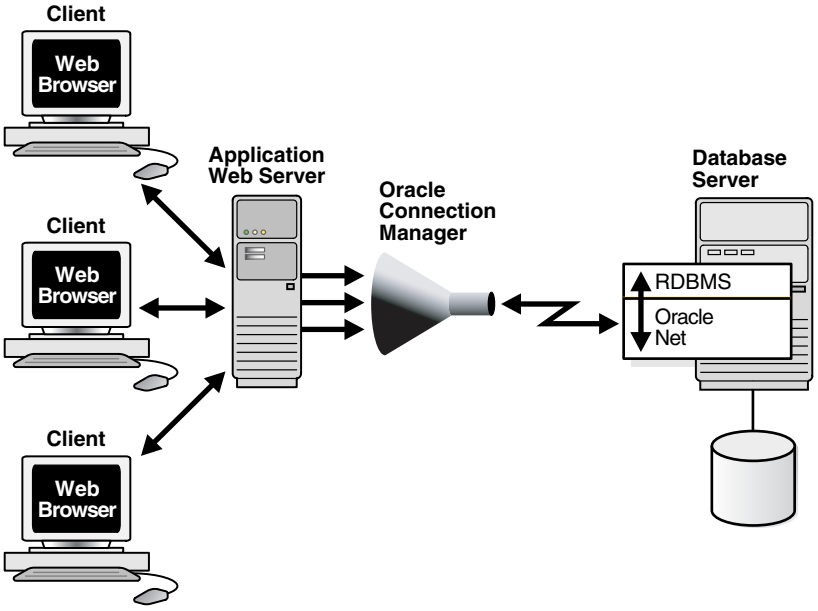


Table 1–1 summarizes the differences between connection pooling and session multiplexing, and provides recommendations for using them in the network.

Table 1–1 Connection Pooling and Session Multiplexing

Feature	Advantages	Disadvantages	Recommended for
Connection Pooling	■ Reduces the number of network resources used for each process ■ Supports large client populations ■ Maximizes the number of client/server sessions over a limited number of process connections ■ Optimizes network traffic and network resource utilization, such as network connection bandwidth ■ Enables identification and monitoring of real users ■ Enables middle-tier application Web servers or applications that need to access back-end database to support additional services, such as Oracle Application Server ■ Requires only a single transport for clients with multiple applications ■ Requires only a single network connection for database links	Database sessions should use the `IDLE_TIME` resource parameter.	Networks where many clients run interactive "high think/search time" applications such as messaging and OLAP.
Session Multiplexing	■ Limits the number of network resources used for each process ■ Supports large client populations ■ Maximizes the number of client/server sessions over a limited number of process connections ■ Optimizes resource utilization ■ Enables identification and monitoring of real users ■ Enables mid-tier applications to support additional services ■ Requires only a single transport for clients with multiple applications ■ Requires only a single network connection for database links	Clients must connect to Oracle Connection Manager.	Networks where continuous connectivity is required.

Understanding Performance

System performance is important to users. Users usually start to notice performance when a system takes longer than one second to respond. Oracle Net configuration can be modified to enhance system performance.

This section discusses performance considerations. It contains the following topics:

- Listener Queue Size
- Session Data Unit Size for Data Transfer Optimization
- Persistent Buffer Flushing for TCP/IP
- Sockets Directory Protocol

- Availability

Listener Queue Size

If you anticipate receiving a large number of connection requests for a listening process (such as a listener or Oracle Connection Manager) over TCP/IP, then Oracle Net enables you to configure the listening queue to be higher than the system default.

Session Data Unit Size for Data Transfer Optimization

Before sending data across the network, Oracle Net buffers and encapsulates data into the **session data unit (SDU)**. Oracle Net sends the data stored in this buffer when the buffer is full, flushed, or when database server tries to read data. When large amounts of data are being transmitted or when the message size is consistent, adjusting the size of the SDU buffers can improve performance, network utilization, or memory consumption. You can deploy SDU at the client, application Web server, and database.

Tuning your application to reduce the number of round trips across the network is the best way to improve your network performance. If this is done, then it is also possible to optimize data transfer by adjusting the size of the SDU.

Table 1–2 outlines considerations for modifying the size of the SDU.

Table 1–2 SDU Considerations

Modify SDU size when:	Do not modify SDU size when:
■ The data coming back from the server is fragmented into separate packets ■ You are on a wide area network (WAN) that has long delays ■ The packet size is consistently the same ■ Large amounts of data are returned	■ The application can be tuned to avoid the delays listed in the adjacent column ■ You have a high speed network where the effect of the data transmission is negligible ■ Your requests return small amounts of data from the server

Note: Starting with Oracle Database 11*g*, Oracle Net Services has optimized bulk data transfer for certain components, such as Oracle SecureFiles LOBs and Oracle Data Guard redo transport services. The SDU size limit, as specified in the network parameter files, does not apply to these bulk data transfers.

See Also: "Configuring Session Data Unit" on page 14-1

Persistent Buffer Flushing for TCP/IP

Under certain conditions for some applications using TCP/IP, Oracle Net packets may not get flushed immediately to the network. Most often, this behavior occurs when large amounts of data are streamed. The implementation of TCP/IP itself is the reason for the lack of flushing, causing unacceptable delays. To remedy this problem, specify no delays in the buffer flushing process.

See Also: *Oracle Database Net Services Reference* for additional information about the `TCP.NODELAY` parameter

Sockets Directory Protocol

Oracle Net Services provides support for InfiniBand high-speed networks. InfiniBand is a high-bandwidth I/O architecture designed to increase communication speed between CPUs, server-side devices, and network subsystems. Specifically, Oracle Net Services provides support for Sockets Directory Protocol (**SDP**). SDP is an industry-standard wire protocol intended for use between InfiniBand network peers.

SDP reduces the overhead of TCP/IP by eliminating intermediate replication of data and transferring most of the messaging burden away from the CPU and onto the network hardware. The result is a low-**latency**, increased bandwidth, high-throughput connection that reduces the amount of CPU cycles dedicated to network processing.

The communication between clients, including Oracle Application Server or any other third-party middle-tier client, and Oracle Database 11*g* can take advantage of high-speed interconnect benefits. Oracle Application Server includes Oracle TCP/IP support as part of its installation.

A driver installed on the Oracle Application Server servers transparently converts TCP/IP support to SDP support. The SDP requests are then sent to an InfiniBand switch that processes and forwards the requests from the Oracle Application Server servers to the database server.

See Also: "Configuring SDP Support for InfiniBand Connections" on page 14-5

Availability

Availability to the database is crucial for any network. You can configure multiple listeners to handle client connection requests for the same database service. This is beneficial in Oracle Real Application Clusters configurations, where each instance has a listener associated with it. Multiple listener configurations enable you to use the following features.

- Connect-time failover enables clients to request a different listener, usually on a different node, if the first listener fails.
- Client load balancing enables clients to randomize requests to the multiple listeners, usually on different nodes. These features can be used together or separately. Together, they ensure access to the database and distribute the load to not overburden a single listener.

Understanding Network Security

Data access and secure transfer of data are important considerations when deploying Oracle Database. Granting and denying access to a database is crucial for a secure network environment. Oracle Net Services enables database access control using firewall access control.

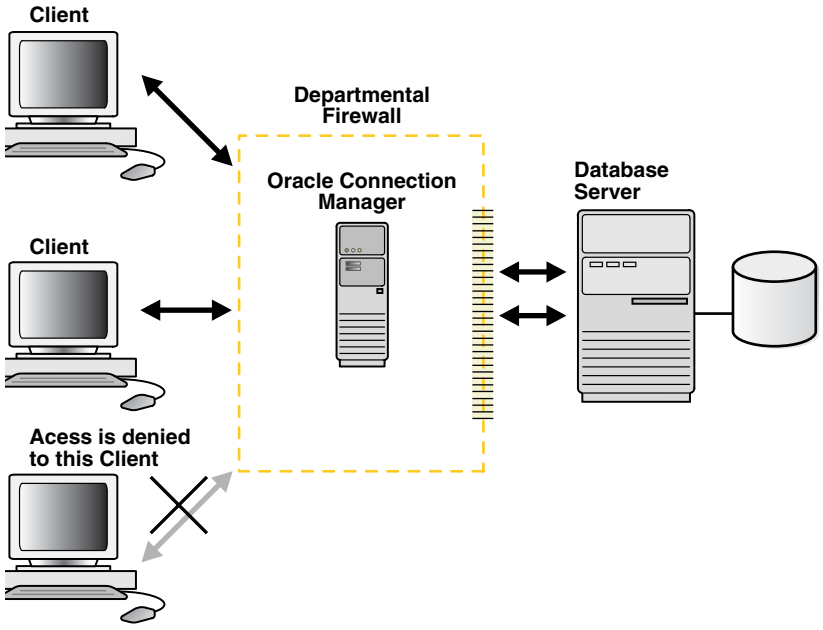
Firewall Access Control

Oracle Connection Manager can be configured to grant or deny client access to a particular database service or a computer. By specifying filtering rules, you can allow or restrict specific client access to a server, based on the following criteria:

- Source host names or IP addresses for clients
- Destination host names or IP addresses for servers
- Destination database service names
- Client use of **Oracle Advanced Security**

Figure 1–12 shows an Oracle Connection Manager positioned between three clients and an Oracle Database server. Oracle Connection Manager is configured to allow access to the first two clients and to deny access to the third.

Figure 1–12 Intranet Network Access Control with Oracle Connection Manager



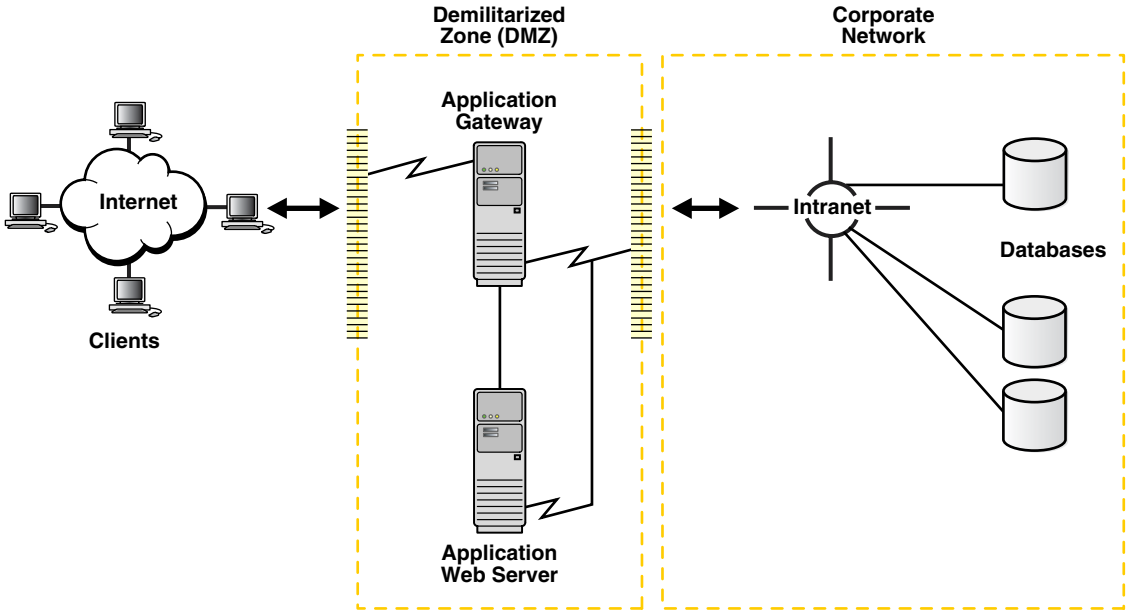
Although Oracle Connection Manager cannot be integrated with third-party firewall products, vendors can package it with their own products in a way that enables this product to serve as an application gateway.

In general, firewalls should be set to receive incoming requests, and allow outbound calls from Oracle Database. By defining filtering rules, you can limit access to the network.

Caution: Incorrectly setting your firewall options can cause security problems. Before changing your firewall settings, discuss the options and your network site policies with your system administrator.

Figure 1–13 shows an application gateway controlling traffic between internal and external networks and providing a single checkpoint for access control and auditing. As a result, unauthorized Internet hosts cannot directly access the database inside a corporation, but authorized users can still use Internet services outside the corporate network. This capability is critical in Internet environments to restrict remote access to sensitive data.

Figure 1–13 Internet Network Access Control with an Application Gateway



It is important to deploy at least two Oracle Connection Manager firewalls or Oracle Net Firewall proxies in an Internet network environment in the event that one firewall goes down.

Oracle Net Services Components

The connectivity, manageability, scalability, and security features are described in the following sections:

- About Oracle Net
- About Oracle Net Listener
- About Oracle Connection Manager
- About Networking Tools
- About Oracle Advanced Security

About Oracle Net

Oracle Net is a software layer that resides on the client and on the Oracle Database server. It is responsible for establishing and maintaining the connection between the client application and server, as well as exchanging messages between them, using industry-standard protocols. Oracle Net has two software components:

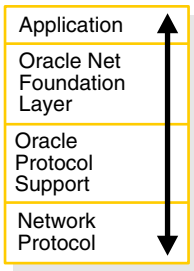
- Oracle Net Foundation Layer
- Oracle Protocol Support

Oracle Net Foundation Layer

On the client side, applications communicate with Oracle Net foundation layer to establish and maintain connections. The Oracle Net foundation layer uses Oracle protocol support that communicates with an industry-standard network protocol, such as TCP/IP, to communicate with the Oracle Database server.

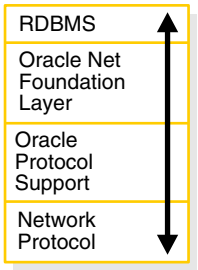
Figure 1–14 illustrates the communication stack on the client.

Figure 1–14 Oracle Net on the Client



The Oracle Database server side is similar to the client side as illustrated in Figure 1–15. A network protocol sends client request information to an Oracle protocol support layer, which then sends information to the Oracle Net foundation layer. The Oracle Net foundation layer then communicates with the Oracle Database server to process the client request.

Figure 1–15 Oracle Net on the Server



Oracle Protocol Support

The Oracle Net foundation layer uses Oracle protocol support to communicate with the following industry-standard network protocols:

- TCP/IP (version 4 and version 6)
- TCP/IP with SSL
- Named Pipes
- SDP

Oracle protocol support maps Oracle Net foundation layer functionality to industry-standard protocols used in client/server connections.

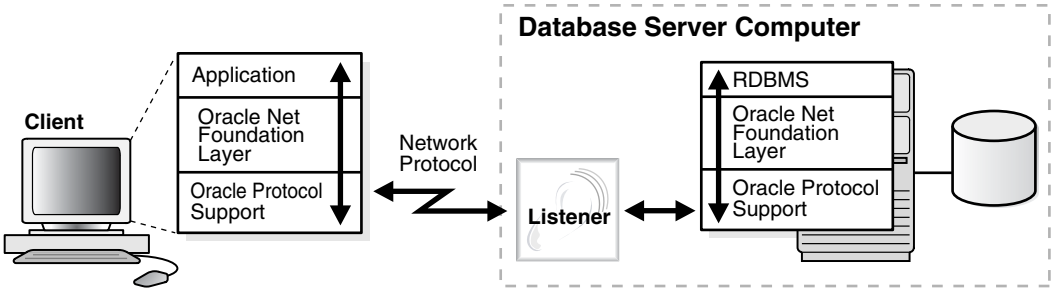
> **See Also:** "Understanding Oracle Protocol Support Layer" on page 4-5

About Oracle Net Listener

Oracle Database server receives the initial connection through **Oracle Net Listener**. Oracle Net Listener, referred to in this document as the **listener**, brokers a client request, handing off the request to the server. The listener is configured with a protocol address, and clients configured with the same protocol address can send connection requests to the listener. When a connection is established, the client and Oracle server communicate directly with one another.

Figure 1–16 shows the listener accepting a connection request from a client and forwarding that request to an Oracle server.

Figure 1–16 Listener in a Connection Request



See Also: Chapter 9, "Configuring and Administering Oracle Net Listener" for additional information about the listener

About Oracle Connection Manager

Oracle Connection Manager is the software component that resides on its own computer, separate from a client or an Oracle Database server. It proxies and screens requests for the database server. In addition, it multiplexes database sessions.

In its session multiplexing role, Oracle Connection Manager funnels multiple sessions through a single transport protocol connection to a particular destination. In this way, Oracle Connection Manager reduces the demand on resources needed to maintain multiple sessions between two processes by enabling the Oracle Database server to use fewer connection end points for incoming requests.

As an access control filter, Oracle Connection Manager controls access to Oracle databases.

See Also:

- "Session Multiplexing" on page 1-8
- "Firewall Access Control" on page 1-12 for a description of filtering

About Networking Tools

Oracle Net Services provides user interface tools and command-line utilities that enable you to easily configure, manage, and monitor the network.

- **Oracle Net Configuration Assistant** is a standalone tool that enables you to configure listeners and naming methods.
- **Oracle Enterprise Manager** combines configuration functionality across multiple file systems, along with listener administrative control to provide an integrated environment for configuring and managing Oracle Net Services.
- **Oracle Net Manager** provides configuration functionality for an Oracle home on a local client or server host.
- Command-line control utilities enable you to configure, administer, and monitor network components, including listeners and Oracle Connection Managers.

With Oracle Enterprise Manager or Oracle Net Manager, you can fine-tune the listener and naming method configuration created with Oracle Net Configuration Assistant. In addition, Oracle Enterprise Manager and Oracle Net Manager offer built-in wizards and utilities that enable to you to test connectivity, migrate data from one naming method to another, and create additional network components.

See Also: Chapter 7, "Managing Oracle Net Services"

About Oracle Advanced Security

Oracle Advanced Security is a separately licensable product that provides a comprehensive suite of security features for the Oracle environment. This suite of security features protects enterprise networks and securely extends corporate networks to the Internet. It provides a single source of integration with network encryption and authentication solutions, single sign-on services, and security protocols. Oracle Advanced Security integrates industry standards and delivers unparalleled security to the Oracle network and other networks.

See Also: *Oracle Database Advanced Security Administrator's Guide*

2

Identifying and Accessing the Database

This chapter explains how databases are identified and how clients access them. This chapter contains the following topics:

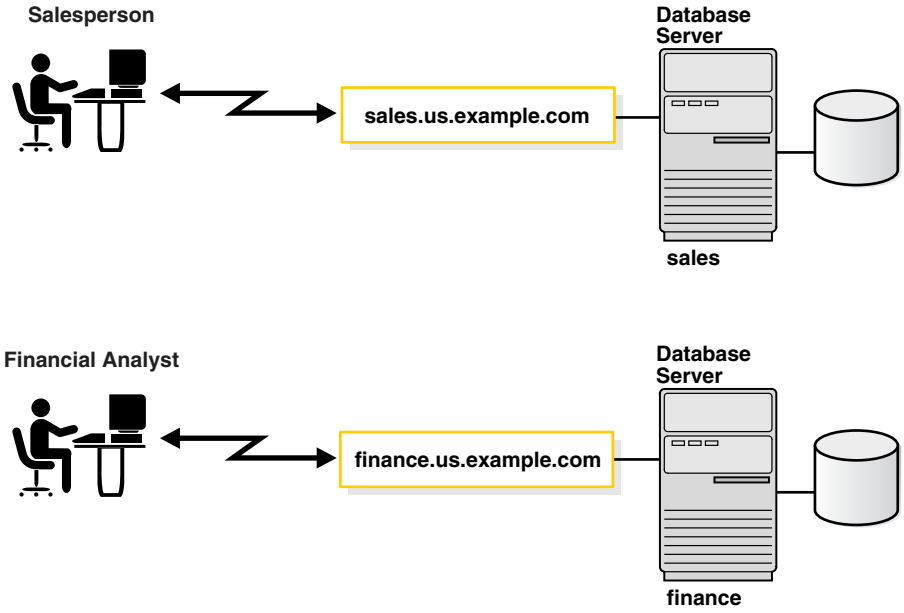
Understanding Database Instances

A database has at least one **instance**. An instance is comprised of a memory area called the **System Global Area (SGA)** and Oracle background processes. The memory and processes of an instance efficiently manage the associated database's data and serve the database users.

> **Note:** An instance also manages other services, such as **Oracle XML DB**.

Figure 2–1 shows two database instances, `sales` and `finance`, associated with their respective databases and service names.

Figure 2–1 One Instance for Each Database

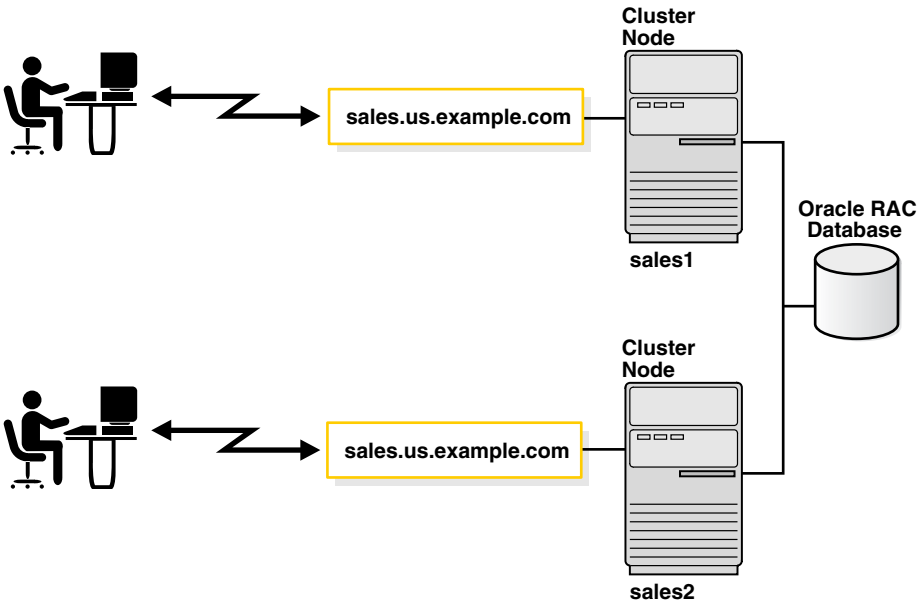


Instances are identified by an **instance name**, `sales` and `finance` in this example. The instance name is specified by the `INSTANCE_NAME` initialization parameter. The instance name defaults to the **Oracle system identifier (SID)** of the database instance.

Some hardware architectures allow multiple computers to share access to data, software, or peripheral devices. **Oracle Real Application Clusters (Oracle RAC)** (Oracle RAC) can take advantage of such architecture by running multiple instances on different computers that share a single physical database.

Figure 2–2 shows an Oracle RAC configuration. In this example, two instances, `sales1` and `sales2`, are associated with one database service, `sales.us.example.com`.

Figure 2–2 Multiple Instances Associated with an Oracle RAC Database

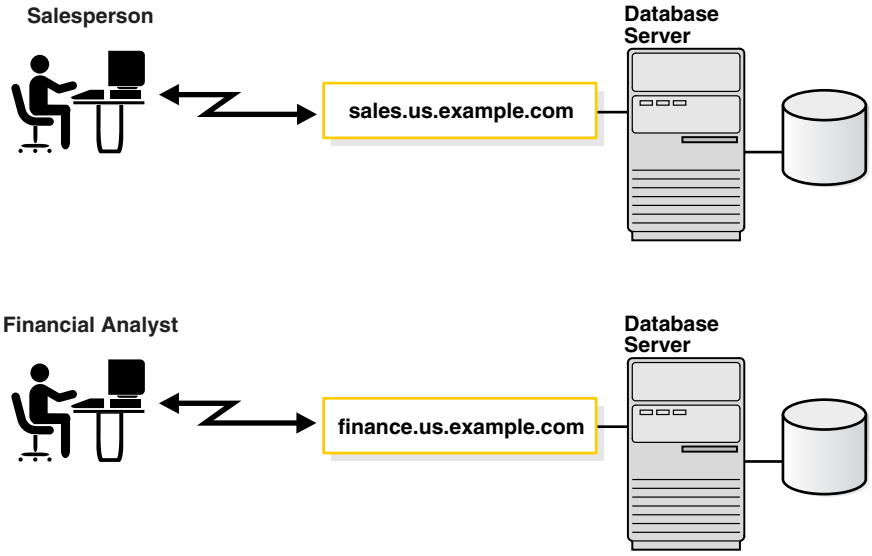


Understanding Database Services

An Oracle database is represented to clients as a **service**. A database can have one or more services associated with it.

Figure 2–3 shows two databases, each with its own database service for clients. One service, `sales.us.example.com`, enables salespersons to access the sales database. Another service, `finance.us.example.com`, enables financial analysts to access the finance database.

Figure 2–3 One Service for Each Database



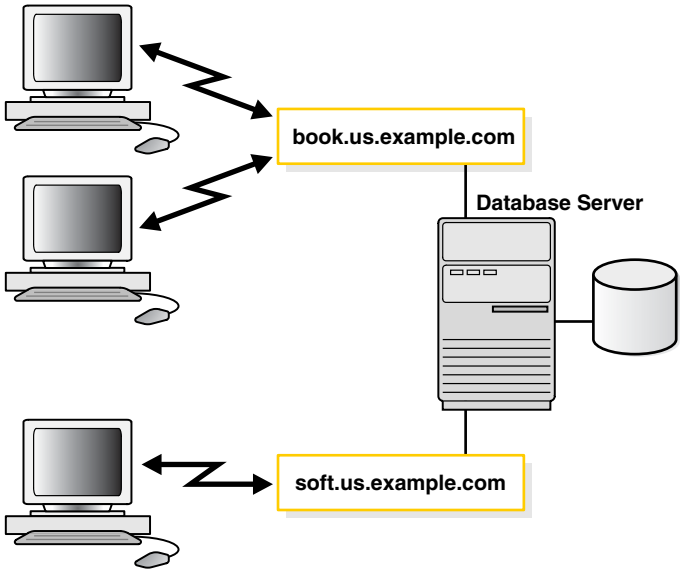
The sales and finance databases are each identified by a **service name**, `sales.us.example.com` and `finance.us.example.com`, respectively. A service name is a logical representation of a database. When an instance starts, it registers itself with a listener using one or more service names. When a client program or database connects to a listener, it requests a connection to a service.

A service name can identify multiple database instances, and an instance can belong to multiple services. For this reason, the listener acts as a mediator between the client and instances and routes the connection request to the appropriate instance. Clients connecting to a service need not specify which instance they require.

The service name is specified by the `SERVICE_NAMES` initialization parameter in the **server parameter file**. The server parameter file enables you to change initialization parameters with `ALTER SYSTEM` commands, and to carry the changes across a shutdown and startup. The DBMS_SERVICE package can also be used to create services. The service name defaults to the **global database name**, a name comprising the database name (`DB_NAME` initialization parameter) and domain name (`DB_DOMAIN` initialization parameter). In the case of `sales.us.example.com`, the database name is `sales` and the domain name is `us.example.com`.

Figure 2–4 shows clients connecting to multiple services associated with one database.

Figure 2–4 Multiple Services Associated with One Database



Associating multiple services with one database enables the following functionality:

- A single database can be identified different ways by different clients.
- A database administrator can limit or reserve system resources. This level of control enables better allocation of resources to clients requesting one of these services.

See Also:

- *Oracle Database Administrator's Guide* for additional information about initialization parameters
- *Oracle Database SQL Reference* for additional information about the `ALTER SYSTEM` statement
- *Oracle Database Reference* for additional information about the `SERVICE_NAMES` parameter
- *Oracle Database PL/SQL Packages and Types Reference* for additional information about the DBMS_SERVICE package.

Connecting to a Database Service

To connect to a database service, clients use a **connect descriptor** that provides the location of the database and the name of the database service. The following example is an Easy Connect descriptor that connects to a database service named `sales.us.example.com`, and the host `sales-server` (the port is 1521 by default):

```
sales-server/sales.us.example.com
```

The following example shows the `tnsnames.ora` entry for the preceding Easy Connect connect descriptor and database service:

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

This section contains the following topics:

- About Connect Descriptors
- About the Protocol Address
- About Service Registration

See Also: "Understanding Naming Methods" on page 2-12

About Connect Descriptors

A connect descriptor is comprised of one or more protocol addresses of the listener and the connect information for the destination service in the `tnsnames.ora` file. Example 2–1 shows a connect descriptor mapped to the `sales` database.

Example 2–1 Connect Descriptor

```
sales=
 (DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
     (SID=sales)
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_NAME=sales)))
```

As shown in Example 2–1, the connect descriptor contains the following parameters:

- The `ADDRESS` section contains the following:
  - `PROTOCOL` parameter, which identifies the listener protocol address. The protocol is `tcp` for TCP/IP.
  - `HOST` parameter, which identifies the host name. The host is `sales-server`.
  - `PORT` parameter, which identifies the port. The port is `1521`, the default port number.
- The `CONNECT_DATA` section contains the following:
  - `SID` parameter, which identifies the SID of the Oracle database. The SID is `sales`.
  - `SERVICE_NAME` parameter, which identifies the service. The destination service name is a database service named `sales.us.example.com`.

    The value for this connect descriptor parameter comes from the `SERVICE_NAMES` initialization parameter (`SERVICE_NAMES` uses a final `S`) in the initialization parameter file. The `SERVICE_NAMES` initialization parameter is typically the **global database name**, which includes the database name and domain name. In the example, `sales.us.example.com` has a database name of `sales` and a domain of `us.example.com`.
  - `INSTANCE_NAME` parameter, which identifies the database instance. The instance name is optional.

    The `INSTANCE_NAME` parameter in the initialization parameter file defaults to the SID entered during installation or database creation.

    See Also: "Understanding Database Instances" on page 2-1, and "Understanding Database Services" on page 2-3

About IPv6 Addresses in Connect Descriptors

A host can use IP version 4 (IPv4) and IP version 6 (IPv6) interfaces. IPv6 addresses and host names that resolve to IPv6 addresses are usable in the `HOST` parameter of a TNS connect address, which can be obtained through any of the supported Net naming methods listed in Table 2–1 on page 2-14.

End-to-end connectivity using IPv6 in Oracle Database 11*g* requires the following configuration:

- The client TNS connect address must connect to the Oracle Net Listener on the IPv6 endpoint.
- The database instance configured for Oracle Net Listener must listen for connection requests on IPv6 endpoints.

For a given host name, Oracle Net attempts to connect to all IP addresses returned by Domain Name System (DNS) name resolution until a successful connection is established or all addresses have been attempted. Suppose that in Example 2–1 the `sales-server` host is an IPv4-only host that is accepting client connections. DNS maps `sales-server` to the following IP addresses:

1. IPv6 address `2001:0DB8:0:0::200C:417A`
2. IPv4 address `192.168.2.213`

In this case, Oracle Net first tries to connect on the IPv6 address because it is first in the DNS list. In this example `sales-server` does not support IPv6 connectivity, so this attempt fails. Oracle Net proceeds to connect to the IPv4 address, which succeeds.

See Also:

- "TCP/IP Protocol" on page 4-6
- "Configuring Listening Protocol Addresses" on page 9-4
- "Configuring Oracle Connection Manager as a Bridge for IPv4 and IPv6" on page 10-7
- "IPv6 Network Connectivity" on page 4-8

About the Protocol Address

The address portion of the connect descriptor is the protocol address of the **listener**. To connect to a database service, clients first contact a listener process that typically resides on the database server. The listener receives incoming client connection requests and sends these requests to the database server. After the connection is established, the client and database server communicate directly.

The listener is configured to accept requests from clients at a **protocol address**. This address defines the protocol the listener is listening on and any other protocol-specific information. For example, the listener could be configured to listen at the following protocol address:

```
(DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)))
```

The preceding example shows a TCP/IP protocol address that specifies the host of the listener and a port number. Client connect descriptors configured with this same protocol address can send connection requests to this listener.

About Service Registration

The connect descriptor specifies the database service name with which clients seek to establish a connection. The listener knows which services can handle connection requests because an Oracle database dynamically registers this information with the listener. This process of registration is called **service registration**. Registration also provides the listener with information about the database instances and the service handlers available for each instance. A service handler can be a **dispatcher** or **dedicated server**.

Specifying an Instance Name

If connecting to a specific instance of the database is required, then clients can specify the `INSTANCE_NAME` of a particular instance in the connect descriptor. This feature can be useful if you have an Oracle Real Application Clusters configuration. For example, the following connect descriptor specifies the instance name `sales1` that is associated with `sales.us.example.com`.

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    (INSTANCE_NAME=sales1))
```

Specifying a Service Handler

Clients that always want to use a particular service handler type can use a connect descriptor to specify the service handler type. In the following example, a connect descriptor is configured to use a dispatcher for a shared server configuration, as indicated by `(SERVER=shared)`.

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
     (SERVER=shared))
```

When the listener receives the client request, it selects one of the registered service handlers. Depending on the type of handler selected, the communication protocol used, and the operating system of the database server, the listener performs one of the following actions:

- Hands the connect request directly off to a dispatcher.
- Sends a redirect message back to the client with the location of the dispatcher or dedicated server process. The client then connects directly to the dispatcher or dedicated server process.
- Spawns a dedicated server process and passes the client connection to the dedicated server process.

After the listener has completed the connection operation for the client, the client communicates directly with the Oracle database without the listener's involvement. The listener resumes listening for incoming network sessions.

The following are considerations when specifying service handlers:

- If you want the client to use a dedicated server, then specify `(SERVER=dedicated)`. If the `SERVER` parameter is not set, then shared server configuration is assumed. However, the client will use a dedicated server if no dispatchers are available.

- If database resident connection pooling is enabled on the server, then specify `(SERVER=pooled)` to get a connection from the pool. If database resident connection pooling is not enabled on the server, then the client request is rejected, and the user receives an error message.

  See Also:

  - "Understanding Service Handlers" on page 2-8 for a description about these service handler types
  - "About the Listener and Connection Requests" on page 5-2 for a discussion about how the listener works with service handlers
  - "About Database Resident Connection Pooling" on page 2-11
  - *Oracle Call Interface Programmer's Guide* and *Oracle Database Administrator's Guide* for additional information about enabling and configuring database resident connection pooling

Understanding Service Handlers

Service handlers act as connection points to an Oracle database. A service handler can be a **shared server process** or a **dedicated server** process, or pooled.

This section contains the following topics:

- About Dispatchers
- About Dedicated Server Processes
- About Database Resident Connection Pooling

About Dispatchers

The shared server architecture uses a dispatcher process to direct client connections to a common request queue. An idle shared server process from a shared pool of server processes picks up a request from the common queue. This approach enables a small pool of server processes to serve a large number of clients. A significant advantage of the shared server model over the dedicated server model is reduced system resources, enabling support of an increased number of users.

The listener uses the dispatcher as a type of **service handler** to which it can direct client requests. When a client request arrives, the listener performs one of the following actions:

- Hands the connection request directly to a dispatcher.
- Issues a redirect message to the client, containing the protocol address of a dispatcher. The client then terminates the network session to the listener and establishes a network session to the dispatcher, using the network address provided in the redirect message.

The listener uses direct hand off whenever possible. Redirect messages are used, for example, when dispatchers are remote to the listener.

Figure 2–5 shows the listener handing a connection request directly to a dispatcher.

1. The listener receives a client connection request.
2. The listener hands the connect request directly to the dispatcher.
3. The client is now connected to the dispatcher.

Figure 2–5 Direct Hand-Off to a Dispatcher

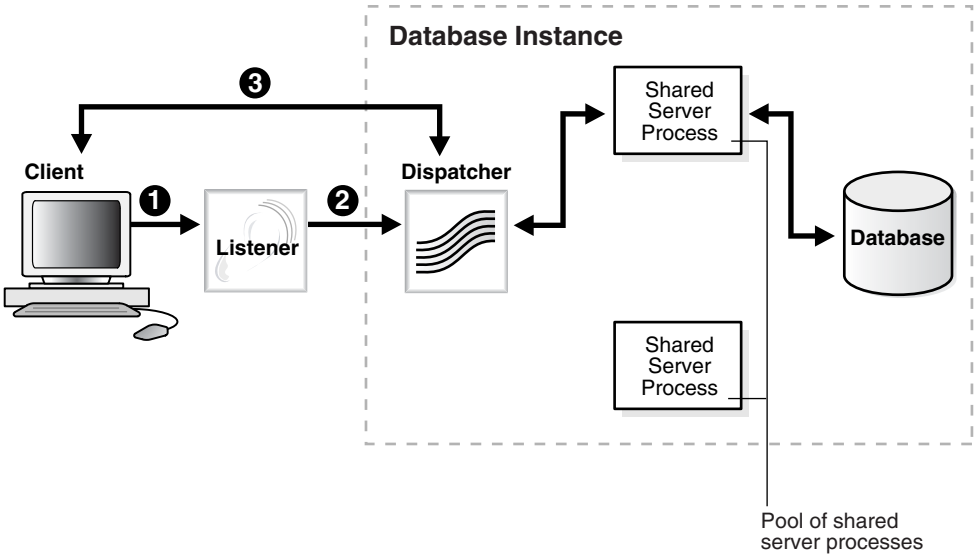
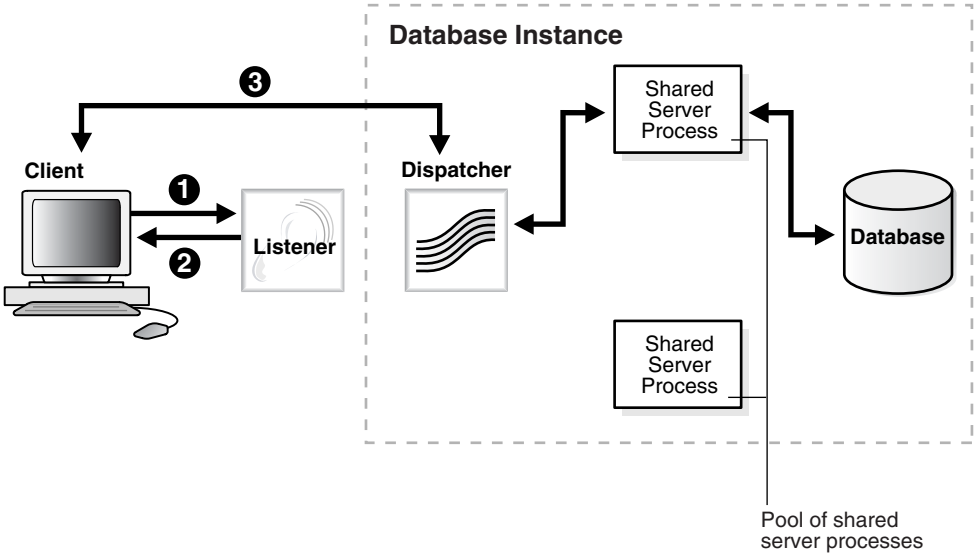


Figure 2–6 shows the role of a dispatcher in a redirected connection.

1. The listener receives a client connection request.
2. The listener provides the location of the dispatcher to the client in a redirect message.
3. The client connects directly to the dispatcher.

Figure 2–6 Redirected Connection to a Dispatcher



About Dedicated Server Processes

In a dedicated server configuration, the listener starts a separate dedicated server process for each incoming client connection request dedicated to servicing the client. After the session is complete, the dedicated server process terminates. Because a dedicated server process has to be started for each connection, this configuration may require more system resources than shared server configurations.

A dedicated server process is a type of service handler that the listener starts when it receives a client request. To complete a client/server connection, one of the following actions occurs:

- The dedicated server inherits the connection request from the listener.
- The dedicated server informs the listener of its listening protocol address. The listener passes the protocol address to the client in a redirect message and terminates the connection. The client connects to the dedicated server directly using the protocol address.

One of the preceding actions is selected based on the operating system and the transport protocol.

If the client and database exist on the same computer, then a client connection can be passed directly to a dedicated server process without going through the listener. This is known as a bequeath protocol. The application initiating the session spawns a dedicated server process for the connection request. This happens automatically if the application used to start the database is on the same computer as the database.

Note: In order for remote clients to connect to dedicated servers, the listener and the database instance must be running on the same computer.

Figure 2–7 shows the listener passing a client connection request to a dedicated server process.

1. The listener receives a client connection request.
2. The listener starts a dedicated server process, and the dedicated server inherits the connection request from the listener.
3. The client is now connected directly to the dedicated server.

Figure 2–7 Connection to a Dedicated Server Process

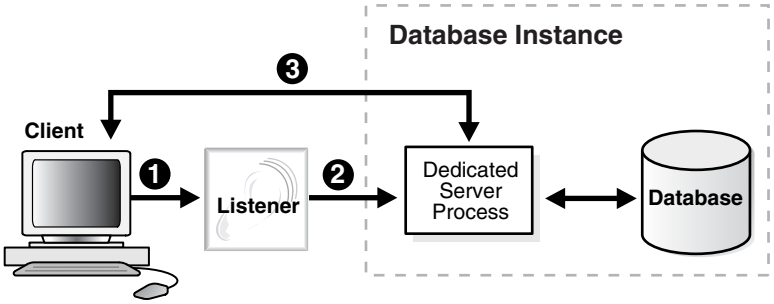
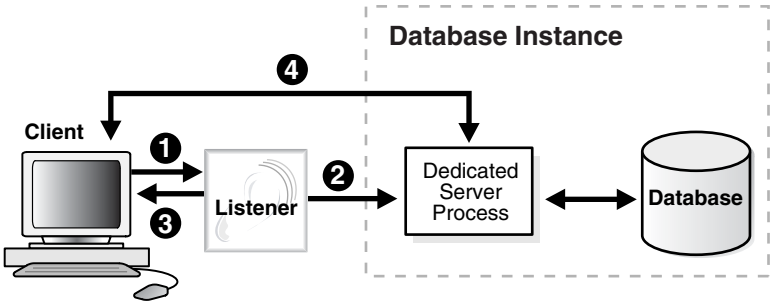


Figure 2–8 shows the role of a dedicated server in a redirected connection.

1. The listener receives a client connection request.
2. The listener starts a dedicated server process.
3. The listener provides the location of the dedicated server process to the client in a redirect message.
4. The client connects directly to the dedicated server.

Figure 2–8 Redirected Connection to a Dedicated Server Process



About Database Resident Connection Pooling

Database resident connection pooling provides a connection pool in the database server for typical Web application usage scenarios in which an application acquires a database connection, works on it for a relatively short duration, and then releases it. Database resident connection pooling pools "dedicated" servers. A pooled server is the equivalent of a server foreground process and a database session combined.

Database resident connection pooling complements middle-tier connection pools that share connections between threads in a middle-tier process. In addition, it enables sharing of database connections across middle-tier processes on the same middle-tier host and even across middle-tier hosts. This results in significant reduction in key database resources needed to support a large number of client connections, thereby reducing the database tier memory footprint and boosting the scalability of both middle-tier and database tiers. Having a pool of readily available servers has the additional benefit of reducing the cost of creating and closing client connections.

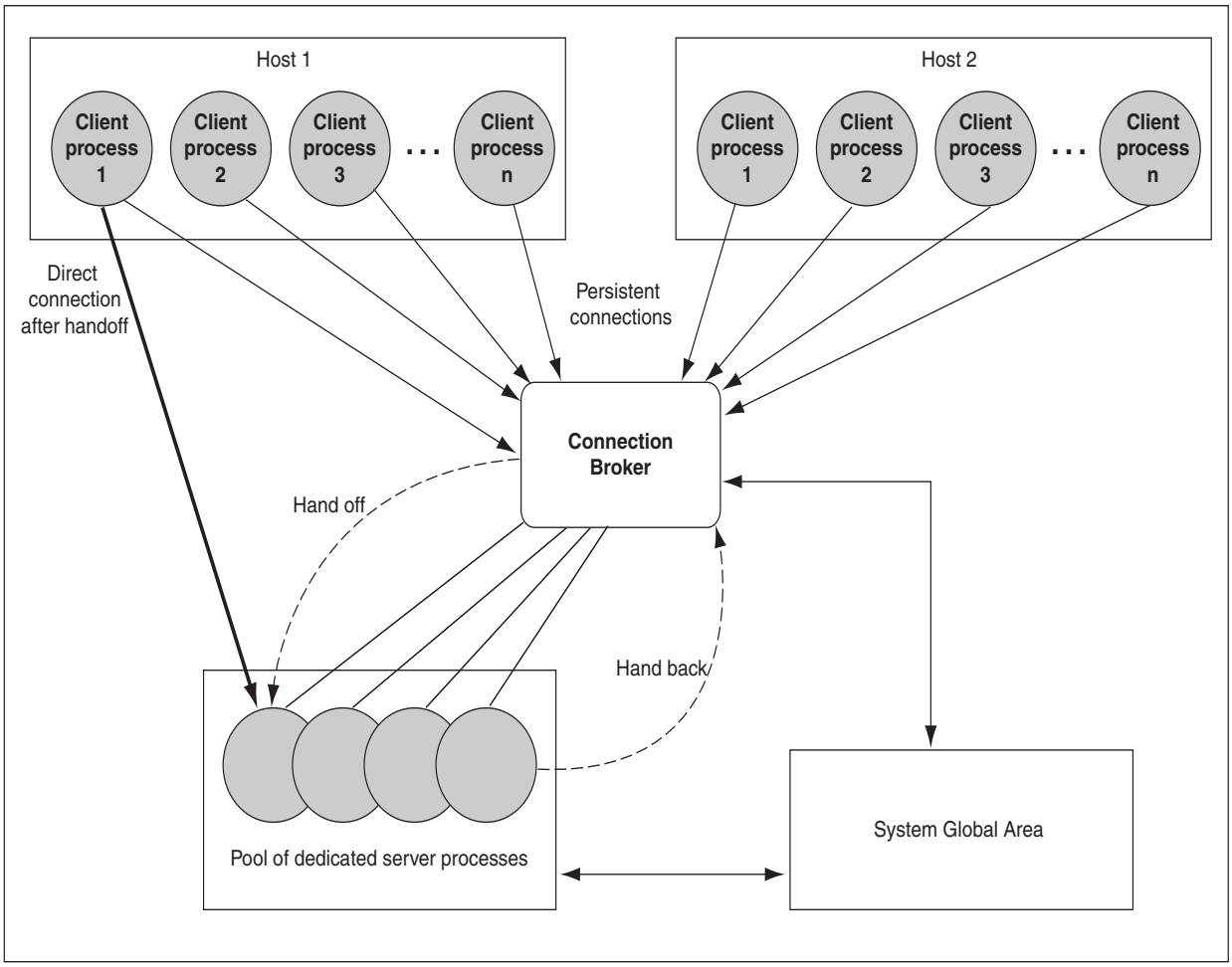
Database resident connection pooling provides pooling for **dedicated connection**s across client applications and processes. This feature is useful for applications that must maintain persistent connections to the database and optimize server resources (such as memory).

Clients obtaining connections out of the database resident connection pool are persistently connected to a background process, the connection broker, instead of the dedicated servers. The connection broker implements the pool functionality and performs the multiplexing of inbound connections from the clients to a pool of dedicated servers with sessions.

When a client must perform database work, the connection broker picks up a dedicated server from the pool and assigns it to the client. Subsequently, the client is directly connected to the dedicated server until the request is served. After the server finishes processing the client request, the server goes back into the pool and the connection from the client is restored to the connection broker.

Figure 2–9 shows the process.

Figure 2–9 Dedicated Server Processes Handling Connections Through the Connection Broker Process



Understanding Naming Methods

A naming method is a resolution method used by a client application to resolve a connect identifier to a connect descriptor when attempting to connect to a database service. Users initiate a connection request by providing a **connect string**. A connect string includes a user name and password, along with a **connect identifier**. A connect identifier can be the connect descriptor or a name that resolves to a connect descriptor. The connect descriptor contains:

- Network route to the service, including the location of the listener through a protocol address
- A database **service name** or **Oracle system identifier (SID)**

The following `CONNECT` command uses a connect string that has a complete connect descriptor as the connect identifier instead of a net service name. The string should be entered on a single line. It is shown on two lines because of page width.

```
SQL> CONNECT hr@(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server1)
(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

One of the most common connect identifiers is a **net service name**, a simple name for a service. The following `CONNECT` command uses a connect string that uses net service name `sales` as the connect identifier:

```
SQL> CONNECT hr@sales

Enter password: password
```

When net service name `sales` is used, connection processing takes place by first mapping `sales` to the connect descriptor. This mapped information is accessed by **naming method**s. The following naming methods are available:

- Local naming
- Directory naming
- Easy Connect naming
- External naming

Choosing a Naming Method

Selecting the appropriate naming method for mapping names to connect descriptors depends upon the size of the organization.

- For a small organization with only a few databases, use Easy Connect naming to make TCP/IP connections with the host name of the database server or local naming to store names in `tnsnames.ora` file on the clients.
- For large organizations with several databases, use directory naming to store names in a centralized directory server.
- For an Internet network, configure the application Web servers needed to connect to the databases with the local naming method.

Table 2–1 summarizes the relative advantages and disadvantages of each naming method and provides recommendations for using them in the network.

Table 2–1 Naming Methods: Advantages and Disadvantages

Naming Method	Description	Advantages/Disadvantages	Recommended for:
Local Naming	Stores net service names and their connect descriptors in a localized configuration file named `tnsnames.ora`, which by default is located in the `network/admin` subdirectory under the Oracle Database home.	**Advantages**: ▪ Provides a straightforward method for resolving net service name addresses. ▪ Resolves net service names across networks running different protocols. **Disadvantage**: Requires local configuration of all net service name and address changes.	Simple distributed networks with a small number of services that change infrequently.
Directory Naming	Stores connect identifiers in a centralized LDAP-compliant **directory server** to access a database service.	**Advantages**: ▪ Centralizes network names and addresses in a single place, facilitating administration of name changes and updates. This eliminates the need for an administrator to make changes to what potentially could be hundreds or even thousands of clients. ▪ Directory stores names for other services. ▪ Tools provide simple configuration. **Disadvantage**: Requires access to a directory server.	Large, complex networks (over 20 databases) that change on a frequent basis.
Easy Connect Naming	Enables clients to connect to an Oracle database server by using a TCP/IP connect string consisting of a host name and optional port and service name.	**Advantages**: ▪ Requires minimal user configuration. The user can provide only the name of the database host to establish a connection. ▪ The easy naming method requires no client-side configuration. ▪ Eliminates the need to create and maintain a local names configuration file (`tnsnames.ora`). **Disadvantage**: Available only in a limited environment, as indicated in the **Recommended for** column.	Simple TCP/IP networks that meet the criteria listed: ▪ Your client and server are connecting using TCP/IP. ▪ No features requiring a more advanced connect descriptor are required.
External Naming	Stores net service names in a supported third-party naming service, such as **Network Information Service (NIS)** External Naming.	**Advantage**: Enables administrators to load Oracle net service name into their native name service using tools and utilities with which they are already familiar. **Disadvantage**: Requires a third-party naming services that cannot be administered using Oracle Net products.	Networks with existing name services.

A naming method configuration consists of the following steps:

1. Select a naming method.
2. Map connect descriptors to the names.
3. Configure clients to use the naming method.

Establishing a Client Session using a Naming Method

A typical process for establishing a client session using a naming method is as follows:

1. The client initiates a connect request by providing a connect identifier.
2. The connect identifier is resolved to a connect descriptor by a naming method.
3. The client makes the connection request to the address provided in the connect descriptor.
4. A listener receives the request and directs it to the appropriate database server.
5. The connection is accepted by the database server.

> **Note:** Besides connect descriptors, you can use naming methods to map a connect name to a protocol address or protocol address list.

Entering a Connect String

After the network components are started you should be able to make a connection across the network. How you make a connection depends upon the **naming method**, and the tool used for the connection.

The **connect string** takes the following format:

```
CONNECT username@connect_identifier
```

On most operating systems, you can define a default **connect identifier**. When using the default, a connect identifier does not need to be specified in the connect string. To define a default connect identifier, use the `TWO_TASK` environment variable on Linux and UNIX platforms or the `LOCAL` environment variable or registry entry on Microsoft Windows.

For example, if the `TWO_TASK` environment variable is set to `sales`, then you can connect to a database from SQL*Plus with `CONNECT username` rather than `CONNECT username@sales`. Oracle Net checks the `TWO_TASK` variable, and uses the value `sales` as the connect identifier.

> **See Also:** Oracle operating system-specific documentation for instructions on setting `TWO_TASK` and `LOCAL`

Connect Identifier and Connect Descriptor Syntax Characteristics

Connect identifiers used in a connect string cannot contain spaces, unless enclosed within single quotation marks (') or double quotation marks ("). In the following examples, a connect identifier and a connect descriptor that contain spaces are enclosed within single quotation marks:

```
CONNECT scott@'(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)
(PORT=1521))(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))'
Enter password: password

CONNECT scott@'cn=sales, cn=OracleContext, dc=us, dc=example, dc=com'
Enter password: password
```

Single quotation marks are required if double quotation marks are used in a connect identifier. For example:

```
CONNECT scott@'sales@"good"example.com'
Enter password: password
```

Similarly, double quotation marks are required if a single quotation marks are used in a connect identifier. For example:

```
CONNECT scott@"cn=sales, cn=OracleContext, ou=Mary's Dept, o=example"
Enter password: password
```

Enhancing Service Accessibility using Multiple Listeners

For some configurations, such as Oracle Real Application Clusters, multiple listeners on multiple nodes can be configured to handle client connection requests for the same database service. In the following example, `sales.us.example.com` can connect using listeners on either `sales1-server` or `sales2-server`.

```
(DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

A multiple-listener configuration also enables you to leverage the failover and load balancing features, either individually or in combination with each other. The following topics describe the features:

- About Connect-time Failover
- About Transparent Application Failover
- About Client Load Balancing
- About Connection Load Balancing

About Connect-time Failover

The **connect-time failover** feature enables clients to connect to another listener if the initial connection to the first listener fails. The number of listener protocol addresses determines how many listeners are tried. Without connect-time failover, Oracle Net attempts a connection with only one listener.

About Transparent Application Failover

The **Transparent Application Failover (TAF)** feature is a run-time failover for high availability environments, such as Oracle Real Application Clusters. TAF fails over and reestablishes application-to-service connections. It enables client applications to automatically reconnect to the database if the connection fails and, optionally, resume a `SELECT` statement that was in progress. The reconnection happens automatically from within the **Oracle Call Interface (OCI)** library.

About Client Load Balancing

The **client load balancing** feature enables clients to randomize connection requests among the listeners. Oracle Net progresses through the list of protocol addresses in a random sequence, balancing the load on the various listeners. Without client load balancing, Oracle Net progresses through the list of protocol addresses sequentially until one succeeds.

About Connection Load Balancing

The **connection load balancing** feature improves connection performance by balancing the number of active connections among multiple dispatchers. In a single-instance environment, the listener selects the least-loaded dispatcher to handle the incoming client requests. In an Oracle Real Application Clusters environment, connection load balancing can balance the number of active connections among multiple instances.

Due to dynamic service registration, a listener is always aware of all instances and dispatchers regardless of their location. Depending on the load information, a listener decides which instance and, if shared server is configured, to which dispatcher to send the incoming client request.

In a shared server configuration, a listener selects a dispatcher in the following order:

1. Least-loaded node
2. Least-loaded instance
3. Least-loaded dispatcher for that instance

In a dedicated server configuration, a listener selects an instance in the following order:

1. Least-loaded node
2. Least-loaded instance

If a database service has multiple instances on multiple nodes, then the listener chooses the least-loaded instance on the least-loaded node. If shared server is configured, then the least-loaded dispatcher of the selected instance is chosen.

3

Managing Network Address Information

This chapter describes how network address information for Oracle Net Services can be stored in local files or in a centralized directory server. Using localized management, network address information is stored in `tnsnames.ora` files on each computer in the network. Using centralized management, network address information is stored in centralized **directory server**.

The chapter contains the following topics:

Using Localized Management

When localized management is used, network computers are configured with the files described in Table 3–1. The files are stored locally on the computers.

Table 3–1 Oracle Net Configuration Files Used with Localized Management

Configuration File	Description
`tnsnames.ora`	Located primarily on the clients, this file contains **net service name**s mapped to **connect descriptor**s. This file is used for the **local naming** method.
`sqlnet.ora`	Located on client and database server computers, this file may include: ■ Client domain to append to unqualified service names or net service names ■ Order of naming methods the client should use when resolving a name ■ Logging and tracing features to use ■ Route of connections ■ External naming parameters ■ Oracle Advanced Security parameters ■ Database access control parameters
`listener.ora`	Located on the database server, this configuration file for the **listener** may include: ■ Protocol addresses it is accepting connection requests on ■ Database and nondatabase services it is listening for ■ Control parameters used by the listener

Table 3–1 (Cont.) Oracle Net Configuration Files Used with Localized Management

Configuration File	Description
`cman.ora`	Located on the computer where Oracle Connection Manager runs, this configuration file includes the following components: ■ A listening endpoint ■ Access control rule list ■ Parameter list Each Oracle Connection Manager configuration is encapsulated within a single name-value (NV) string, which consists of the preceding components.

Configuration files are typically created in the `ORACLE_HOME/network/admin` directory. However, configuration files can be created in other directories. Oracle Net will check the other directories for the configuration files. For example, the order checking the `tnsnames.ora` file is as follows:

1. The directory specified by the `TNS_ADMIN` environment variable. If the file is not found in the directory specified, then it is assumed that the file does not exist.
2. If the `TNS_ADMIN` environment variable is not set, then Oracle Net will check the `ORACLE_HOME/network/admin` directory.

Note: On Microsoft Windows, the `TNS_ADMIN` environment variable is used if it is set in the environment of the process. If the `TNS_ADMIN` environment variable is not defined in the environment, or the process is a service which does not have an environment, then Microsoft Windows scans the registry for a `TNS_ADMIN` parameter.

See Also: Oracle operating system-specific documentation

Using a Directory Server for Centralized Management

When an Oracle network uses a directory server, the directory server manages global database links as net service names for the Oracle databases in the network. Users and programs can use a global link to access objects in the corresponding database.

Oracle Net Services uses a centralized directory server as one of the primary methods for storing **connect identifier**s. Clients use the connect identifiers in their connect string, and the directory server resolves the connect identifier to a connect descriptor that is passed back to the client. This feature is called **directory naming**. This type of infrastructure reduces the cost of managing and configuring resources in a network.

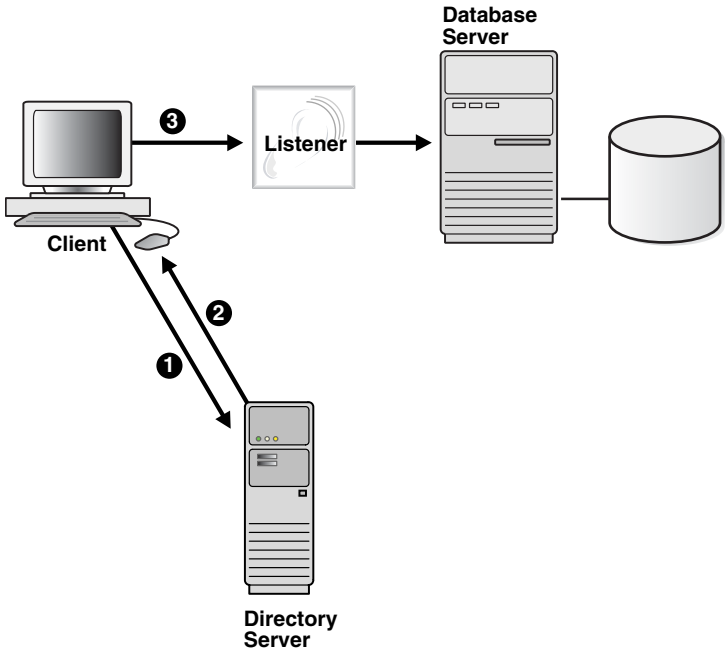
Oracle Internet Directory provides a centralized mechanism for managing and configuring a distributed Oracle network. The directory server can replace client-side and server-side localized `tnsnames.ora` files.

Figure 3–1 shows a client resolving a connect identifier through a directory server.

1. The client contacts to the directory server to resolve a connect identifier to a connect descriptor.
2. The directory server resolves the connect identifier and retrieves the connect descriptor for the client.

3. The client sends the connection request to the listener, using the connect descriptor.

Figure 3–1 Client Using a Directory Server to Resolve a Connect Identifier



Note: **Java Database Connectivity (JDBC) Drivers** support directory naming. See the *Oracle Database JDBC Developer's Guide and Reference* for additional information.

This section contains the following topics:

- Understanding the Directory Information Tree
- Understanding Oracle Context
- Understanding Net Service Alias Entries
- Who Can Add or Modify Entries in the Directory Server
- Client Connections Using Directory Naming
- Considerations When Using Directory Servers
- Limitations of Directory Naming Support with Microsoft Active Directory

Understanding the Directory Information Tree

Directory servers store information in a hierarchical namespace structure called a **directory information tree (DIT)**. Each node in the tree is called an **entry**. Oracle Net Services uses both the tree structure and specific entries in the tree. DITs are commonly aligned with an existing domain structure such as a Domain Name System (DNS) structure or a geographic and organization structure.

Figure 3–2 shows a DIT structured according to DNS domain components.

Figure 3–2 DNS Domain Component DIT

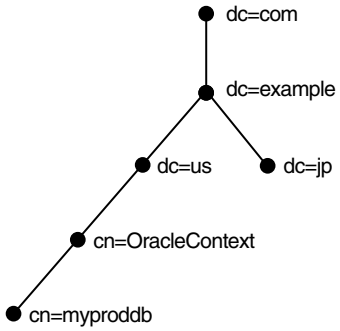
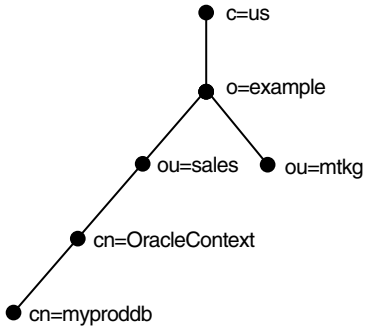


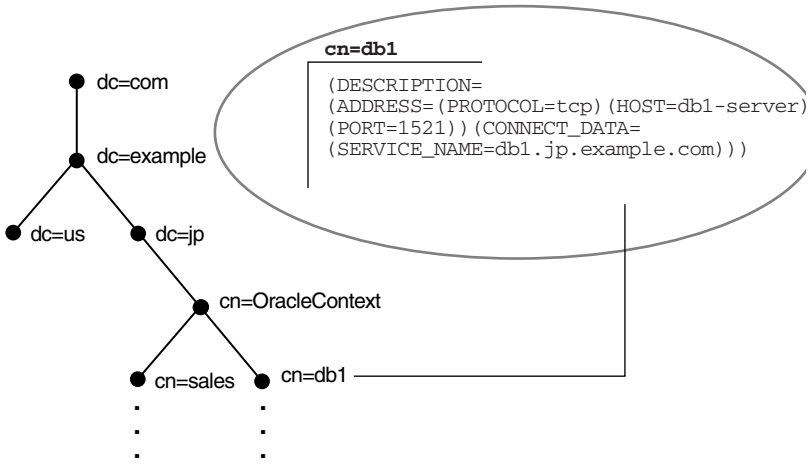
Figure 3–3 shows a DIT structured according to country, organization, and organizational units. This structure is commonly referred to as an X.500 DIT.

Figure 3–3 X.500 DIT



For example, consider Figure 3–4. The `cn=sales` and `cn=db1` entries represent a net service name and a database service, respectively. Additional entries under `cn=sales` and `cn=db1` contain the connect descriptor information. These entries are not represented in the graphic. The `cn=sales` and `cn=db1` entries enable clients to connect to the database using connect strings `CONNECT` *`username`*`@sales` and `CONNECT` *`username`*`@db1`.

Figure 3–4 Database Service and Net Service Entries in a DIT



Each entry is uniquely identified by a **distinguished name (DN)**. The DN indicates exactly where the entry resides in the directory server hierarchy. The DN for `db1` is

dn:cn=db1,cn=OracleContext,dc=jp,dc=example,dc=com. The DN for sales is dn:cn=sales,cn=OracleContext,dc=jp,dc=example,dc=com. The format of a DN places the lowest component of the DIT to the left, then moves progressively up the DIT.

Each DN is made up of a sequence of relative distinguished names (RDNs). The RDN consists of an **attribute**, such as cn, and a value, such as db1 or sales. The RDN for db1 is cn=db1, and the RDN for sales is cn=sales. The attribute and its value uniquely identify the entry.

Fully-qualified Names for Domain Component Namespaces

For domain component namespaces, the default directory entry defined for the client must be in one of the following formats:

```
dc[,dc][...]
```

```
ou,dc[,dc][...]
```

In the preceding syntax, [dc] represents an optional domain component and [...] represents additional domain component entries.

The fully-qualified name used in the connect identifier by the client must be in one of the following formats:

```
cn.dc[.dc][...]
```

```
cn[.ou]@dc[.dc][...]
```

In the preceding syntax, [cn] represents the Oracle Net entry.

Example 3–1 Using a Fully-qualified Name with an Organizational Unit

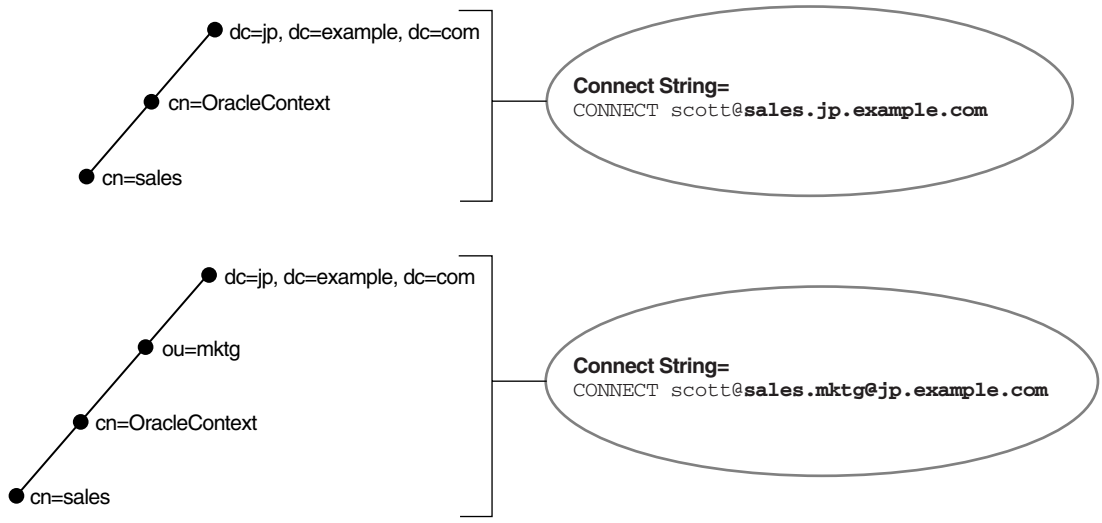
Consider a directory server that contains an entry for database object sales with a DN of cn=sales,cn=OracleContext,dc=jp,dc=example,dc=com. In this example, the client requires a connect identifier of sales.jp.example.com.

Consider a similar entry that contains database object sales with a DN of cn=sales,cn=OracleContext,**ou=mktg**,dc=jp,dc=example,dc=com.

Because domain components must be separated from organization units, the client must use the format cn.ou@dc.dc.dc. The client requires the connect identifier to be sales.mktg@jp.example.com.

Figure 3–5 illustrates the preceding example.

Figure 3–5 Fully-qualified Name for Domain Component Namespaces



Fully-qualified Names for X.500 Namespaces

For X.500 namespaces, the default directory entry defined for the client must be in one of the following formats:

```
[ou],o
```

```
[ou],o,c
```

In the preceding formats, `[ou]` represents an optional organizational unit name.

The fully-qualified name the client uses as the connect identifier must be in one of the following formats:

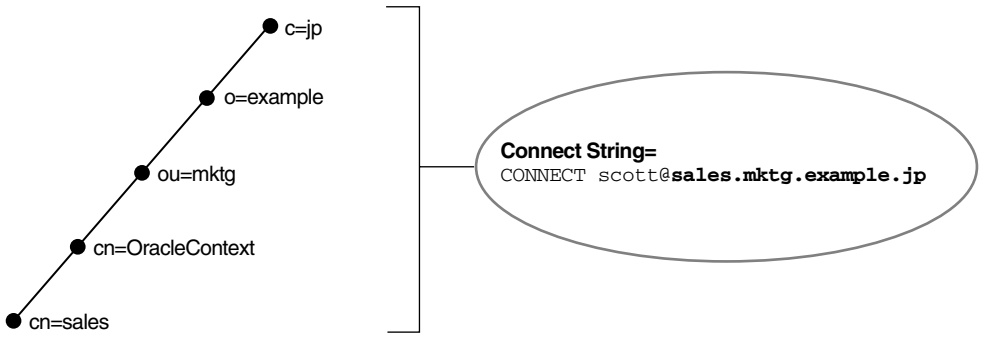
```
cn[.ou].o
```

```
cn[.ou].o.c
```

In the preceding formats, `[cn]` represents the Oracle Net entry.

For example, if the directory contains database object `sales` with a DN of `cn=sales,cn=OracleContext,ou=mktg,o=example,c=jp`, then the client requires a connect identifier of `sales.mktg.example.jp`. Figure 3–6 illustrates this example.

Figure 3–6 Fully-qualified Name for X.500 Namespaces

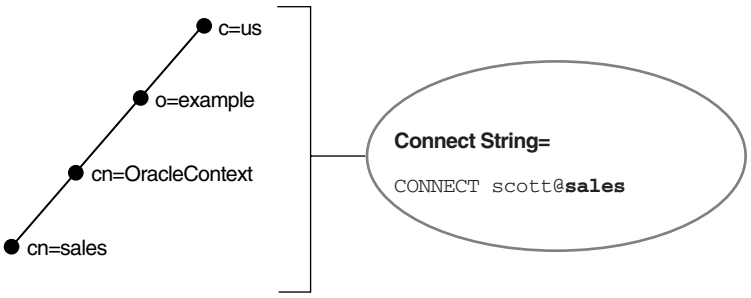


Using an Entry's Relative Name

If a client is configured with a default realm Oracle Context, then an entry can be identified by its relative name, and the service can be referred to by its common name. A relative name can be used if the entry is in the same Oracle Context that was configured to be the default Oracle Context for the client's Oracle home.

Consider a directory server that contains an entry for a database called `sales` with a DN of `dn:cn=sales,cn=OracleContext,o=example,c=us`, as shown in Figure 3–7. If the client is configured with a default realm Oracle Context of `cn=OracleContext,o=example,c=us`, then the connect identifier can simply be `sales`.

Figure 3–7 Relative Naming



Note: The **JDBC OCI Driver** supports both relative and full-qualified naming. The **JDBC Thin Driver** supports fully-qualified naming only when the complete DN is used.

See Also:

- *Oracle Internet Directory Administrator's Guide* for additional information about how clients locate a directory
- *Oracle Database JDBC Developer's Guide and Reference* for additional information about JDBC drivers

Using an Entry's Fully-qualified Name

Consider the same directory structure as shown Figure 3–7, but with the client's Oracle home configured with a default realm Oracle Context of `cn=OracleContext,o=example,c=jp`.

Because the client is configured with a default Oracle Context that does not match the location of `sales` in the directory server, a connect string that uses `sales` does not work. Instead, the client must specifically identify the location of `sales`, which can be done in one of the following ways:

- The entry's complete DN can be used in the connect string, for example:

  ```
  CONNECT username@"cn=sales,cn=OracleContext,o=example,c=us"
  Enter password: password
  ```

  JDBC Thin drivers support fully-qualified naming only when the complete DN is used. However, many applications do not support the use of a DN.

- The entry can be referred to by a fully-qualified DNS-style name which is mapped by the Directory Naming adapter to the full x.500 DN of the database object in the LDAP directory, for example:

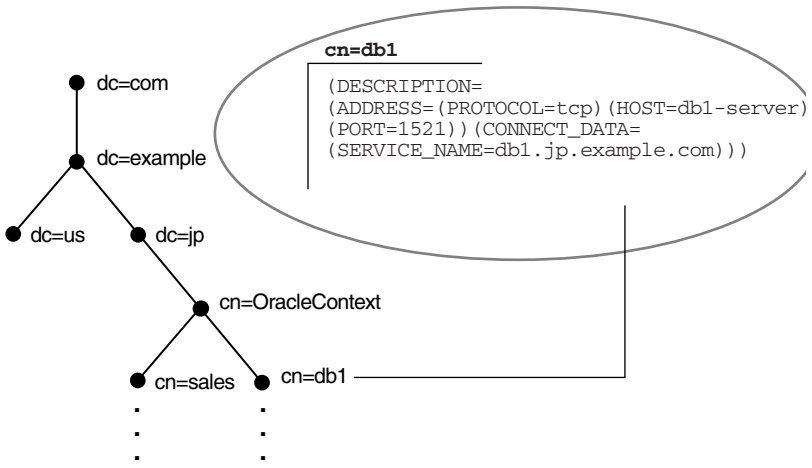
```
CONNECT username@sales.example.us
Enter password: password
```

Note: **JDBC OCI Driver**s support fully-qualified naming. **JDBC Thin Driver**s support fully-qualified naming only when the complete DN is used. See the *Oracle Database JDBC Developer's Guide and Reference* for additional information.

Understanding Oracle Context

In Figure 3–8, the entries `db1` and `sales` reside under `cn=OracleContext`. This entry is a special RDN called an **Oracle Context**. The entries under Oracle Context support various directory-enabled features, including directory naming.

Figure 3–8 Oracle Context in the DIT



During directory configuration, you set the default Oracle Context. Clients use this Oracle Context as the default location to look up connect identifiers in the directory server. With Oracle Internet Directory, an Oracle Context located at the root of the DIT, with DN of `dn:cn=OracleContext`, points to a default Oracle Context in an **identity management realm**. An identity management realm is a collection of identities governed by the same administrative policies. This Oracle Context is referred to as a **realm Oracle Context**. Unless configured to use another Oracle Context, clients use this realm-specific Oracle Context.

The default Oracle Context affects the connect string. For example, if a client must access the `db1` and `sales` entry frequently, then a reasonable default Oracle Context would be `dc=jp,dc=example,dc=com`. If a client directory entry does not match the directory entry where a service is located, then the client must specify an entry's fully-qualified name in the connect string, as described in "Client Connections Using Directory Naming" on page 3-11.

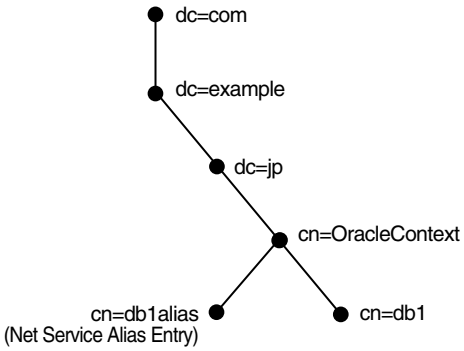
Note: The RDN `cn=OracleContext` does not have to be explicitly specified in the connect string.

See Also: *Oracle Internet Directory Administrator's Guide* for additional information about an identity management realm

Understanding Net Service Alias Entries

Directory naming enables you to create **net service alias** entries in addition to database service and net service name entries. A net service alias is an alternative name for a net service name or database service. A net service alias entry does not have connect descriptor information. Instead, it references the location of the entry for which it is an alias. When a client requests a directory lookup of a net service alias, the directory determines that the entry is a net service alias and completes the lookup as if it is the referenced entry. For example, in Figure 3–9, a net service alias of `db1alias` is created for a database service of `db1`. When `db1alias` is used to connect to a database service, as in `CONNECT username@db1alias`, it will actually resolve to and use the connect descriptor information for `db1`.

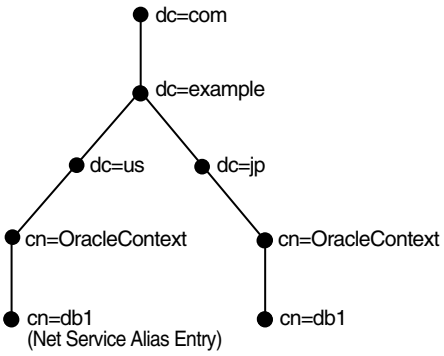
Figure 3–9 Net Service Alias db1alias in a Directory Server



There are several uses for using net service aliases. As shown in Figure 3–9, a net service alias can be useful as a way for clients to refer to a net service name by another name. Another use is to have a net service alias in one Oracle Context for a database service or net service name in a different Oracle Context. This enables a database service or net service name to be defined once in the directory server, but referred to by clients that use other Oracle Contexts.

In Figure 3–10, database service `db1` resides in `dc=jp,dc=example,dc=com`. A net service alias named `db1` is created in `dc=us,dc=example,dc=com`. This enables clients in both Japan and the United States to use the connect string `CONNECT username@db1` as opposed to clients in the Unites States needing to specify `CONNECT username@db1.jp.example.com`.

Figure 3–10 Net Service Alias db1 in a Directory Server



Who Can Add or Modify Entries in the Directory Server

Database Configuration Assistant creates database service entries during or after installation. You can then use **Oracle Enterprise Manager** or **Oracle Net Manager** to modify the Oracle Net attributes of the database service entries. You can also use these tools to create net service name and net service alias entries.

To use these configuration tools, a DIT structure containing a **root Oracle Context** and identity management realm must exist. The directory administrator creates this structure with Oracle Internet Directory Configuration Assistant. For some deployments, the directory administrator may need to create additional Oracle Contexts. Additional Oracle Contexts are usually used to subdivide large sites, or separate a production environment from a test environment.

Certain tools are used by certain groups, and you must be a member of the group to use the tools, as shown in the following:

- To create a database service entry with Database Configuration Assistant:
  - `OracleDBCreators` group (`cn=OracleDBCreators,cn=OracleContext...`)
  - `OracleContextAdmins` group (`cn=OracleContextAdmins,cn=Groups,cn=OracleContext...`)
- To create net service names or net service aliases with Oracle Net Manager:
  - `OracleNetAdmins` group (`cn=OracleNetAdmins,cn=OracleContext...`)
  - `OracleContextAdmins` group

The `OracleContextAdmins` group is a super-user group for Oracle Context. Members of the `OracleContextAdmins` group can add all supported types of entries to Oracle Context.

The directory user that created Oracle Context is automatically added to these groups. Other users can be added to these groups by the directory administrator.

See Also:

- "Configuring the Directory Naming Method" on page 8-12 for additional information about using Oracle Net Manager
- *Oracle Database Administrator's Guide* for additional information about how to register a database service with Database Configuration Assistant

Client Connections Using Directory Naming

Most clients need to perform name lookups in the directory server. To perform a lookup, the directory server must allow anonymous authentication. Directory servers usually do this by default.

To look up entries, a client must be able to find the directory server in which that entry resides. Clients locate a directory server in one of two ways:

- Dynamically using DNS. In this case, the directory server location information is stored and managed in a central domain name server. The client, at request processing time, retrieves this information from DNS.
- Statically in the directory server usage file, `ldap.ora`, created by Oracle Internet Directory Configuration Assistant and stored on the client host.

After a directory is found, clients are directed to the realm Oracle Context from the root Oracle Context.

Clients make connections to a database using connect identifiers in the same way they might use other naming methods. A connect identifier can be a database service, net service name, or net service alias. These can be referred to by their common names, or they can require additional directory location information. The default Oracle Context determines how the connect identifier must be specified, either by the entry's relative name or its fully-qualified name.

Considerations When Using Directory Servers

The following considerations should be reviewed when using directory servers:

- Performance Considerations
- Security Considerations
- Object Classes

Performance Considerations

Connect identifiers are stored in a directory server for all clients to access. Depending on the number of clients, there can be a significant load on a directory server.

During a connect identifier lookup, a name is searched under a specific Oracle Context. Users expect relatively quick performance so the database connect time is not affected. Because of the scope of the lookup, users may begin to notice slow connect times if lookups takes more than one second.

You can resolve performance problems by changing the network topology or implementing replication.

> **See Also:** Directory server vendor documentation for details on resolving performance issues

Security Considerations

Administrative clients can create and modify entries in the directory server, so security is essential. This section contains the following security-related topics:

- Authentication Methods
- Access Control Lists

Authentication Methods Clients use different methods of authentication depending upon the task that is to be performed.

- Clients that perform lookups for information in the directory server typically use anonymous authentication. In Oracle Database 11*g*, it is possible to configure clients to authenticate their LDAP bind during name lookup. Sites that need to protect their net service data or disable anonymous binds to the directory must configure their clients to use wallets to authenticate during name lookup. These clients require setting the following two parameters in the `sqlnet.ora` file:

  ```
  names.ldap_authenticate_bind = TRUE
  wallet_location = location_value
  ```

- Clients that administer the directory server entries authenticate with the directory server. Database Configuration Assistant or Oracle Net Manager may be used to add or modify the entries. Only authenticated users with proper privileges can modify entries. Use one of the following authentication methods:

  – Simple Authentication

    The client identifies itself to the directory server by means of a DN and a password. The server verifies that the DN and password sent by the client matches the DN and password stored in the directory server.

  – Strong Authentication

    The client identifies itself to the directory server by means of a public-key encryption available with Secure Sockets Layer (SSL). In public-key encryption, the sender of a message encrypts the message with the public key of the recipient. Upon delivery, the recipient decrypts the message using the recipient's private key.

Access Control Lists Authentication is used with **access control lists (ACLs)** to determine whether clients can read, modify, or add information in the directory server.

ACLs specify the following:

- The entries that the user can access.
- The authentication method used to access the entry.
- The access rights, or what the user can do with the object, such as read/write.

ACLs are established for a group of users. During Oracle Context creation, the `OracleDBCreators`, `OracleNetAdmins`, and `OracleContextAdmins` groups are created.

The user who creates Oracle Context using Oracle Net Configuration Assistant is automatically added as the first member of these groups.

Table 3–2 describes ACL requirements for these groups, anonymous users, and their relation to Oracle Net entries in the directory server.

Table 3–2 ACL Requirements for User Groups

Group	ACL Requirements
Anonymous users	All Oracle Net attributes and objects in the directory server have read access for the anonymous user. Read access of these objects for anonymous is also applied to Oracle Context. This enables anonymous users to browse directory naming entries contained within the `cn=OracleContext` RDN. This does not include objects used for enterprise user security. Oracle Net Configuration Assistant sets up this access during client installation.

Table 3–2 (Cont.) ACL Requirements for User Groups

Group	ACL Requirements
`OracleContextAdmins` group users	Members of `OracleContextAdmins` (`cn=OracleContextAdmins,cn=Groups,cn=OracleContext,...`) have create, modify, and read access to all directory naming objects. Oracle Net Configuration Assistant establishes these access rights for this group during Oracle Context creation. In addition to the Oracle Context creator, other users can be added to this group by the directory administrator using Oracle Enterprise Manager.
`OracleDBCreators` group users	Members of `OracleDBCreators` (`cn=OracleDBCreators,cn=OracleContext,...`) have create and read access to database service objects and attributes. Oracle Net Configuration Assistant establishes the access rights for this group during Oracle Context creation. In addition to the Oracle Context creator, other users can be added to this group by the directory administrator with Oracle Enterprise Manager.
`OracleNetAdmins` group users	Members of `OracleNetAdmins` (`cn=OracleNetAdmins,cn=OracleContext,...`) have create, modify, and read access to directory naming objects and attributes. Oracle Net Configuration Assistant establishes these access rights for this group during Oracle Context creation. In addition to the Oracle Context creator, other users can be added to this group by the directory administrator.

Situations in which a high degree of security is desired for lookup or read-access to Oracle Net Services name and related data, administrators can define additional read-access control for some or all of the data. Such ACL definitions can prevent anonymous users from reading the Oracle Net Services data. If read-access to Oracle Net Services data is restricted, then clients must use authenticated binds to do name lookup.

There are no predefined groups or procedures for the Oracle configuration tools for defining read-access restrictions on this data, so administrators must use standard object management tools from their directory system to manually create any necessary groups and ACLs.

ACLs can be added to Oracle Net Services objects using `ldapmodify` and an LDIF-format file. The following example restricts all access for user, `cn=user1`:

```
dn: cn=sales,cn=oraclecontext,dc=example,dc=com
replace: orclentrylevelaci
orclentrylevelaci: access to attr=(*)
 by dn="cn=user1" (noread,nosearch,nowrite,nocompare)
```

The preceding example illustrates the basic form of an ACL for a single object. This approach is not necessarily the best way to define access because access definitions for objects are complex and may involve security properties which are inherited from parent nodes in the DIT. Oracle recommends that administrators refer to the documentation for their directory systems, and integrate access management for Oracle Net Services objects into a directory-wide policy and security implementation.

For Oracle Internet Directory directories, `oidadmin` has functionality to create users, groups, and also define ACLs for objects and general directory security.

See Also:

- Documentation from your directory server vendor for information about how to set ACLs on directory entry
- "Authentication Methods" on page 3-11 for information about configuring clients to use authenticated binds for name lookup
- *Oracle Database Advanced Security Administrator's Guide* for additional information about the `OracleDBCreators` group
- "About the OracleNetAdmins Group" on page 7-5 for information about adding users to the `OracleNetAdmins` group
- *Oracle Internet Directory Administrator's Guide*

Object Classes

Directories must be populated with the correct version of the Oracle schema before Oracle Context, a database service or net service name entry can be created. The Oracle schema defines the type of objects, called **object class**es, that can be stored in the directory server and their attributes. Table 3–3 lists the object classes for database service, net service name, and net service alias entries.

Table 3–3 Oracle Net Services LDAP Main Object Classes

Object Class	Description
`orclDbServer`	Defines the attributes for database service entries
`orclNetService`	Defines the attributes for net service name entries
`orclNetServiceAlias`	Defines the attributes for net service alias entries

Table 3–4 lists the object classes used by `orclDbServer`, `orclNetService`, and `orclNetServiceAlias`.

Table 3–4 Oracle Net Services LDAP Derived Object Classes

Object Class	Description
`orclNetAddress`	Defines a listener protocol address
`orclNetAddressList`	Defines a list of addresses
`orclNetDescription`	Specifies a connect descriptor containing the protocol address of the database and the connect information to the service
`orclNetDescriptionList`	Defines a list of connect descriptors

These object classes use attributes that specify the contents of connect descriptors.

See Also: *Oracle Database Net Services Reference* for additional information about these object classes and their attributes

Limitations of Directory Naming Support with Microsoft Active Directory

In addition to Oracle Internet Directory, directory naming support is also provided with **Microsoft Active Directory** with the following limitations:

- Oracle provides support for Microsoft Active Directory only on Microsoft Windows operating systems. Therefore, client computers and the database server

must run on Microsoft Windows operating systems to access or create entries in Microsoft Active Directory.

- The following features are not supported by Microsoft Active Directory:
  - Multiple Oracle Contexts

    Microsoft Active Directory can support only one Oracle Context.
  - Net service aliases

    You cannot create net service aliases in Microsoft Active Directory. However, you can create net service names.
  - Automatic client discovery of directory servers for clients

    You must statically configure directory server usage on the clients. The Oracle Internet Directory Configuration will not provide directory server usage for Microsoft Active Directory. You must use **Oracle Net Configuration Assistant**.

    See Also: Microsoft Active Directory documentation for Microsoft-specific information

4

Understanding the Communication Layers

The primary function of Oracle Net is to establish and maintain connections between a client application and an Oracle database server. Oracle Net is comprised of several communication layers that enable clients and database servers to share, modify, and manipulate data.

This chapter contains the following topics:

- Understanding Oracle Net Stack Communication for Client/Server Applications
- Using Oracle Net Stack Communication for Java Applications
- Using Oracle Net Stack Communication for Web Clients
- Understanding Oracle Protocol Support Layer

> **See Also:** Chapter 1, "Introducing Oracle Net Services" for an introductory level overview of Oracle Net architecture

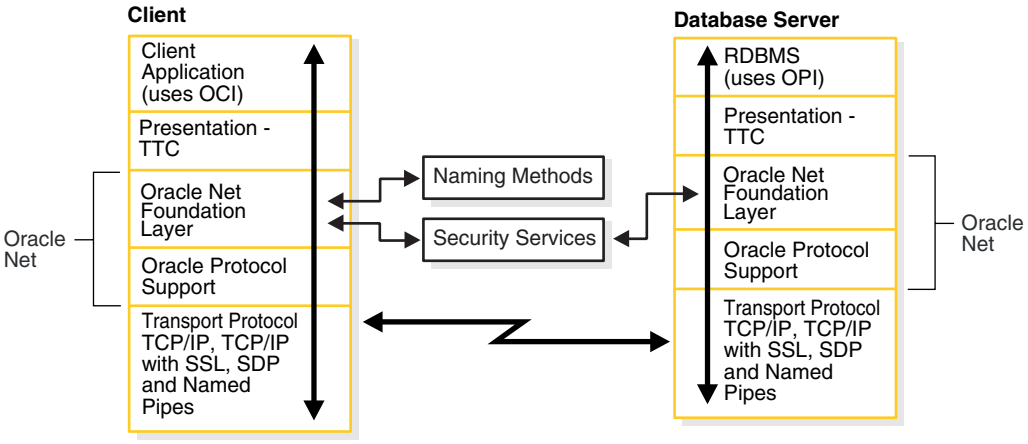
Understanding Oracle Net Stack Communication for Client/Server Applications

A database server is the Oracle software managing a database, and a client is an application that requests information from a server. The way the client and server communicate is known as the client/server stack.

Information passed from a client application sent by the client communication stack across a network protocol is received by a similar communications stack on the database server side. The process flow on the database server side is the reverse of the process flow on the client side, with information ascending through the communication layers.

Figure 4–1 illustrates the various layers on the client and on the database server after a connection has been established.

Figure 4–1 Layers Used in a Client/Server Application Connection

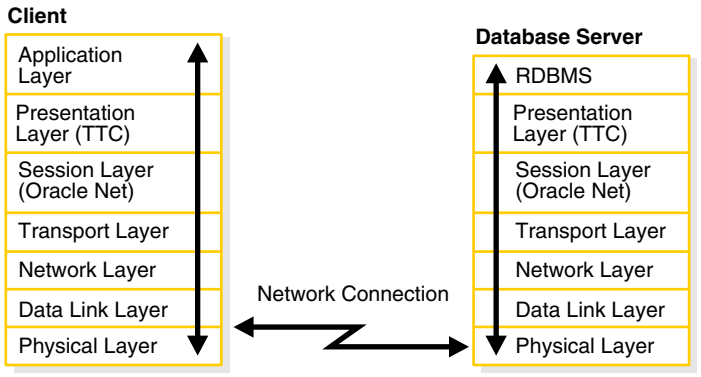


This communication architecture is based on the **Open Systems Interconnection (OSI)** model. In the OSI model, communication between separate computers occurs in a stack-like fashion with information passing from one node to the other through several layers of code, including:

1. Physical layer
2. Data link layer
3. Network layer
4. Transport layer
5. Session layer
6. Presentation layer
7. Application layer

Figure 4–2 shows how Oracle Net software consisting of the Oracle Net Foundation layer and Oracle Protocol Support fits into the session layer of the OSI model.

Figure 4–2 OSI Communication Layers



See Also: The following URL to learn about the OSI stack:

`http://www.ietf.org`

This section contains the following topics:

About the Client Communication Stack

The client communication stack includes the following:

Client Application Layer

During a session with the database, the client uses **Oracle Call Interface (OCI)** to interact with the database server. OCI is a software component that provides an interface between the application and SQL.

See Also: *Oracle Call Interface Programmer's Guide*

Presentation Layer

Character set differences can occur if the client and database server run on different operating systems. The presentation layer resolves any differences. It is optimized for each connection to perform conversion when required.

The presentation layer used by client/server applications is **Two-Task Common (TTC)**. TTC provides character set and data type conversion between different character sets or formats on the client and database server. At the time of initial connection, TTC is responsible for evaluating differences in internal data and character set representations and determining whether conversions are required for the two computers to communicate.

Oracle Net Foundation Layer

The Oracle Net foundation layer is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. The Oracle Net foundation layer can perform these tasks because of **Transparent Network Substrate (TNS)** technology. TNS provides a single, common interface for all industry-standard OSI transport and network layer protocols. TNS enables peer-to-peer application connectivity, where two or more computers can communicate with each other directly, without the need for any intermediary devices.

On the client side, the Oracle Net foundation layer receives client application requests and resolves all generic computer-level connectivity issues, such as:

- The location of the database server or destination
- How many protocols are involved in the connection
- How to handle interrupts between client and database server based on the capabilities of each

On the server side, the Oracle Net foundation layer performs the same tasks as it does on the client side. It also works with the listener to receive incoming connection requests.

In addition to establishing and maintaining connections, the Oracle Net foundation layer communicates with naming methods to resolve names and uses security services to ensure secure connections.

Oracle Protocol Support Layer

Oracle protocol support layer is positioned at the lowest layer of the Oracle Net foundation layer. It is responsible for mapping TNS functionality to industry-standard protocols used in the client/server connection. This layer supports the following network protocols:

- TCP/IP (**IPv4** and **IPv6**)
- TCP/IP with SSL
- Named Pipes
- Sockets Directory Protocol (SDP)

All Oracle software in the client/server connection process requires an existing network protocol stack to establish the computer-level connection between the two computers for the transport layer. The network protocol is responsible for transporting data from the client computer to the database server computer, at which point the data is passed to the server-side Oracle protocol support layer.

> **See Also:** "Understanding Oracle Protocol Support Layer" on page 4-5 for descriptions of the protocols

About the Server Communication Stack

The server communication stack uses the same layers as the client stack with the exception that the database uses **Oracle Program Interface (OPI)**. For each statement sent from OCI, OPI provides a response. For example, an OCI request to fetch 25 rows would elicit an OPI response to return the 25 rows after they have been fetched.

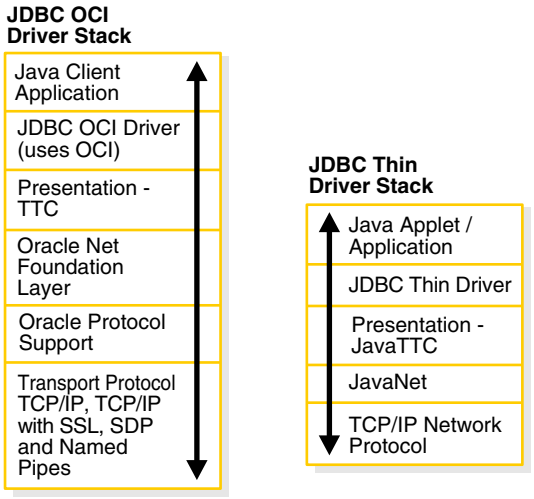
Using Oracle Net Stack Communication for Java Applications

The Oracle **Java Database Connectivity (JDBC) Driver**s provide Java applications access to an Oracle database. Oracle offers two JDBC drivers.

- **JDBC OCI Driver** driver is a type 2 JDBC driver which is used by client/server Java applications. The JDBC OCI driver uses a communication stack similar to a standard client/server communication stack. The JDBC OCI driver converts JDBC invocations to calls to OCI which are then sent over Oracle Net to the Oracle database server.
- **JDBC Thin Driver** is a type 4 driver which is used by Java applets. The JDBC Thin Driver establishes a direct connection to the Oracle database server over Java sockets. The JDBC Thin driver uses a Java implementation of the Oracle Net foundation layer called JavaNet and a Java implementation of TTC called JavaTTC to access the database.

Figure 4–3 shows the stack communication layers used by JDBC drivers.

Figure 4–3 Layers Used for Java-Client Applications



Note: SDP is not supported with JDBC Thin Driver stack.

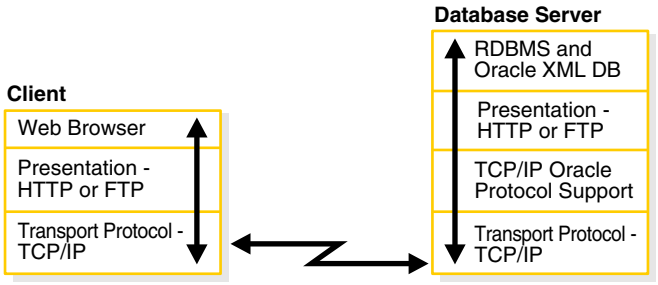
See Also: *Oracle Database JDBC Developer's Guide and Reference*

Using Oracle Net Stack Communication for Web Clients

The Oracle database server supports many other implementations for the presentation layer that can be used for Web clients accessing features inside the database in addition to TTC. The listener facilitates this by supporting any presentation implementation requested by the database.

Figure 4–4 shows the stack communication layers used in an HTTP or FTP connection to **Oracle XML DB** in the Oracle database instance. WebDAV connections use the same stack communication layers as HTTP and FTP.

Figure 4–4 Layers Used in Web Client Connections



See Also: *Oracle XML DB Developer's Guide*

Understanding Oracle Protocol Support Layer

A network protocol is responsible for transporting data from the client computer to the database server computer. This section describes the protocols used by the Oracle Protocol Support layer of the Oracle Net communication stack. It contains the following topics:

- TCP/IP Protocol
- TCP/IP with SSL Protocol
- Named Pipes Protocol
- Sockets Directory Protocol (SDP)

> **See Also:** *Oracle Database Net Services Reference* for information about the settings for each protocol

TCP/IP Protocol

Transmission Control Protocol/Internet Protocol (TCP/IP) is the standard communication protocol suite used for client/server communication over a network. TCP is the transport protocol that manages the exchange of data between hosts. IP is a network layer protocol for packet-switched networks.

Oracle Net supports IP in two versions: IP version 4 (IPv4) and IP version 6 (IPv6). IPv6 addresses the shortcomings of the currently used IPv4. The primary benefit of IPv6 is a large address space derived from the use of 128-bit addresses.

> **See Also:** `http://tools.ietf.org/html/rfc2460` for the IPv6 specification

IPv6 Address Notation

Oracle Database supports the standard IPv6 address notations specified by RFC 2732. A 128-bit IP address is generally represented as 8 groups of 4 hexadecimal digits, with the colon (`:`) symbol as the group separator. For example, the following address is in a valid IPv6 format:

```
2001:0DB8:0000:0000:0000:0000:200C:417A
```

Each hexadecimal digit in the address represents 4 bits, so each group in the address represents 16 bits. The following addresses represent the first and last hosts in the `2001:0DB8:0000:0000` subnet:

```
2001:0DB8:0000:0000:0000:0000:0000:0000
2001:0DB8:0000:0000:FFFF:FFFF:FFFF:FFFF
```

In shorthand notation, consecutive zero fields can be compressed with a double colon (`::`) separator, as shown in the following equivalent notations:

```
2001:0DB8:0:0::200C:417A
2001:0DB8::200C:417A
2001:DB8::200C:417A
```

> **See Also:**
>
> - `http://www.ietf.org/rfc/rfc2732.txt` for RFC 2732, and information about notational representation
> - `http://www.ietf.org/rfc/rfc3513.txt` for RFC 3513, and information about proper IPv6 addressing

CIDR Notation Classless Inter-Domain Routing (CIDR) is a method of grouping IP addresses into subnets that are independent of the value of the addresses. Classless routing was designed to overcome the exhaustion of address space in the IP class system and the unmanageable growth in the size of routing tables.

CIDR denotes a network by the first address in the network and the size in bits of the network prefix in decimal, separated with a slash (/). For example, `2001:0DB8::/32` indicates that the first 32 bits of the address identify the network, whereas the remaining bits identify the hosts in the network.

CIDR uses an analogous notation for IPv4 address. For example, in the notation `192.168.2.1/24` the first 24 bits of the address represent the network prefix. The `DBMS_NETWORK_ACL_ADMIN` package, which provides an API to manage access control lists, supports CIDR notation for both IPv6 and IPv4 addresses and subnets.

See Also:

- *Oracle Database PL/SQL Packages and Types Reference* to learn about `DBMS_NETWORK_ACL_ADMIN`
- `http://tools.ietf.org/html/rfc4632` for RFC 4632

IPv6 Addresses in URLs In URLs, IPv6 addresses are enclosed by the left bracket ([) and right bracket (]) characters. For example, the IPv6 address `[2001:0DB8:0:0:8:800:200C:417A]` forms part of the following URLs:

```
http://[2001:0DB8:0:0:8:800:200C:417A]
http://[2001:0DB8:0:0:8:800:200C:417A]:80/index.html
```

IPv4-Mapped Addresses IPv4-mapped addresses are a subclass of IPv6 addresses in which the following conditions are true:

- The first 80 bits are set to 0 in the standard IPv6 notation
- The second 16 bits are set to 1 in the standard IPv6 notation
- The last 32 bits are in IPv4 notation

An IPv4-mapped address can represent the addresses of IPv4-only nodes as IPv6 addresses.

Example 4–1 shows the same IP address in different notations. The first address uses standard IPv6 notation. The second address is an IPv4-mapped address in which the last 32 bits use dotted-decimal IPv4 notation. The last address uses a shorthand notation to compress the consecutive zero fields.

Example 4–1 IPv4-Mapped Address

```
0000:0000:0000:0000:0000:FFFF:C0A8:0226
0000:0000:0000:0000:0000:FFFF:192.168.2.38
::FFFF:192.168.2.38
```

See Also: `http://tools.ietf.org/html/rfc4942` for security consideration relating to the use of IPv4-mapped addresses

IPv6 Interface and Address Configurations

A host may have different IPv4 and IPv6 interface configurations. The following configurations are possible for a host:

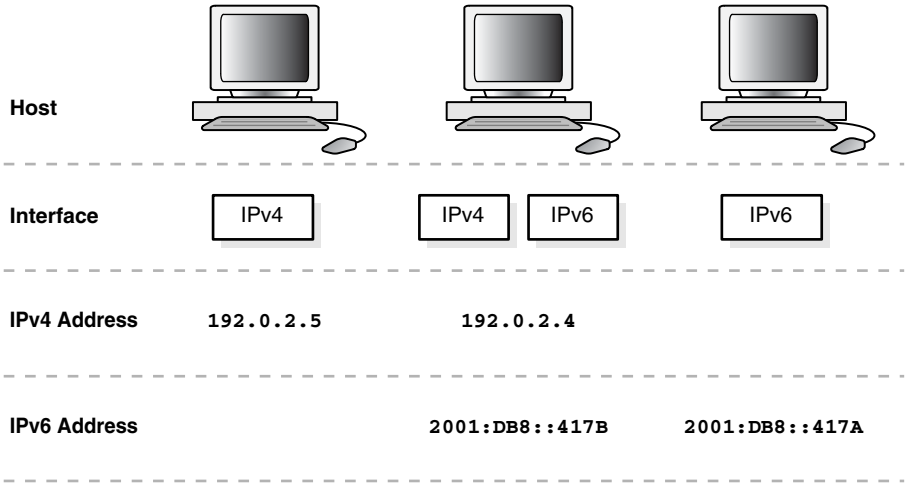
- Only an IPv4 interface, in which case the host is an IP4-only host.
- Only an IPv6 interface, in which case the host is an IPv6-only host.
- Both an IPv4 and IPv6 interface, in which case the host is a dual-stack host.

A single host may also use different types of IP address. For example, a domain name server may associate a dual-stack host both an IPv4 and an IPv6 address or only an IPv6 address. The IP address configurations that are not supported are the following:

- An IPv4-only host cannot use an IPv6 address.
- An IPv6-only host cannot use an IPv4 address.

Figure 4–5 shows possible host and interface configurations. The dual-stack host in the center of the diagram can communicate with IPv4 hosts over IPv4 and with IPv6 hosts over IPv6.

Figure 4–5 Supported Host and Interface Configurations



IPv6 Network Connectivity

The network connectivity of a host refers to its ability to communicate with another host over a network. For example, if a dual-stack client must communicate with an IPv6-only server, then the network and router must make end-to-end communication between these hosts possible.

A client or server host is IPv6-capable if it meets the following criteria:

- It has a configured IPv6 interface.
- It can connect to other hosts using the IPv6 protocol.

The IPv6 capability of a host is partially dependent on the network and partially dependent on its interface and address configuration. Figure 4–6 shows the possibilities for connectivity in a client/server network. For example, an IPv4-only host can connect to either an IPv4-only or dual-stack server, but not an IPv6-only server. Both dedicated and shared server modes are supported.

Figure 4–6 Client/Server Connectivity

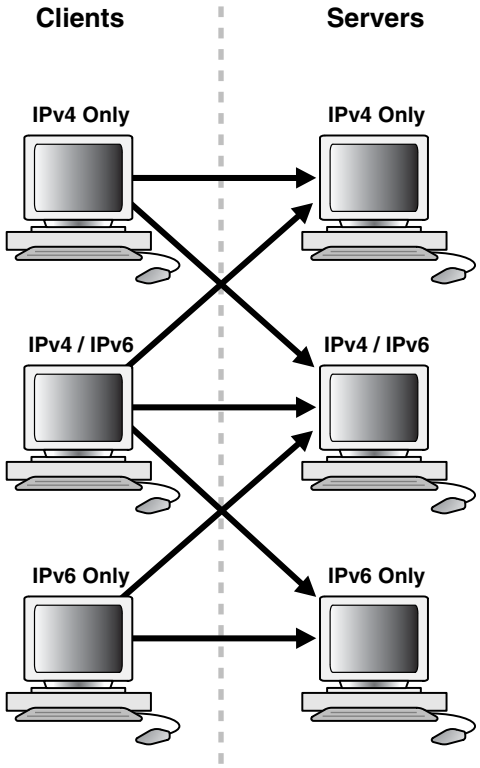


Table 4–1 summarizes the IP protocols used for client/server connectivity with different host and network configurations.

Table 4–1 Supported Host and Network Configurations

Client	IPv4-Only Server	Dual-Stack Server	IPv6-Only Server
IPv4-Only Client	Supported (v4)	Supported (v4)	Not Supported
Dual-Stack Client	Supported (v4)	Supported (v4, v6)	Supported (v6)
IPv6-Only Client	Not Supported	Supported (v6)	Supported

See Also:

- "About IPv6 Addresses in Connect Descriptors" on page 2-6
- "Configuring Listening Protocol Addresses" on page 9-4
- "Configuring Oracle Connection Manager as a Bridge for IPv4 and IPv6" on page 10-7

IPv6 Support in Oracle Database 11*g*

Components in this release of Oracle Database 11*g* support IPv6 in the configurations described in "IPv6 Network Connectivity", with the following exceptions:

- Oracle Real Application Clusters (Oracle RAC) and Oracle Clusterware
- Oracle Fail Safe

See Also: The documentation for each Oracle Database component for specific information about its support for IPv6

TCP/IP with SSL Protocol

The TCP/IP with **Secure Sockets Layer (SSL)** protocol enables an Oracle application on a client to communicate with remote databases through TCP/IP and SSL. **Oracle Advanced Security** is required to use TCP/IP with SSL.

SSL stores authentication data, such as certificates and private keys, in an Oracle Wallet. When the client initiates a connection to the database server, SSL performs a handshake between the two using the certificate. During the handshake, the following processes occur:

- The client and database server negotiate a cipher suite made up a set of authentication, encryption, and data integrity types to apply to the messages they exchange.
- Depending on its configuration, the database server sends its certificate to the client in a message encrypted with the client's public key. The database server may also send a request for the client's certificate in the same message. The client decrypts this message by using its own private key, then verifies that the database server's certificate bears the certificate authority's signature.
- If required, the client may send the user's certificate to the database server. The certificate ensures that the user's information is correct and that the public key actually belongs to that user.

The database checks the user certificate to verify that it bears the signature of the certificate authority.

See Also: *Oracle Database Advanced Security Administrator's Guide*

Named Pipes Protocol

Named Pipes is specifically designed for Microsoft Windows LAN environments. The **Named Pipes protocol** is a high-level interface providing interprocess communications between clients and database servers using distributed applications. One server-side process creates a named pipe, and the client-side process opens it by name. What one side writes, the other can read.

If a remote Oracle database is running on a host system that supports network communication using Named Pipes, then Oracle Net enables applications on a client to communicate with the Oracle database using Named Pipes.

Sockets Directory Protocol (SDP)

The Sockets Directory Protocol (SDP) is an industry-standard wire protocol between InfiniBand network peers. When used over an InfiniBand network, SDP reduces TCP/IP overhead by eliminating intermediate replication of data and transferring most of the messaging burden away from the CPU and onto the network hardware.

5

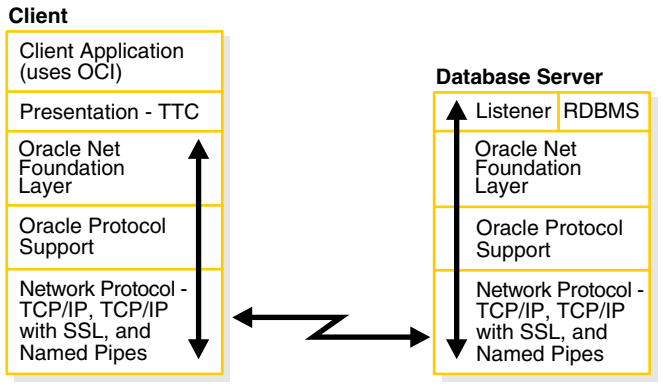
Understanding Oracle Net Architecture

The Oracle Net listener is an application positioned on top of the Oracle Net foundation layer. The database receives an initial connection from a client application through the listener.

The listener brokers client requests, handing off the requests to the Oracle database server. Every time a client requests a network session with a database, the listener receives the initial request.

Figure 5–1 illustrates the various layers on the client and database during an initial connection. As shown in the diagram, the listener is at the top layer of the server-side network stack.

Figure 5–1 Layers Used in an Initial Connection



Note: SDP is supported but is not represented in Figure 5–1.

This chapter contains the following topics:

- About Service Registration
- About the Listener and Connection Requests
- About Oracle Restart
- About Blocked Connection Requests
- Understanding Database Server Process Architecture
- Understanding Oracle Connection Manager Architecture
- Complete Architecture

About Service Registration

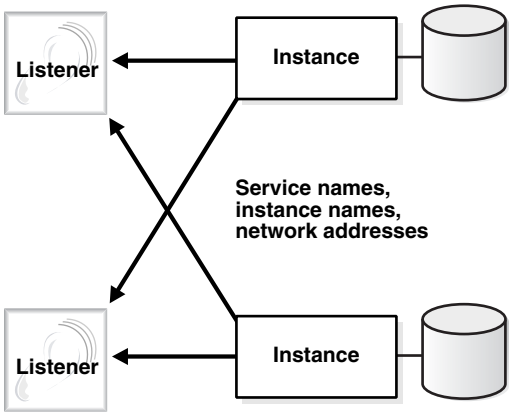
The listener determines whether a database service and its service handlers are available through **service registration**. During registration, the **PMON process** provides the listener with information about the following:

- Names of the database services provided by the database
- Name of the database **instance** associated with the services and its current and maximum load
- Service handlers (**dispatcher**s and dedicated servers) available for the instance, including their type, protocol addresses, and current and maximum load

The preceding information enables the listener to direct a client request appropriately.

Figure 5–2 shows two database instances registering information with two listeners. The figure does not represent all the information that can be registered. For example, listening endpoints, such as the port numbers, can be dynamically registered with the listener.

Figure 5–2 Service Registration



If the listener is not running when an instance starts, then the process monitor (PMON) cannot register the service information. PMON attempts to connect to the listener periodically, but it may take up to 60 seconds before PMON registers with the listener after it has been started. To initiate service registration immediately after the listener is started, use the SQL statement `ALTER SYSTEM REGISTER`. This statement is especially useful in high availability configurations.

About the Listener and Connection Requests

Each listener is configured with one or more **protocol address**es that specify its listening endpoints. The protocol address defines the protocol the listener listens on and any other protocol specific information. For example, the listener could be configured to listen at the following protocol address:

```
(DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)))
```

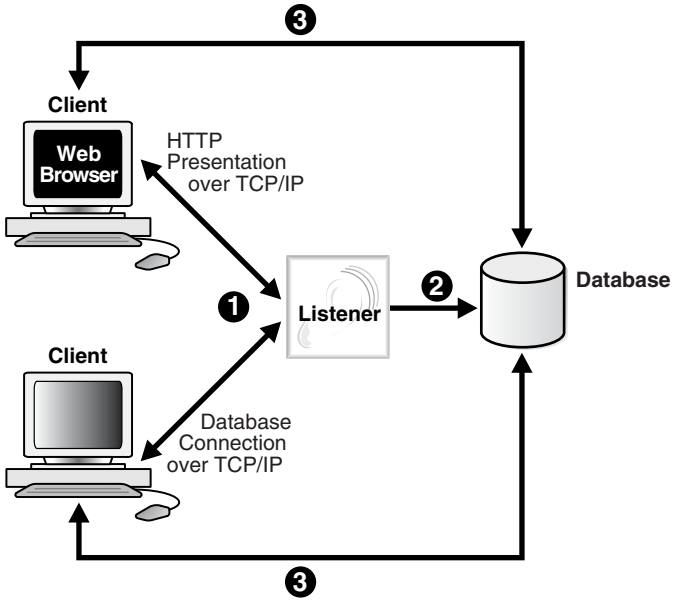
The preceding example shows a TCP/IP address that specifies the host of the listener (`sales-server`) and a port number (1521).

Clients configured with a protocol address can send connection requests to the listener. When a client request reaches the listener, it selects an appropriate **service handler** to service the request and forwards the request to the handler. A service handler is a **dispatcher** or a **dedicated server** process that acts as a connection point to a database.

Figure 5–3 illustrates the role of the listener during the establishment of a connection. The figure shows a browser making an HTTP connection and a client making a database connection.

1. The client sends a connection request to the listener.
2. The listener parses the client request and forwards it to the service handler for the database service requested.
3. The client connects to the database.

Figure 5–3 Listener Architecture



About Oracle Restart

The Oracle Restart feature enhances the availability of Oracle databases in a single-instance environment. Using the Server Control (SRVCTL) utility, you can add components such as the listener to an Oracle Restart configuration. The configuration enables the listener to start automatically when the listener fails or is not running.

When using Oracle Restart, note the following:

- Use the SRVCTL utility to start and stop the listener. Do not use the listener control, LSNRCTL, utility.
- Each listener must have a unique name.

See Also:

- "Managing a Listener in an Oracle Restart Configuration" on page 9-17
- *Oracle Database Administrator's Guide* to learn how to configure Oracle Restart

About Blocked Connection Requests

Blocked connection requests can occur when an incoming request occurs before the respective instance has been registered, or when a database is in restricted mode, such as when a shutdown of the database is in progress. If a database instance is in restricted mode, then PMON instructs the listener to block all connections to the instance. Clients attempting to connect receive one of the following errors:

- `ORA-12526: TNS:listener: all appropriate instances are in restricted mode`
- `ORA-12527: TNS:listener: all appropriate instances are in restricted mode or blocking new connections`
- `ORA-12528: TNS:listener: all appropriate instances are blocking new connections`

The ORA-12528 error occurs when a database instance is not yet registered with the listener.

See Also:

- *Oracle Database Error Messages* for additional information about these error messages
- *Oracle Database SQL Reference* for additional information about the `ALTER SYSTEM REGISTER` statement
- *Oracle XML DB Developer's Guide* for information about dynamically registering HTTP, FTP, and WebDAV listening endpoints

Understanding Database Server Process Architecture

Based on the service handler type registered with the listener, the listener forwards requests to either a shared server or dedicated server process. The shared server architecture enables a database server to allow many user processes to share server processes. In a dedicated server configuration, the listener starts a separate dedicated server process for each incoming client connection request dedicated to servicing the client.

This section contains the following topics:

- About Shared Server Processes
- About Dedicated Server Processes

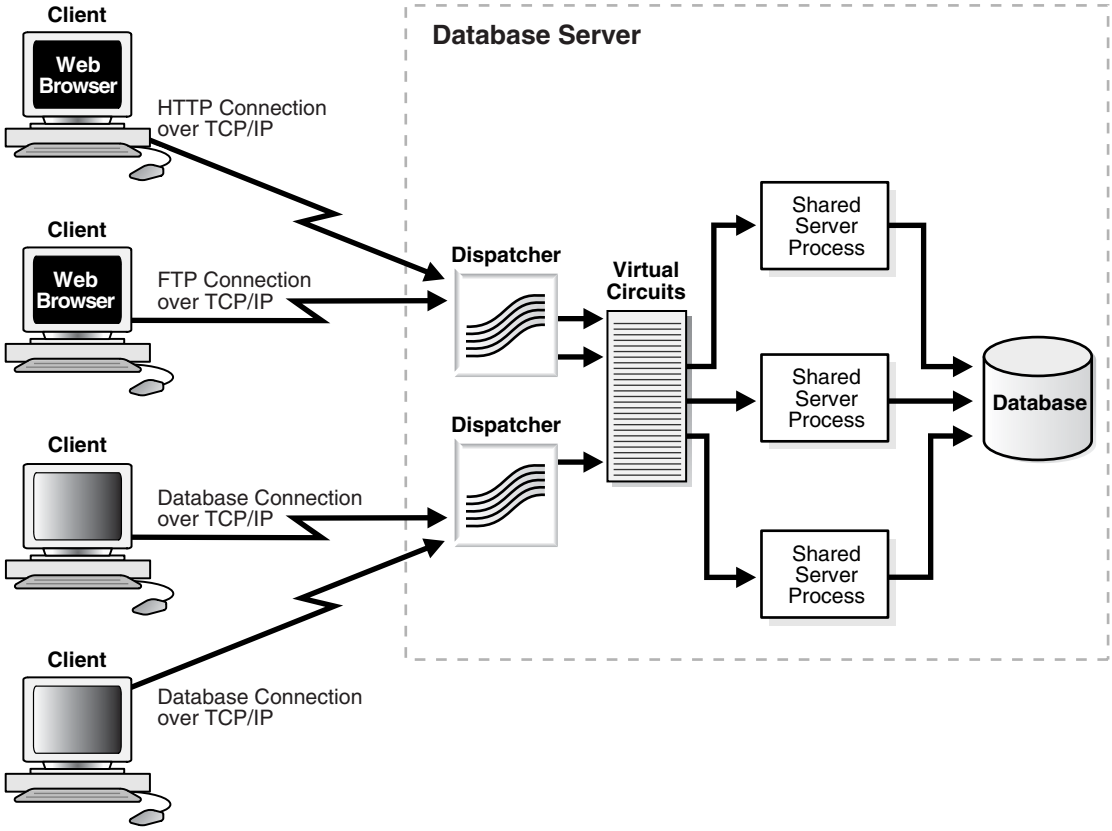
About Shared Server Processes

Shared server processes are used in the shared server architecture, as shown in Figure 5–4 on page 5-5. With shared server architectures, client processes ultimately connect to a dispatcher. The PMON process registers the location and load of the

dispatchers with the listener, enabling the listener to forward requests to the least loaded dispatcher. This registration process is not shown in the figure.

A dispatcher can support multiple client connections concurrently. Each client connection is bound to a **virtual circuit**. A virtual circuit is a piece of shared memory used by the dispatcher for client database connection requests and replies. The dispatcher places a virtual circuit on a common request queue when a request arrives. An idle shared server picks up the virtual circuit from the request queue, services the request, and relinquishes the virtual circuit before attempting to retrieve another virtual circuit from the request queue. Shared servers place all completed requests into a dispatcher's response queue. Each dispatcher has its own response queue in the SGA. This approach enables a small pool of server processes to serve a large number of clients.

Figure 5–4 Shared Server Architecture



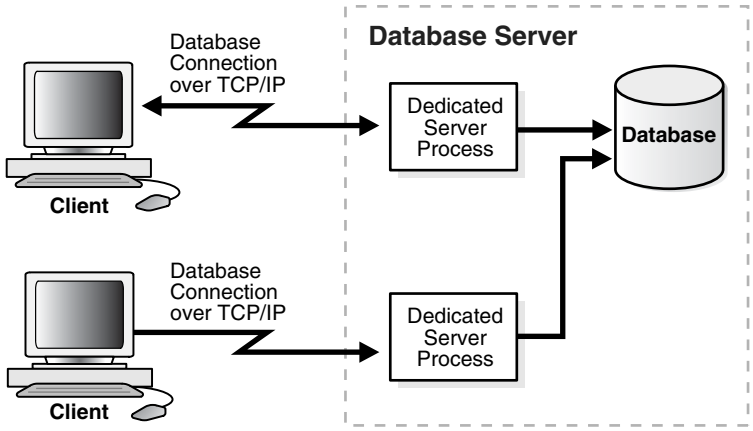
About Dedicated Server Processes

In a dedicated server architecture, each client process connects to a dedicated server process. The server process is not shared by any other client. Figure 5–5 on page 5-6 illustrates a dedicated server architecture.

PMON registers information about dedicated server processes with the listener. This enables the listener to start a dedicated server process when a client request arrives and forward the request to it.

Note: Dedicated server architectures do not support HTTP, FTP, or WebDAV clients. Only database clients are supported.

Figure 5–5 Dedicated Server Architecture



Understanding Oracle Connection Manager Architecture

Oracle Connection Manager is a gateway through which client connection requests are sent either to the next hop or directly to the database server. Clients who relay connection requests through an Oracle Connection Manager can take advantage of the session multiplexing and access control features configured on that Oracle Connection Manager. It carries no service information until a PMON process registers its services.

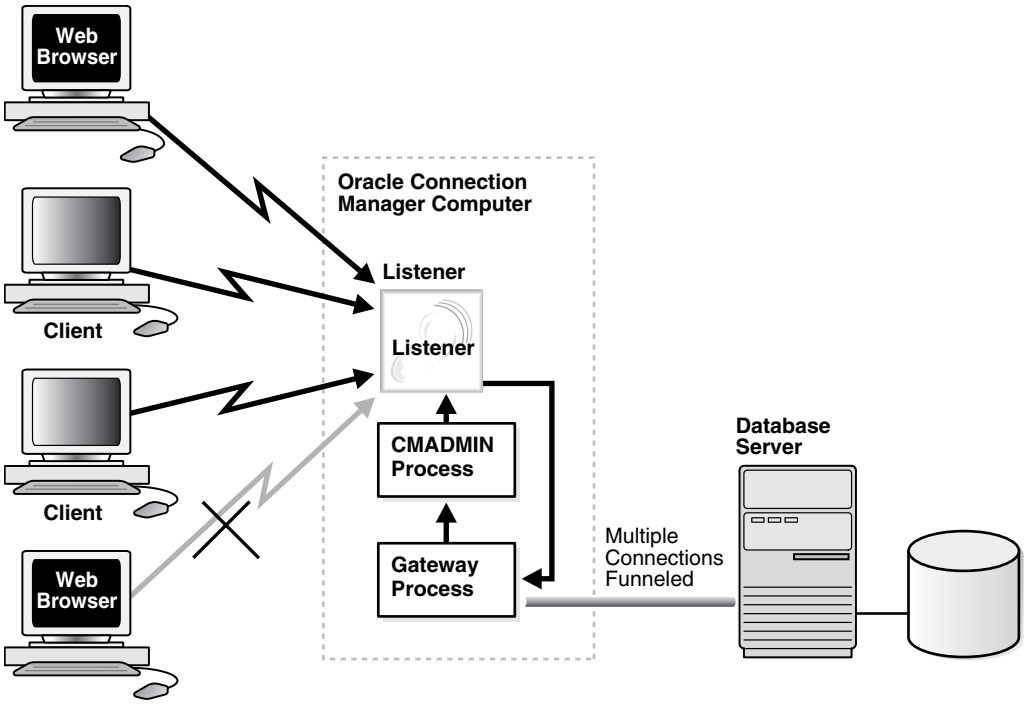
Oracle Connection Manager consists of three components:

- **listener**
- **CMGW (Oracle Connection Manager gateway)**
- **CMADMIN (Oracle Connection Manager Administration)**

The listener receives client connections and evaluates against a set of rules whether to deny or allow access. If it allows access, then the listener forwards a request to a gateway process, selecting the one with the fewest connections. The CMGW process, in turn, forwards the request to another Oracle Connection Manager or directly to the database server, relaying data until the connection terminates. If a connection to the server already exists, then the gateway multiplexes, or funnels, its connections through the existing connection. CMADMIN monitors the state of the gateway processes and the listener, shutting down or starting up processes as needed. In addition, it registers the location and load of the gateway processes with the listener, and it answers requests from the Oracle Connection Manager Control utility.

In Figure 5–6, the listener screens connection requests. A gateway process registers with the CMADMIN process. And the CMADMIN process registers with the listener. Finally, the listener forwards the connection requests to the gateway process. After receiving the three valid client connections, the gateway process multiplexes them through a single network protocol connection to the database. The fourth connection is denied when it is evaluated against the set of rules.

Figure 5–6 Oracle Connection Manager Architecture

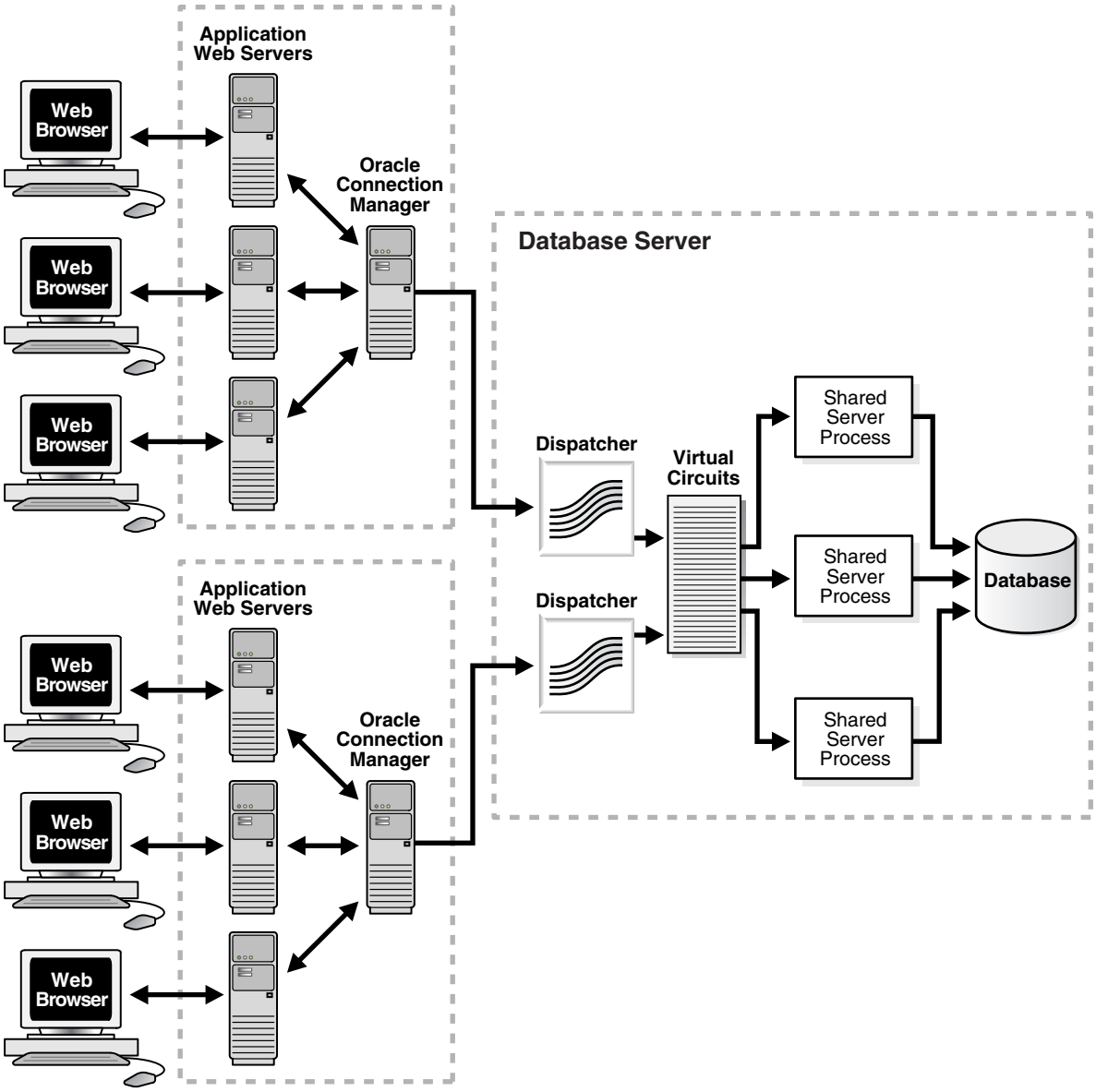


Complete Architecture

Oracle Net provides an architectural solution that allows for greater scalability in Internet and intranet environments.

Figure 5–7 shows how multiple connections to an Oracle database server are made more scalable with Oracle Connection Manager and a shared server architecture. Oracle Connection Manager is used to offload some of the network I/O of the application Web servers, and a shared server is used to serve more concurrent users.

Figure 5–7 Scalable Architectural Solutions



Part II

Configuration and Administration of Oracle Net Services

Part II describes how to set up and configure Oracle Net Services.

This part contains the following chapters:

6

Quick Start to Oracle Net Services

This chapter is designed to help novice users set up and test a simple but common network configuration, such as one between a client application and a database over a TCP/IP network.

This chapter contains the following topics:

Prerequisites for Establishing Connectivity

The tasks in this chapter show a TCP/IP connection between a client computer and a database server. The following conditions are assumed about the database server and client computers:

- Database server computer
  - It is running on a network that can access the client.
  - An Oracle database is installed.
  - TCP/IP protocol support is installed.
  - A listener is configured.
- Client computer
  - It is running on a network that can access the database server.
  - Oracle Client is installed.
  - TCP/IP protocol support is installed.

Confirming Network Availability

Before using Oracle Net to connect a client computer to a database server, confirm that the client computer can successfully communicate with the database server computer. Evaluating network connectivity can eliminate network-based errors.

To confirm network connectivity, do the following:

1. Confirm that the database server computer can communicate with itself with a **loopback test**.

   A loopback test is a connection from the database server back to itself. Many network protocols provide a means of testing network connections. The PING utility can be used with a TCP/IP network.

   In a TCP/IP network, each computer has a unique **IP address**. A name resolution service, such as **Domain Name System (DNS)**, can be used to map the IP address of a computer with its host name. If a name resolution service is not used, then the mapping is typically stored in a centrally maintained file called `hosts`. This file is located in the `/etc` directory on Linux and the `\windows\system32\drivers\etc` directory on Microsoft Windows. For example, an entry for a database server computer named `sales-server` may look like the following:

   ```
   #IP address of server     host name       alias
   192.168.2.203             sales-server    sales.us.example.com
   ```

   a. To confirm hardware connectivity, enter the following command at the command line:

      ```
      ping ip_address
      ```

      In the preceding command, *ip_address* is the IP address of the database server computer, such as the following:

      ```
      ping 192.168.2.203
      ```

   b. To confirm the DNS or host name is configured properly, enter the following command at the command line:

      ```
      ping host_name
      ```

      in the preceding command, *host_name* is the host name of the server.

   c. To test the TCP/IP setup for the server, enter the following command:

      ```
      ping 127.0.0.1
      ```

2. Verify the client computer can successfully communicate with the database server computer.

   This varies according to the network protocol. For TCP/IP, you can use PING, FTP or TELNET utilities.

   If the client computer cannot reach the server, then verify that the network cabling and network interface cards are correctly connected. Contact your network administrator to correct these problems.

Starting Oracle Net Listener and the Oracle Database Server

Oracle Net Listener and the Oracle Database server must be running in order for the database server to receive connections. To start Oracle Net Listener, do the following:

1. Start the listener with the Listener Control utility. From the command line, enter the following:

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

In the preceding command, *listener_name* is the name of the listener defined in the `listener.ora` file. It is not necessary to identify the listener if you are using the default name `LISTENER`.

A status message indicating that the listener has successfully started displays.

2. Start the database, as follows:

   a. Start SQL*Plus without connecting to the database using the following command:

      ```
      sqlplus /nolog
      ```

   b. Connect to the database as SYSDBA using the following command:

      ```
      SQL> CONNECT username as sysdba
      ```

      For example, `SYS` is a SYSDBA user. You will be prompted to enter a password.

      Note: For simplicity, this example does not perform the password management techniques that a deployed system normally uses. In a production environment, follow the Oracle Database password management guidelines, and disable any sample accounts. See *Oracle Database Security Guide* for password management guidelines and other security recommendations.

   c. Start the database using the following command:

      ```
      SQL> STARTUP database_name
      ```

      See Also: *Oracle Database Administrator's Guide* for additional information about starting the database

3. Confirm that database **service registration** with the listener has completed using the Listener Control utility using the following command:

   ```
   LSNRCTL> SERVICES [listener_name]
   ```

   The `SERVICES` command lists the services supported by the database, along with at least one available **service handler**. If the database service registration is not listed, then enter the following SQL command:

   ```
   SQL> ALTER SYSTEM REGISTER
   ```

   See Also: "Monitoring Services of a Listener" on page 9-20 for additional information about the `SERVICES` command

Starting Oracle Connection Manager

If you have installed Oracle Connection Manager, then start Oracle Connection Manager as follows:

1. Start Oracle Connection Manager Control utility (CMCTL) using the following commands:

   ```
   cmctl
   CMCTL> ADMINISTER [instance_name]
   ```

In the preceding commands, *instance_name* is the name of Oracle Connection Manager to administer. You can determine the name by viewing the `cman.ora` file. The file is located on the Oracle Connection Manager computer in the `ORACLE_HOME/network/admin` directory.

Oracle Connection Manager displays a status message indicating the name of the instance and informing you that the instance has not yet been started.

Note: If you do not provide an instance name as an argument, then provide Oracle Connection Manager with a fully qualified host name. This is the default. After you issue the `ADMINISTER` command, CMCTL displays the instance name as follows:

`CMAN_`*`fully_qualified_host_name`*

---

2. Start the Oracle Connection Manager that you have chosen to administer using the following command:

```
cmctl> STARTUP
```

Oracle Connection Manager confirms that the instance has been started, and provides status for the instance.

Note: On Microsoft Windows, Oracle Connection Manager can also be started through the Control Panel, as follows:

1. Select **Services** in the Control Panel.
2. Select the `Oracle`*`HOME_NAME`*`CMan` service, and then click **Start**.
3. In the Services window, click **Close**.

---

3. Exit from the Oracle Connection Manager Control utility using the following command:

```
cmctl> EXIT
```

Connecting to the Database

There are several methods to connect to an Oracle database. Table 6–1 lists syntax to connect to a database.

Table 6–1 Connecting to a Database

Type of Connection	Connection Syntax	Description

From the command line	The general form of connecting an application to a database server from the command line is:
	`tool username@connect_identifier`
	`Enter password: password`
	
	You will be prompted to enter your password which will be encrypted.
	For example:
	`SQLPLUS system@sales`
	`Enter password: password`
		Most Oracle tools can use the operating system command line to connect, and some provide alternatives.

From a login screen	`username@connect_identifier`	Some tools provide a logon screen as an alternative form to log on. A user can log on to a database server by identifying both the username and connect identifier in the username field of the tool logon screen, and entering the password as usual in the password field.

From a 3GL application	`exec sql connect :username identified by :password`
	
	In the preceding connection request, *:username* and *:password* are 3GL variables that can be set within the program either statically or by prompting the user. When connecting to a database server, the value of the *:username* variable is in the form:
	`username@net_service_name`
	
	The *:password* variable contains the password for the database account to which you are connecting.
		Applications written in 3GL, such as OCI and pre-compilers, are used by middle-tier and database application developers for direct database access from a client program.

From within SQL*Plus	`sqlplus /nolog`
	`SQL> CONNECT username@net_service_name`
	`Enter password: password`
	
	For example:
	`sqlplus /nolog`
	`SQL> CONNECT scott@serverx`
	`Enter password: password`
	In the preceding commands, *username* and *password* are the database user and password, and *net_service_name* is the net service name.
		Some Oracle tools have commands for database connections to allow an alternative username to be specified without leaving the tool.
		Other Oracle tools use slightly different methods specific to their function or interface. For example, Oracle CDE tools use logon buttons and a window with fields for the username, password, and remote database ID.

Using Easy Connect to Connect to a Database

After network connectivity has been verified as described in "Confirming Network Availability" on page 6-1, you can use the Easy Connect naming method to connect to the database. This naming method provides out-of-the-box TCP/IP connectivity to databases. It extends the functionality of the **host naming** method by enabling clients to connect to a database server with an optional port and service name in addition to the host name of the database. The following is the syntax to connect using Easy Connect:

```
CONNECT username/password@host[:port][/service_name][:server][/instance_name]
```

Note: In Oracle Call Interface documentation, *server* is referred to as `connect_type`.

If you have performed Oracle Database server installation in Typical mode, then the default service name used by the Oracle instance is `ORCL`, and the following Easy Connect syntax can be used to connect to that instance:

```
sqlplus /nolog
SQL> CONNECT username@"host/ORCL"
SQL> Enter password: password
```

NOTE: Starting with Oracle Database 10*g*, Oracle Database does not support the use of Oracle Names. Oracle Database 11*g* clients and databases cannot use Oracle Names, including those from an LDAP proxy, to resolve naming. Oracle9*i* clients can still use Oracle Names to resolve naming for an Oracle Database 11*g* database. However, customers are strongly encouraged to migrate to LDAP to take advantage of the new features of Oracle Database 11*g*.

See Also: "Using the Easy Connect Naming Method" on page 8-1 for additional information about this method

7

Managing Oracle Net Services

This chapter introduces the various administration tools of Oracle Net Services. It discusses the main administration applications, **Oracle Enterprise Manager** and **Oracle Net Manager**. It also introduces the command-line control utilities.

This chapter contains the following topics:

Using the User Interface Tools

Oracle Net Services provides tools to help you perform configuration and administrative tasks. This section contains the following topics:

Using Oracle Enterprise Manager to Configure Oracle Net Services

Oracle Enterprise Manager enables you to configure Oracle Net Services for any Oracle home across multiple file systems. It also provides common administration functions for listeners. Oracle Enterprise Manager provides an integrated environment for configuring and managing Oracle Net Services.

You can use Oracle Enterprise Manager to configure and administer the following from multiple Oracle homes:

- Listeners: Configure listeners to receive client connections.
- Naming: Define **connect identifier**s and map them to **connect descriptor**s to identify the network location of a service. Oracle Net Manager supports configuration of connect descriptors in local `tnsnames.ora` files or a centralized directory service.
- File Location: Specify the file location of the Oracle Net configuration files.

> **See Also:** Oracle Enterprise Manager documentation set and online Help for information about using Oracle Enterprise Manager

Accessing the Net Services Administration Page

To access the Net Services Administration page using Oracle Enterprise Manager:

1. From the Login to Database page, enter the database credentials, and then click **Login**.

   The Database page appears.

2. In the General section, click the **listener.**

   The Listener Home page appears.

3. In the Related Links section, click **Net Services Administration**.

   The Net Services Administration page appears.

From the Net Services Administration page, you can administer the listeners, naming methods, preferences, and so on. The administration procedures are described in other chapters of this book.

Using Oracle Net Manager to Configure Oracle Net Services

Oracle Net Manager enables you to configure Oracle Net Services for an Oracle home on a local client or server host.

You can use Oracle Net Manager to configure the following network components:

- Listeners: Create and configure listeners to receive client connections.
- Naming: Define **connect identifier**s and map them to **connect descriptor**s to identify the network location and identification of a service. Oracle Net Manager supports configuration of connect descriptors in local `tnsnames.ora` files or a centralized directory service.
- Naming Methods: Configure the ways connect identifiers are resolved to connect descriptors.
- Profiles: Configure preferences for enabling and configuring Oracle Net features on the client or server.

This section introduces the features of Oracle Net Manager. However, the primary documentation for using Oracle Net Manager is online Help. It contains the following topics:

- Starting Oracle Net Manager
- Navigating Oracle Net Manager
- Using Oracle Net Manager Wizards

Starting Oracle Net Manager

You can start Oracle Net Manager using the Oracle Enterprise Manager Console or as an independent application as follows:

- To start Oracle Net Manager from the Oracle Enterprise Manager console, select Service Management from the Tools menu, and then select Oracle Net Manager.
- To start Oracle Net Manager as standalone application, do the following:
  - On Linux, run `netmgr` from the `ORACLE_HOME/bin` directory.
  - On Microsoft Windows, select **Programs** from the Start menu, and then select Oracle - *HOME_NAME*. Next, select **Configuration and Migration Tools**, and then **Net Manager**.

Navigating Oracle Net Manager

The Oracle Net Manager interface includes a toolbar and various menu options, as well as property sheets for configuring network components.

The navigator pane provides a tree view of network objects and the objects they contain, organized in folder hierarchy. You can expand and contract the folders to monitor or manage objects such as connect identifiers, listeners, and profiles. Click an object to make changes to it.

Table 7–1 lists the main folders in the navigator pane.

Table 7–1 Oracle Net Manager Navigator Pane Folders

Folder	Description
Local	Displays networking elements configured in local configuration files: ■ Net service names in the `tnsnames.ora` file ■ Listeners in the `listener.ora` file ■ Profile in the `sqlnet.ora` file
Directory	Displays connect identifiers configured in a directory server

Using Oracle Net Manager Wizards

The Oracle Net Manager wizards provide step-by-step guidance for tasks. The wizards simplify complex tasks by guiding you through the tasks in manageable steps. The wizards are not intended to provide all configuration options. After you have completed a task with a wizard, use other components of Oracle Net Manager to modify the configuration.

The following topics describe the Oracle Net Manager wizards:

- Using the Net Service Name Wizard
- Using the Directory Server Migration Wizard

Using the Net Service Name Wizard The Net Service Name wizard guides you through creating a basic net service name in a directory server or a `tnsnames.ora` file.

The following procedure describes how to start the Net Service Name wizard to create net service names:

1. In the navigator pane, select **Directory** or **Local**, and then select **Service Naming**.
2. Click the plus sign (+) on the toolbar, or select **Create** from the Edit menu.

> **See Also:** Oracle Net Manager online help for detailed information about using the Net Service Name wizard to create a net service name

Using the Directory Server Migration Wizard If a `tnsnames.ora` file already exists, then its net service names can be exported to a directory server with the Directory Server Migration wizard.

The following procedure describes how to use the Directory Server Migration wizard:

1. Select **Directory** from the Command menu.
2. Select **Export Net Service Names** from the Oracle Net Manager menu.

See Also: "Exporting Local Naming Entries to a Directory Naming Server" on page 8-19

Deciding When to Use Oracle Enterprise Manager and Oracle Net Manager

In Oracle Database 11*g*, much of the functionality previously available only in Oracle Net Manager has been integrated with Oracle Enterprise Manager. Oracle Enterprise Manager provides the ability to manage configuration for multiple Oracle homes across multiple file systems. Oracle Net Manager only enables you to manage configuration for one Oracle home on a local host computer. Table 7–2 describes the key differences between the tools.

Table 7–2 Comparing Oracle Enterprise Manager and Oracle Net Manager

User Interface Tool	Features
Oracle Enterprise Manager	■ Configures the following features: - Local naming (`tnsnames.ora` files) - Directory naming - Listeners ■ Provides Oracle home support across multiple file system ■ Provides the ability to search and sort local and directory naming entries ■ Export directory naming entries to a `tnsnames.ora` file ■ Performs the following administrative tasks for a selected listener: - Show current status - Change status - Change tracing level settings - Change logging settings - Set connect-time failover and load balancing methods when there is more than one listener
Oracle Net Manager	■ Configures the following features: - Local naming (`tnsnames.ora` files) - Directory naming - Listeners - Profiles ■ Provides Oracle home support for single host ■ Sets connect-time failover and load balancing methods when there is more than one listener

Note: When Automatic Diagnostic Repository (ADR) is enabled, any changes to the tracing and logging settings using Oracle Enterprise Manager are ignored by the system.

Using Oracle Net Configuration Assistant to Configure Network Components

Oracle Net Configuration Assistant is provided to configure basic network components during installation, including:

- Listener names and protocol addresses

- Naming methods the client uses to resolve connect identifiers to connect descriptors
- Net service names in a `tnsnames.ora` file
- Directory server usage

Oracle Net Configuration Assistant runs automatically during software installation, as described in your Oracle installation guide. It can also be run after installation in standalone mode to configure naming methods, the listener, net service names in the `tnsnames.ora` file, and directory server usage.

To start Oracle Net Configuration Assistant do the following:

- On Linux and UNIX, run `netca` from the `ORACLE_HOME/bin` directory.
- On Microsoft Windows, select **Programs** from the Start menu, and then select **Oracle - *HOME_NAME***. Next, select **Configuration and Migration Tools**, and then **Oracle Net Configuration Assistant**.

See Also: Oracle Net Configuration Assistant online help

Table 7–3 describes the configuration options on the Oracle Net Configuration Assistant Welcome page:

Table 7–3 Oracle Net Configuration Assistant

Option	Description
Listener configuration	Create, modify, delete, or rename a listener.
Naming Methods configuration	Configure this computer to resolve connect identifiers to connect descriptor through one or more of following naming methods: ■ Local naming ■ Directory naming ■ Easy Connect naming ■ External naming
Local Net Service Name configuration	Create, modify, delete, rename, or test connectivity of a connect descriptor stored in a local `tnsnames.ora` file.
Directory Usage Configuration	Configure a directory server for directory-enabled features.

About the OracleNetAdmins Group

To use Oracle Net Manager, you must be a member of the `OracleNetAdmins` group or the `OracleContextAdmins` group. Oracle Net Configuration Assistant establishes these access rights for these groups during Oracle Context creation.

The `OracleNetAdmins` group is owned by itself. Members of the `OracleNetAdmins` group have create, modify, and read access to Oracle Net objects and attributes. The can also add or delete members in the group, and add or delete groups to be owners of the `OracleNetAdmins` group.

The `OracleContextAdmins` group is a super-user group for Oracle Context. Members of the OracleContextAdmins group can add all supported types of entries to Oracle Context.

This section contains the following topics:

- Adding Users To the OracleNetAdmins Group

- Removing Users From the OracleNetAdmins Group
- Changing Ownership of the OracleNetAdmins Group

Note: Members of the `OracleContextAdmins` groups can also add and delete members of the `OracleNetAdmins` group.

Adding Users To the OracleNetAdmins Group

To add a user to the `OracleNetAdmins` group with `ldapmodify`, do the following:

1. Create an LDIF file that specifies that you want to add a user to the `OracleNetAdmins` group.

   You can use the following sample LDIF file. Use the appropriate DN for `cn=OracleNetAdmins` and the user that you want to add.

   ```
   dn: cn=OracleNetAdmins,cn=OracleContext,...
   changetype: modify
   add: uniquemember
   uniquemember: DN of user being added to group
   ```

2. Enter the following syntax at the command line to refresh the file:

   ```
   $ ldapmodify -h directory_host -p port -D binddn -q -f ldif_file
   ```

   In the preceding command, *directory_host* is the directory server host, *port* is the listening TCP/IP port for the directory server, *binddn* is the directory administrator or user DN, and *ldif_file* is the input file name. If the port is not specified, then the default port of 389 is used. The `-q` option prompts for a single bind password to be entered.

Removing Users From the OracleNetAdmins Group

To remove a user from the `OracleNetAdmins` group with `ldapmodify`, do the following:

1. Create an LDIF file that specifies that you want to delete a user to the `OracleNetAdmins` group.

   You can use the following sample LDIF file. Enter the appropriate DN for `cn=OracleNetAdmins` and the user that you want to delete.

   ```
   dn: cn=OracleNetAdmins,cn=OracleContext,...
   changetype: modify
   delete: uniquemember
   uniquemember: DN of user being deleted from group
   ```

2. Enter the following `ldapmodify` syntax at the command line to delete the user:

   ```
   $ ldapmodify -h directory_host -p port -D binddn -q -f ldif_file
   ```

   In the preceding command, *directory_host* is the directory server host, *port* is the listening TCP/IP port for the directory server, *binddn* is the directory administrator or user DN, and *ldif_file* is the input file name. If the port is not specified, then the default port of 389 is used. The `-q` option prompts for a single bind password to be entered.

Changing Ownership of the OracleNetAdmins Group

By default, the owner of the `OracleNetAdmins` group is the `OracleNetAdmins` group itself. Any member of the `OracleNetAdmins` group can add or delete other members from the `OracleNetAdmins` group. If you prefer another group to add or delete `OracleNetAdmins` members, then you can change the owner attribute of the `OracleNetAdmins` group to another group.

The owner cannot be an individual user entry but must be a group entry, and the group entry is one comprised of the LDAP schema object classes `GroupOfUniqueNames` and `orclPriviledgeGroup`.

To add a group as an owner of an `OracleNetAdmins` group, do the following:

1. Create an **LDAP Data Interchange Format (LDIF)** file, as follows:

   a. Specify the group you want to add as an owner.

      You can use the following sample LDIF file. Enter the appropriate **distinguished name (DN)** for `cn=OracleNetAdmins` and the DN of the group that you want to add.

      ```
      dn: cn=OracleNetAdmins,cn=OracleContext,...
          changetype: modify
          add: owner
          owner: DN of group to add
      ```

      For example, the following LDIF syntax changes the ownership from the `OracleNetAdmins` group to another group named `cn=ExampleSecurityAdmins`. The group can be either inside or outside Oracle Context.

      ```
      dn: cn=OracleNetAdmins,cn=OracleContext,...
           changetype: modify
           add: owner
           owner: cn=ExampleSecurityAdmins
      ```

   b. Optionally, specify the group to delete as an owner.

      ```
      dn: cn=OracleNetAdmins,cn=OracleContext,...
          changetype: modify
          delete: owner
          owner: DN of group to delete
      ```

2. Enter the following syntax at the command line to refresh the file:

   ```
   $ ldapmodify -h directory_host -p port -D binddn -q -f ldif_file
   ```

   In the preceding command, *directory_host* is the directory server host, *port* is the listening TCP/IP port for the directory server, *binddn* is the directory administrator or user DN, and *ldif_file* is the input file name. If the port is not specified, then the default port of 389 is used. The `-q` option prompts for a single bind password to be entered.

Using Listener Control Utility to Administer the Listener

Oracle Net Services provides tools to help you start, stop, configure, and control each network component. The Listener Control utility enables you to administer the listener. The utility is started by the user that owns the Oracle installation, or a member of the designated group, and on the same machine where the listener is running. The basic syntax for this utility is as follows:

```
lsnrctl command [listener_name]
```

For example, the following command starts a listener named `lsnr`:

```
lsnrctl START lsnr
```

You can also issue Listener Control utility commands at the `LSNRCTL>` program prompt. To obtain the prompt, enter `lsnrctl` with no arguments at the operating system command line. When you run `lsnrctl`, the utility is started, and you can enter the necessary commands from the program prompt.

For example:

```
lsnrctl
LSNRCTL> START lsnr
```

See Also:

- "Customizing Oracle Net Listener Configuration" on page 9-3 for additional information about the listener
- *Oracle Database Net Services Reference* for additional information about the Listener Control utility

Performing Common Network Tasks

Network configuration and administration tasks are described throughout this guide. The following tables list the common tasks, the tools associated with them, and points you to the topic in the guide that describes the task:

- Table 7–4, " Configuring Directory Server for Oracle Net Usage"
- Table 7–5, " Configuring Naming Methods"
- Table 7–6, " Migrating to Directory Naming"
- Table 7–7, " Configuring Profiles"
- Table 7–8, " Configuring Listeners"
- Table 7–9, " Administering Listeners"
- Table 7–10, " Configuring Oracle Connection Manager"

Table 7–4 shows the tasks for configuring directory server for Oracle Net.

Table 7–4 Configuring Directory Server for Oracle Net Usage

Task	Tools to Perform Task	See Also
Configure directory server usage.	Oracle Internet Directory Configuration Assistant	*Oracle Internet Directory Administrator's Guide*

Table 7–4 (Cont.) Configuring Directory Server for Oracle Net Usage

Task	Tools to Perform Task	See Also
Add users to the `OracleNetAdmins` group.	ldapmodify	"About the OracleNetAdmins Group" on page 7-5
Authenticate with the directory.	Oracle Enterprise Manager Oracle Net Manager	Online Help in Oracle Enterprise Manager Choose **Directory** > **Service Naming** > **How To** > **Change the Oracle Context** in the online Help for Oracle Net Manager *Oracle Database Enterprise User Security Administrator's Guide*
Change Oracle Context.	Oracle Net Manager	Online Help in Oracle Enterprise Manager Choose **Directory** > **Service Naming** > **How To** > **Set Authentication Credentials** in the online Help for Oracle Net Manager

Table 7–5 shows the tasks for configuring naming methods.

Table 7–5 Configuring Naming Methods

Task	Tools to Perform Task	See Also
Configure the local naming method.	Oracle Enterprise Manager Oracle Net Manager Oracle Net Configuration Assistant	"Configuring the Local Naming Method" on page 8-6
Configure the directory naming method.	Oracle Enterprise Manager Oracle Net Manager	"Configuring the Directory Naming Method" on page 8-12
Configure the Easy Connect naming method.	Oracle Net Manager	"Using the Easy Connect Naming Method" on page 8-1
Configure external naming methods.	Oracle Net Manager	"Configuring External Naming Methods" on page 8-22

Table 7–6 shows the tasks for migrating to directory naming.

Table 7–6 Migrating to Directory Naming

Task	Tools to Perform Task	See Also
Export from `tnsnames.ora` files.	Oracle Enterprise Manager Oracle Net Manager	"Exporting Directory Naming Entries to a tnsnames.ora File" on page 8-22

Table 7–7 shows the tasks for configuring profiles.

Table 7–7 Configuring Profiles

Task	Tools to Perform Task	See Also
Prioritize naming methods.	Oracle Net Manager Oracle Net Configuration Assistant	"Prioritizing Naming Methods" on page 12-3
Configure a default domain that is automatically appended to any unqualified net service name.	Oracle Net Manager Oracle Net Configuration Assistant	"Specifying a Default Domain for Clients" on page 12-2
Route connection requests.	Oracle Net Manager Oracle Net Configuration Assistant	"Routing Connection Requests to a Process" on page 12-4
Configure access control.	Oracle Net Manager	"Configuring Database Access Control" on page 12-4
Configure an **authentication method** available with **Oracle Advanced Security**.	Oracle Net Manager	"Configuring Oracle Advanced Security" on page 12-9 Choose **Oracle Advanced Security** > **How To** in the online help *Oracle Database Advanced Security Administrator's Guide*
Configure connect request timeouts.	Manual Configuration	"Limiting Resource Consumption by Unauthorized Users" on page 14-7

Table 7–8 shows the tasks for configuring listeners.

Table 7–8 Configuring Listeners

Task	Tools to Perform Task	See Also
Configure listening protocol addresses.	Oracle Enterprise Manager Oracle Net Manager Oracle Net Configuration Assistant	"Configuring Listening Protocol Addresses" on page 9-4
Configure dynamic service registration.	Automatic	"Configuring Service Registration" on page 9-9
Configure static service registration.	Oracle Enterprise Manager Oracle Net Manager	"Configuring Static Service Information" on page 9-5
Configure password authentication.	Oracle Enterprise Manager Oracle Net Manager	"Configuring and Changing the Oracle Net Listener Password" on page 9-7
Configure connect request timeouts.	Manual Configuration	"Limiting Resource Consumption by Unauthorized Users" on page 14-7

Table 7–9 shows the tasks for administering listeners.

Table 7–9 Administering Listeners

Task	Tools to Perform Task	See Also
Start and stop listeners.	Listener Control Utility	"Starting and Stopping a Listener" on page 9-16
View registered information.	Listener Control Utility	"Monitoring Services of a Listener" on page 9-20

Table 7–10 shows the tasks for configuring Oracle Connection Manager.

Table 7–10 Configuring Oracle Connection Manager

Task	Tools to Perform Task	See Also
Configure session multiplexing.	Manual Configuration	"Enabling Session Multiplexing" on page 10-7
Configure access control.	Manual Configuration	"Enabling Access Control" on page 10-4

8

Configuring Naming Methods

This chapter describes how to configure connectivity information for client connections to the database server.

This chapter contains the following topics:

- Using the Easy Connect Naming Method
- Configuring the Local Naming Method
- Configuring the Directory Naming Method
- Creating Multiple Default Contexts in a Directory Naming Server
- Exporting Local Naming Entries to a Directory Naming Server
- Exporting Directory Naming Entries to a tnsnames.ora File
- Configuring External Naming Methods

> **See Also:** "Understanding Naming Methods" on page 2-12 for an overview of naming methods

Using the Easy Connect Naming Method

The Easy Connect naming method eliminates the need for service name lookup in the `tnsnames.ora` files for TCP/IP environments. In fact, no naming or directory system is required if you use this method.

This naming method provides out-of-the-box TCP/IP connectivity to databases. It extends the functionality of the **host naming** method by enabling clients to connect to a database server with an optional port and service name in addition to the host name of the database:

```
CONNECT username@[//]host[:port][/service_name][:server][/instance_name]
Enter password: password
```

The connect identifier converts to the following connect descriptor:

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=host)(PORT=port))
  (CONNECT_DATA=
    (SERVICE_NAME=service_name)
    (SERVER=server)
    (INSTANCE_NAME=instance_name)))
```

If you installed Oracle Database in **Typical** mode, then the default service name used by the `oracle` instance is `ORCL`, and the following Easy Connect syntax can be used to connect to that instance:

```
CONNECT username@"host/ORCL"
Enter password: password
```

Table 8–1 lists the Easy Connect syntax elements and descriptions for each.

Table 8–1 Connect Identifier for Easy Connection Naming Method

Syntax Element	Description

//	Specify // for a URL.
	For URL or JDBC connections, it is required that the connect identifier is preceded by a double-slash (//). For example:
	`scott@//sales-server`
	`Enter password: password`
	
	For SQL connections, it is optional that the connect identifier is preceded by a double-slash (//). For example, the following connect strings are semantically equivalent:
	`SQL> CONNECT scott@sales-server`
	`Enter password: password`
	
	`SQL> CONNECT scott@//sales-server`
	`Enter password: password`

host	Required. Specify the host name or IP address of the database host computer. The host name is domain-qualified if the local operating system configuration specifies a domain. You may use an IPv4 or IPv6 address as a value. IPv6 addresses or host names that resolve to IPv6 addresses must be enclosed in square brackets, as in [2001:0DB8:0:0::200C:417A] and [salesdb].
port	Optional. Specify the listening port. The default is 1521.
service_name	Optional. Specify the service name of the database. If a user specifies a service name, then the listener connects the user to that specific database. Otherwise, the listener connects to the database specified by the DEFAULT_SERVICE_*listener_name* parameter in the listener.ora file. If DEFAULT_SERVICE_*listener_name* is not configured for the listener and a service name is not explicitly specified by the user as part of the Easy Connect syntax, then the listener returns an error.
server	Optional. Specify the database server type to use. This parameter instructs the listener to connect the client to a specific type of service handler. The values for the server parameter are dedicated, shared, and pooled. If server is not specified in the Easy Connect syntax, then the type of server is chosen by the listener (shared server if configured, otherwise a dedicated server is used). **Note**: In Oracle Call Interface documentation, server is referred to as connect_type.
instance_name	Optional. Used to identify the database instance to access. The instance name can be obtained from the INSTANCE_NAME parameter in the initialization parameter file.

> **See Also:** *Oracle Database Net Services Reference* for additional information about configuring the `DEFAULT_SERVICE_`*`listener_name`* parameter, and the `INSTANCE_NAME` initialization parameter

The connect strings in Example 8–1 connect the client to database service `sales.us.example.com` with a listening endpoint of 1521 on database server `sales-server`.

Example 8–1 Easy Connect Strings

```
CONNECT scott@sales-server:1521/sales.us.example.com
CONNECT scott@//sales-server/sales.us.example.com
CONNECT scott@//sales-server.us.example.com/sales.us.example.com
```

After each of the connect strings in Example 8–1, you must enter a password to connect to the database service.

The connect strings in Example 8–1 convert into the following connect descriptor:

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

Configuring Easy Connect Naming on the Client

Clients can connect to Oracle Database using Easy Connect naming if the following conditions are met:

- Oracle Net Services software is installed on the client.
- Oracle TCP/IP protocol is supported on both the client and database server.
- No features require a more advanced connect descriptor are required.

Easy Connect naming is not suitable for large or complex environments with advanced features, such as **connection pooling**, **external procedure** calls, or **Heterogeneous Services**, that require additional connect information. In these cases, another naming method is recommended.

Easy Connect naming is automatically configured at installation. Before using it, you may want to ensure that `EZCONNECT` is specified by the `NAMES.DIRECTORY_PATH` parameter in the `sqlnet.ora` file. This parameter specifies the order of naming methods Oracle Net can use to resolve connect identifiers to connect descriptors.

To verify that the Easy Connect naming method is configured:

1. Start Oracle Net Manager.

   > **See Also:** "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, expand **Local**, and then select **Profile**.
3. From the list in the right pane, select **Naming**.
4. Click the **Methods** tab.

   Verify that `EZCONNECT` is listed in the Selected Methods list. If it is not, then proceed to Step 5. If it is listed, then proceed to Step 7.

5. From the Available Methods list, select **EZCONNECT**, and then click the right-arrow button.
6. In the Selected Methods list, select **EZCONNECT**, and then use the Promote button to move the selection to the top of the list.
7. Select **Save Network Configuration** from the File menu.

   The `sqlnet.ora` file updates the `NAMES.DIRECTORY_PATH` parameter, listing `hostname` first:

   ```
   NAMES.DIRECTORY_PATH=(ezconnect, tnsnames)
   ```

Configuring Easy Connect Naming to Use a DNS Alias

You can optionally configure a DNS alias for the host name, as provided with the host naming method in Oracle Database 11*g*. With host naming, clients use a connect string that uses the following pattern:

```
CONNECT username@DNSalias
Enter password: password
```

To configure a DNS alias:

1. Ensure the database service is registered with the listener.

   If the database can find the listener, then information about the database service is dynamically registered with the listener during **service registration**, including the service name. The listener is found if the following conditions are met:

   - The default listener named `LISTENER` on TCP/IP, port 1521 is running.
   - The `LOCAL_LISTENER` parameter is set in the initialization file.

   If the database cannot find the listener, then configure the `listener.ora` file with the `GLOBAL_DBNAME` parameter, as shown in the following example

   ```
   SID_LIST_listener=
   (SID_LIST=
    (SID_DESC=
     (GLOBAL_DBNAME=sales.example.com)
     (SID_NAME=sales)
     (ORACLE_HOME=/u01/app/oracle))
   ```

2. Establish a host name resolution environment.

   You can configure a mechanism such as DNS, NIS, or a centrally-maintained TCP/IP host file, `/etc/hosts`. For example, if a service name of `sales.us.example.com` for a database exists on a computer named `sales-server`, then the entry in the `/etc/hosts` file would look like the following:

   ```
   #IP address of server     host name       alias
   192.168.2.35              sales-server    sales.us.example.com
   ```

   The domain section of the service name must match the network domain.

3. Connect to the database using the DNS alias.

   Using the example in the previous step, the client can use `sales.example.com` in the connect string:

   ```
   CONNECT username@sales.us.example.com
   Enter password: password
   ```

If the client and server are in the same domain such as `us.example.com`, then the client must enter only `sales` in the connect string.

See Also: "Configuring Static Service Information" on page 9-5

Examples of Easy Connect Naming Method

Table 8–2 shows examples of Easy Connect naming syntax and how each string converts into a connect descriptor.

Table 8–2 Examples of Easy Connect Naming

Naming Option	Connect String	Connect Descriptor

Easy Connect string with host.
The host name is `sales-server`.
	`sales-server`	`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=))`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		`(HOST=sales-server)`
		`(PORT=1521)))`

Easy Connect string with host and port.
The host name is `sales-server`, and the port is `3456`.
	`sales-server:3456`	`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=))`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		`(HOST=sales-server)`
		`(PORT=3456)))`

Easy Connect string with host and service name.
The host name is `sales-server` and the service name is `sales`.
	`sales-server/sales`	`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=sales))`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		`(HOST=sales-server)`
		`(PORT=1521)))`

Easy Connect string with IPv6 address.
The IPv6 address of the host is `2001:0DB8:0:0::200C:417A`, the port is `80`, and the service name is `sales`.
	`[2001:0DB8:0:0::200C:417A]:80/sales`
	
	Square brackets are required around IPv6 host names.
		`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=sales)`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		
		`(HOST=2001:0DB8:0:0::200C:417A)`
		`(PORT=80)))`

Easy Connect string with IPv6 host address.
The IPv6 address of the host is `sales-server`, the port is `80`, and the service name is `sales`.
	`sales-server:80/sales`	`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=sales)`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		
		`(HOST=2001:0DB8:0:0::200C:417A)`
		`(PORT=80)))`

Table 8–2 (Cont.) Examples of Easy Connect Naming

Naming Option	Connect String	Connect Descriptor

Easy Connect string with host, service name, and server.
The host name is `sales-server`, the service name is `sales`, the server is `dedicated`, and the instance name is `inst1`
	`sales-server/sales:dedicated/inst1`	`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=sales)`
		`(INSTANCE_NAME=inst1)`
		`(SERVER=dedicated))`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		`(HOST=sales-server)`
		`(PORT=1521)))`

Easy Connect with host and instance name.
The host name is `sales-server` and the instance name is `inst1`.
	`sales-server//inst1`	`(DESCRIPTION=`
		`(CONNECT_DATA=`
		`(SERVICE_NAME=)`
		`(INSTANCE_NAME=inst1))`
		`(ADDRESS=`
		`(PROTOCOL=TCP)`
		`(HOST=sales-server)`
		`(PORT=1521)))`

See Also:

- "TCP/IP Protocol" on page 4-6
- "Configuring Listening Protocol Addresses" on page 9-4
- "Configuring Oracle Connection Manager as a Bridge for IPv4 and IPv6" on page 10-7

Configuring the Local Naming Method

The local naming method adds net service names to the `tnsnames.ora` file. Each net service name maps to a connect descriptor.

Example 8–2 shows the net service name `sales` mapped to the connect descriptor contained in `DESCRIPTION`. The `DESCRIPTION` section contains the protocol address and identifies the destination database service. In this example, the protocol is TCP/IP and the port is 1521.

Example 8–2 Connector Descriptor

```
sales=
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)))
```

Example 8–3 shows a valid `tnsnames.ora` entry to connect to a host identified with an IPv6 address and a port number of 1522.

Example 8–3 Connect Descriptor for IPv6 Connection

```
salesdb =
  ( DESCRIPTION =
    ( ADDRESS=(PROTOCOL=tcp)(HOST=2001:0DB8:1:1::200C:417A)(PORT=1522) )
    ( CONNECT_DATA =
        (SERVICES_NAME=sales.example.com) )
  )
```

You can configure local naming during or after installation, as described in the following sections:

- Configuring the tnsnames.ora File During Installation
- Configuring the tnsnames.ora File After Installation

See Also: "IPv6 Network Connectivity" on page 4-8

Configuring the tnsnames.ora File During Installation

Oracle Net Configuration Assistant enables you to configure net service names for clients. Oracle Universal Installer launches Oracle Net Configuration Assistant after software installation. The configuration varies depending on the installation mode.

- Administrator or run-time installation: Oracle Net Configuration Assistant prompts you to configure net service names in the `tnsnames.ora` file to connect to an Oracle Database service.
- Custom installation: Oracle Net Configuration Assistant prompts you to select naming methods to use. If local is selected, then Oracle Net Configuration Assistant prompts you to configure net service names in the `tnsnames.ora` file to connect to an Oracle Database service.

Configuring the tnsnames.ora File After Installation

You can add net service names to the `tnsnames.ora` file at any time after installation. To configure the local naming method, perform the following tasks:

- Task 1, "Configure Net Services Names"
- Task 2, "Configure Local Naming as the First Naming Method"
- Task 3, "Copy the Configuration to the Other Clients"
- Task 4, "Configure the Listener"
- Task 5, "Connect to the Database"

Note: The underlying network connection must be operational before attempting to configure connectivity with Oracle Net.

Task 1 Configure Net Services Names

To configure the net services names, use one of the following methods:

- Configuration using Oracle Enterprise Manager
- Configuration using Oracle Net Manager
- Configuration using Oracle Net Configuration Assistant

Each method provides similar functionality. However, Oracle Net Manager has more configuration options for the `sqlnet.ora` file.

- Configuration using Oracle Enterprise Manager

  To configure net service names in the `tnsnames.ora` file with Oracle Enterprise Manager, do the following

  1. Access the Net Services Administration page in Oracle Enterprise Manager.

See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Local Naming** from the Administer list, and then select the Oracle home that contains the location of the configuration files.
3. Click **Go**.

   The Local Naming page appears. You may be prompted to log in to the database server.
4. Click **Create Like**.

   The Create Net Service Name page appears.
5. Enter any name in the Net Service Name field.

   You can qualify the net service name with the client's domain. The net service name is automatically domain qualified if the `sqlnet.ora` file parameter `NAMES.DEFAULT_DOMAIN` is set.

   See Also: "Specifying a Default Domain for Clients" on page 12-2
6. In the Database Information section, configure service support:
   a. Enter a destination service name.

      See Also: "About Connect Descriptors" on page 2-5 for additional information about the service name string to use

   b. Select a database connection type.

      The default setting of Database Default is recommended for the connection type. If **shared server** is configured in the initialization parameter file, then you can select **Dedicated Server** to force the listener to spawn a dedicated server, bypassing shared server configuration. If shared server is configured in the initialization parameter file and you want to guarantee the connection always uses shared server, then select **Shared Server**.

      See Also: Chapter 11, "Configuring Dispatchers" for additional information about shared server configuration
7. In the Addresses section, configure protocol support, as follows:
   a. Click **Add**.

      The Add Address page appears.
   b. From the Protocol list, select the protocol on which the listener is configured to listen. This protocol must also be installed on the client.
   c. Enter the appropriate parameter information for the selected protocol in the fields provided.

      See Also: *Oracle Database Net Services Reference* for additional information about protocol parameter settings
   d. Optionally, in the Advanced Parameters section, specify the I/O buffer space limit for send and receive operations of sessions in the Total Send Buffer Size and Total Receive Buffer Size fields.

See Also: "Configuring I/O Buffer Space" on page 14-3 for additional information about buffer space

e. Click **OK**.

The protocol address is added to the Addresses section.

8. Click **OK** to add the net service name.

The net service name is added to the Local Naming page.

9. Select connect-time failover and client load balancing option for the addresses.

10. Click **OK**.

See Also:

- "Creating a List of Listener Protocol Addresses" on page 13-1 to configure multiple protocol addresses
- "Configuring Advanced Connect Data Parameters" on page 13-5 to configure additional `CONNECT_DATA` options

- Configuration using Oracle Net Manager

To configure net service names in the `tnsnames.ora` file with Oracle Net Manager, do the following:

1. Start Oracle Net Manager.

See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Service Naming** from Local.
3. Click the plus sign (+) from the toolbar, or select **Create** from the Edit menu.

The Welcome page of the Net Service Name wizard appears.

4. Enter a name in the Net Service Name field.

You can qualify the net service name with the client's domain. The net service name is automatically domain qualified if the `sqlnet.ora` file parameter `NAMES.DEFAULT_DOMAIN` is set.

See Also: "Specifying a Default Domain for Clients" on page 12-2

5. Click **Next**.

The Protocol page appears.

6. Select the protocol on which the listener is configured to listen. Note that this protocol must also be installed on the client.
7. Click **Next**.

The Protocol Settings page appears.

8. Enter the appropriate parameter information for the selected protocol in the fields provided.

See Also: *Oracle Database Net Services Reference* for additional information about protocol parameter settings

9. Click **Next**.

   The Service page appears.

10. Select a release, enter a destination service name, and optionally, select a database connection type.

    Oracle recommends that you use the default setting of Database Default for the connection type. If **shared server** is configured in the initialization parameter file, then you can select Dedicated Server to force the listener to spawn a dedicated server, bypassing shared server configuration. If shared server is configured in the initialization parameter file and you want to guarantee the connection always uses shared server, then select Shared Server.

    See Also:

    - Chapter 11, "Configuring Dispatchers" for additional information about shared server configuration
    - "About Connect Descriptors" on page 2-5 for additional information about the service name string to use

11. Click **Next**.

    The Test page appears.

12. Click **Test** to verify that the net service name works, or click **Finish** to dismiss the Net Service Name wizard.

    If you click Test, then Oracle Net connects to the database server by using the connect descriptor information you configured. Therefore, the listener and database must be running for a successful test. If they are not, then see "Starting Oracle Net Listener and the Oracle Database Server" on page 6-2 to start components before testing. During testing, a Connection Test dialog box appears, providing status and test results. A successful test results in the following message:

    ```
    The connection test was successful.
    ```

    If the test was successful, then click **Close** to close the Connect Test dialog box, and proceed to Step 13.

    If the test was not successful, then do the following:

    a. Ensure that the database and listener are running, and then click **Test**.

    b. Click **Change Login** to change the user name and password for the connection, and then click **Test**.

13. Click **Finish** to close the Net Service Name wizard.

14. Select **Save Network Configuration** from the File menu.

    See Also:

    - "Creating a List of Listener Protocol Addresses" on page 13-1 to configure multiple protocol addresses
    - "Configuring Advanced Connect Data Parameters" on page 13-5 to configure additional `CONNECT_DATA` options

- Configuration using Oracle Net Configuration Assistant

To configure net service names in the `tnsnames.ora` file with Oracle Net Configuration Assistant, do the following:

1. Start Oracle Net Configuration Assistant.

   See Also: "Using Oracle Net Configuration Assistant to Configure Network Components" on page 7-4

   The Welcome page appears.

2. Select **Local Net Service Name Configuration**, and then click **Next**.

   The Net Service Name Configuration page appears.

3. Click **Add**, and then click **Next**.

   The Service Name Configuration page appears.

4. Enter a service name in the Service Name field.

5. Click **Next**.

6. Follow the prompts in the wizard and online help to complete net service name creation.

Task 2 Configure Local Naming as the First Naming Method

Configure local naming as the first method specified in the `NAMES.DIRECTORY_PATH` parameter in the `sqlnet.ora` file. This parameter specifies the order of naming methods Oracle Net uses to resolve connect identifiers to connect descriptors.

To configure the local naming method as the first naming method, use one of the following methods:

- Configuration using Oracle Enterprise Manager
- Configuration using Oracle Net Manager

Each method provides the same functionality.

Configuration using Oracle Enterprise Manager

To specify local naming as the first naming method using Oracle Enterprise Manager, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Network Profile** from the Administer list.

3. Select **Naming Methods**.

4. Select **TNSNAMES** from the Available Methods list.

5. Click **Move** to move the selection to the Selected Methods list.

6. Use the promote button (^) to move TNSNAMES to the top of the list.

7. Click **OK**.

Configuration using Oracle Net Manager

To specify local naming as the first naming method using Oracle Net Manager, do the following:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select Profile from the Local menu.
3. From the list in the right pane, select **Naming**.
4. Click the **Methods** tab.
5. From the Available Methods list, select **TNSNAMES**, and then click the right-arrow button.
6. From the Selected Methods list, select **TNSNAMES**, and then use the **Promote** button to move the selection to the top of the list.
7. Choose **Save Network Configuration** from the File menu.

   The `sqlnet.ora` file updates with the `NAMES.DIRECTORY_PATH` parameter, listing `tnsnames` first:

   ```
   NAMES.DIRECTORY_PATH=(tnsnames, EZCONNECT)
   ```

Task 3 Copy the Configuration to the Other Clients

After one client is configured, it is best to simply copy the `tnsnames.ora` and `sqlnet.ora` configuration files to the same location on the other clients. This ensures that the files are consistent. Alternatively, you can use Oracle Net Assistant on every client.

Task 4 Configure the Listener

Ensure that the listener located on the server is configured to listen on the same protocol address configured for the net service name. By default, the listener is configured for the TCP/IP protocol on port 1521.

See Also: Chapter 9, "Configuring and Administering Oracle Net Listener" for listener configuration details

Task 5 Connect to the Database

Clients can connect to the database using the following syntax:

```
CONNECT username@net_service_name
Enter password: password
```

Configuring the Directory Naming Method

With the directory naming method, connect identifiers are mapped to connect descriptors contained in an LDAP-compliant directory server, such as Oracle Internet Directory and Microsoft Active Directory. A directory provides central administration of database services and net service names, making it easier to add or relocate services.

A database service entry is created with **Database Configuration Assistant** during installation. Oracle Enterprise Manager and Oracle Net Manager are used to create and modify net service names and **net service alias** entries, and to modify the database service entry. Clients can use these entries to connect to the database.

To configure the directory naming method, perform the following tasks:

- Task 1, "Verify Directory Compatibility"

- Task 2, "Create Net Service Names in the Directory"
- Task 3, "Modify Connectivity Information for Database Service Entries"
- Task 4, "Create Net Services Aliases"
- Task 5, "Configure LDAP as the First Naming Method for Client Lookups"
- Task 6, "Configure the Listener"
- Task 7, "Connect to the Database"

Task 1 Verify Directory Compatibility

On the computer from which you plan to create net service names, do the following verification steps:

1. Ensure that computer has the latest release of Oracle Net Services software. The release information is located in the About Net Manager option on the Help menu.
2. Run Oracle Internet Directory Configuration Assistant to verify directory server, Oracle Context, and Oracle schema releases.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for additional information about configuring directory server usage

Task 2 Create Net Service Names in the Directory

You can configure clients to use a net service name rather than the database service entry. To create net service names, do the following:

Notes:

- Only users that are members of either the `OracleNetAdmins` or `OracleContextAdmins` group can create net service name entries in a directory. To add or remove users from the `OracleNetAdmins` group, see "Adding Users To the OracleNetAdmins Group" on page 7-6.
- You can export existing net service names from a `tnsnames.ora` file. See "Exporting Local Naming Entries to a Directory Naming Server" on page 8-19.

---

1. Access the Net Services Administration page in Oracle Enterprise Manager.

> **See Also:** "Accessing the Net Services Administration Page" on page 7-2

2. Select **Directory Naming** from the Administer list, and then select the Oracle home that contains the location of the directory server.
3. Click **Go**.

   The Directory Naming page appears.
4. Click the **Net Service Names** tab.
5. In the Results section, click **Create**.

   The Create Net Service Name page with the General tab appears.
6. Enter any name in the Net Service Name field.

7. In the Database Information section, configure service support, as follows:

   a. Enter a destination service name.

      If the destination service name is for an Oracle9*i* database or later, then select **Use Service Name**, and enter a service name in the Service Name field.

      See Also: "About Connect Descriptors" on page 2-5 for additional information about the service name string to use

   b. Select a database connection type. Oracle recommends that you use the Database Default for the connection type. If **shared server** is configured in the initialization parameter file, then the following options are available:

      – Select Dedicated Server to force the listener to spawn a dedicated server, and bypass shared server configuration.

      – Select Shared Server to guarantee the connection always uses shared server.

      See Also: Chapter 11, "Configuring Dispatchers" for additional information about shared server configuration

8. In the Addresses section, configure protocol support, as follows:

   a. Click **Add**.

      The Add Address page appears.

   b. From the Protocol list, select the protocol on which the listener is configured to listen. This protocol must also be installed on the client.

   c. Enter the appropriate parameter information for the selected protocol in the fields provided.

      See Also: *Oracle Database Net Services Reference* for additional information about protocol parameter settings

   d. (Optional) In the Advanced Parameters section, specify the I/O buffer space limit for send and receive operations of sessions in the Total Send Buffer Size and Total Receive Buffer Size fields.

      See Also: "Configuring I/O Buffer Space" on page 14-3 for additional information

   e. Click **OK**.

      The protocol address is added to the Addresses section.

9. Click **OK** to add the net service name.

   The net service name is added to the Results section of the Net Service Names tab.

   See Also:

   - "Creating a List of Listener Protocol Addresses" on page 13-1 to configure multiple protocol addresses
   - "Configuring Advanced Connect Data Parameters" on page 13-5 to configure additional `CONNECT_DATA` options

Task 3 Modify Connectivity Information for Database Service Entries

When database registration with the directory naming completes, Database Configuration Assistant creates a database service entry in the directory. By default, this entry contains network route information with the location of the listener through a protocol address. You can re-create this information or modify the existing network route information.

Note: Only users that are members of the `OracleNetAdmins` or `OracleContextAdmins` group can modify network information for a database service in a directory. To add or remove users from these groups, see "Adding Users To the OracleNetAdmins Group" on page 7-6.

To create or modify network route information for a database service, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Directory Naming** from the Administer list, and then select the Oracle home that contains the location of the directory server.
3. Click **Go**. You may be prompted to log in to the database server and the directory server.

   The Directory Naming page appears.
4. Click the **Database Services** tab.
5. In the Simple Search section, select Oracle Context and search criteria to see the net service names for Oracle Context.

   The database service names display in the Results section.
6. In the Results section, select a database service, and then click **Edit**.

Task 4 Create Net Services Aliases

Net service aliases in a directory server enable clients to refer to a database service or a net service name by an alternative name. For example, a net service alias of `salesalias` can be created for a net service name of `sales`. When `salesalias` is used to connect to a database, as in `CONNECT scott@salesalias`, it resolves to and use the connect descriptor information for `sales`.

There are two main uses of net service aliases:

- Use a net service alias as a way for clients to refer to a database service or net service name by another name.
- Use a net service alias in one Oracle Context for a database service or net service name in a different Oracle Context. This enables a database service or net service name to be defined once in the directory server, and referred to by clients that use other Oracle Contexts.

  See Also: "Understanding Net Service Alias Entries" on page 3-9 for an overview of net service aliases

Notes:

- Only users that are members of either the `OracleNetAdmins` or `OracleContextAdmins` group can create or modify net service alias entries in a directory. To add or remove users from the `OracleNetAdmins` group, see "Adding Users To the OracleNetAdmins Group" on page 7-6.
- To create or access net service aliases, ensure that the Oracle home is at least release 9.2.0.4.
- Net service aliases are not supported by Microsoft Active Directory.
- Ensure the NLS_LANG environment variable is set for the clients when using net service aliases.

To create a net service alias, use one of the following methods:

- Configuration using Oracle Enterprise Manager
- Configuration using Oracle Net Manager

Each method provides similar functionality.

Configuration using Oracle Enterprise Manager

To configure a net service alias using Oracle Enterprise Manager, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Directory Naming** from the Administer list, and then select the Oracle home that contains the location of the directory server.
3. Click **Go**.

   The Directory Naming page appears.
4. Click the **Net Service Aliases** tab.
5. In the Results section, click **Create**.

   The Create Net Service Alias page appears.
6. Enter a name for the alias in the Net Service Alias Name field.
7. In the Referenced Service Detail section, enter the following information in the fields:
   - Oracle Context: Select the Oracle Context of the database service or net service name from the list or enter one in the field.
   - Referenced Service Name: Select the DN of the database service or net service name.
8. Click **OK** to add the net service alias.

   The net service alias is added to the Directory Naming page.

Configuration using Oracle Net Manager

To configure a net service alias using Oracle Net Manager, do the following:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Service Naming** from Directory.
3. Select **Aliases**.
4. Select **Create** from the Edit menu.
5. Enter the net service alias in the Net Service Alias field.
6. Select Oracle Context and name.
7. Click **Create**.
8. Select **Save Network Configuration** from the File menu.

Task 5 Configure LDAP as the First Naming Method for Client Lookups

Configure directory naming as the first method to be used in the `NAMES.DIRECTORY_PATH` parameter in the `sqlnet.ora` file. This parameter specifies the order of naming methods Oracle Net uses to resolve connect identifiers to connect descriptors. To configure LDAP as the first naming method you can use one of the following methods:

- Configuration using Oracle Enterprise Manager
- Configuration using Oracle Net Manager

Configuration using Oracle Enterprise Manager

To specify directory naming as the first naming method using Oracle Enterprise Manager, do the following

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Network Profile** from the Administer list.
3. Select **Naming Methods**.
4. Select **LDAP** from the Available Methods list.
5. Click **Move** to move the selection to the Selected Methods list.
6. Use the promote button (^) to move LDAP to the top of the list.
7. Click **OK**.

Configuration using Oracle Net Manager

To specify directory naming as the first naming method using Oracle Net Manager, do the following:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **Naming**.

4. Click the **Methods** tab.
5. From the Available Methods list, select **LDAP**, and then click the right-arrow button.
6. From the Selected Methods list, select **LDAP**, and then use the **Promote** button to move the selection to the top of the list.
7. Select **Save Network Configuration** from the File menu.

   The `sqlnet.ora` file updates with the `NAMES.DIRECTORY_PATH` parameter, listing `ldap` first, such as the following:

   ```
   NAMES.DIRECTORY_PATH=(ldap, tnsnames, hostname)
   ```

Task 6 Configure the Listener

Ensure that the listener located on the server is configured to listen on the same protocol address configured for the net service name. By default, the listener is configured to listen on the TCP/IP protocol, port 1521.

> **See Also:** Chapter 9, "Configuring and Administering Oracle Net Listener" for listener configuration details

Task 7 Connect to the Database

Clients that are configured with a default directory entry that matches the directory location of the database service or net service name can connect to the database using the following syntax:

```
CONNECT username@connect_identifier
Enter password: password
```

Clients that are configured with a default directory entry that does not match the entry's directory location must use the entry's distinguished name or its fully-qualified name.

> **See Also:**
>
> - "Connect Identifier and Connect Descriptor Syntax Characteristics" on page 2-15 for connect identifier syntax rules
> - "Understanding the Directory Information Tree" on page 3-3 for fully-qualified name usage

Creating Multiple Default Contexts in a Directory Naming Server

If you want clients to use discovery in directories which have more than one Oracle Context, then you can define the `orclCommonContextMap` attribute in the base `admin` context; this overrides the `orclDefaultSubscriber` attribute. During name lookup the discovery operation returns both values, and the client decides based on these which Oracle Context to use.

If the `orclCommonContextMap` attribute is not defined, then the `orclDefaultSubscriber` will be used as the default. If `orclCommonContextMap` is defined, then the client finds the default Oracle Context which is associated with its DNS domain in the `orclCommonContextMap`.

To enable multiple default contexts, define the `orclCommonContextMap` with a list of associations between a domain and a DN to be used as the default `oracleContext`. A sample LDIF file entry is shown here:

```
$ ldapmodify -v -h sales-server -p 1389 -D cn=orcladmin -q
```

```
dn: cn=Common,cn=Products,cn=OracleContext
replace: orclCommonContextMap
orclCommonContextMap:
(contextMap=
  (domain_map=(domain=us.example.com)(DN="dc=example,dc=com"))
  (domain_map=(domain=uk.example.com)(DN="dc=sales,dc=com"))
 )
```

The `contextMap` entry must be entered without line breaks.

> **See Also:** *Oracle Internet Directory Administrator's Guide* for additional information about how to configure the directory for context mapping

Exporting Local Naming Entries to a Directory Naming Server

This section explains how to export data stored in a `tnsnames.ora` file to a directory server. It includes the following tasks:

If a `tnsnames.ora` file already exists, then its net service names can be exported to a directory server. The export procedure is performed for one domain at a time.

The tasks to export data from a `tnsnames.ora` file are as follows:

- Task 1, "Create Structure in the Directory Server"
- Task 2, "Create Oracle Contexts"
- Task 3, "Configure Directory Server Usage"
- Task 4, "Export Objects to a Directory Server"

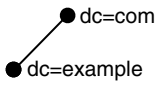
> **Note:** These tasks assume the directory server has been installed and is running.

Task 1 Create Structure in the Directory Server

In the directory server, create the **directory information tree (DIT)** with the structure in which you want to import net service names. Create the structure leading to the top of the **Oracle Context**.

For example, if the `tnsnames.ora` file supports a domain structure `example.com` and you want to replicate this domain in the directory, then create domain component entries of `dc=com` and `dc=example` in the directory, as shown in Figure 8–1.

Figure 8–1 example.com in Directory Server



You can replicate the domain structure you currently use with `tnsnames.ora`, or you can develop an entirely different structure. Introducing an entirely different structure can change the way clients enter the net service name in the connect string. Oracle recommends considering relative and fully-qualified naming issues before changing the structure.

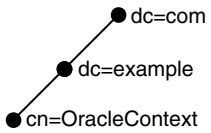
See Also:

- Directory server vendor documentation for directory entry configuration instructions
- "Client Connections Using Directory Naming" on page 3-11

Task 2 Create Oracle Contexts

Create an Oracle Context under each DIT location that you created in Task 1 using Oracle Internet Directory Configuration Assistant. Oracle Context has a **relative distinguished name (RDN)** of `cn=OracleContext`. Oracle Context stores network object entries, as well as other entries for other Oracle components. In Figure 8–2, `cn=OracleContext` is created under `dc=example,dc=com`.

Figure 8–2 Oracle Context



See Also:

- Chapter 3, "Managing Network Address Information" for additional information about Oracle Context
- *Oracle Internet Directory Administrator's Guide* for instructions on creating an Oracle Context

Task 3 Configure Directory Server Usage

If not done as a part of creating Oracle Contexts, then configure the Oracle home for directory server use. The Oracle home you configure should be the one that performs the export.

See Also: *Oracle Internet Directory Administrator's Guide* for additional information about configuring directory server usage

Task 4 Export Objects to a Directory Server

To export net service names contained in a `tnsnames.ora` file to a directory, use either Oracle Enterprise Manager or Oracle Net Manager.

- To export objects using Oracle Enterprise Manager, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Directory Naming** from the Administer list, and then select the Oracle home that contains the location of the directory server.
3. Click **Go**.

   The Directory Naming page appears.
4. Click the **Net Service Names** tab.
5. In the Related Links section, click **Import Net Service Names To Directory Server**.

The Import Net Service Names To Directory Server page appears.

6. From the Oracle Context list in the Oracle Internet Directory Server Destination section, select Oracle Context to which you want to export the selected net service names.
7. In the Net Service Names to Import section, select the net service names.
8. Click **Add** to add the net service names to the directory.

   The net service name is added to the Directory Naming page.

- To export objects using Oracle Net Manager, do the following:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. If the `tnsnames.ora` file you want to export is not loaded in Oracle Net Manager, then select **Open Network Configuration** from the File menu to select the `tnsnames.ora` file to export to the directory.
3. Select **Directory** from the Command menu, and then select **Export Net Service Names**.

   The Directory Server Migration wizard starts.
4. Click **Next**.

   If net service names with multiple domain were detected in the `tnsnames.ora` file, then the Select Domain page appears. Continue to Step 5.

   If the net service names are not domain qualified, then the Select Net Service Names page appears. Skip to Step 6.
5. Select the network domain whose net service names you want to export, and then click **Next**.

   The Select Net Service Names page appears.
6. Select the net service names from the list to export, and then click **Next**.

   The Select Destination Context page appears.
7. In the Select Destination Context page, perform the following:
   a. From the Directory Naming Context list, select the directory entry that contains the Oracle Context. The directory naming context is part of a directory subtree that contains one or more Oracle Contexts.
   b. From the Oracle Context list, select the Oracle Context to which you want to export the selected net service names.
   c. Click **Next**.

   The Directory Server Update page appears with the status of the export operation.
8. Click **Finish** to dismiss the Directory Server Migration wizard.

Exporting Directory Naming Entries to a tnsnames.ora File

After you create the directory naming entries, consider exporting the entries to a local `tnsnames.ora` file, and distributing that file to clients. Clients can use the locally saved file when a directory server is temporarily unavailable.

To export directory naming entries to a local `tnsnames.ora` file, do the following:

1. Access the Oracle Net Administration page in Oracle Enterprise Manager.

   See Also: "Accessing the Net Services Administration Page" on page 7-2

2. Select **Directory Naming** from the Administer list, and then select the Oracle home that contains the location of the directory server.
3. Click **Go**.

   The Directory Naming page appears.
4. Click the **Net Service Names** tab.
5. In the Simple Search section, select Oracle Context and search criteria to see the net service names for a particular Oracle Context.

   The net service names display in the Results section.
6. In the Results section, click **Save to tnsnames.ora**.

   The Processing: Create tnsnames.ora File page appears, informing you of the creation process.

Configuring External Naming Methods

External naming refers to the method of resolving a net service name, stored in a third-party naming service, to a network address, such as network information services (NIS). Organizations and corporations using NIS as part of their systems infrastructure have the option to store net service names and addresses in NIS, using NIS external naming.

For example, when a user gives a command such as the following and `payroll` is a net service name:

```
sqlplus scott@payroll
Enter password: password
```

NIS external naming on the node running the client program or database server acting as a client program contacts an NIS server located on the network, and passes the net service name to the NIS server. The NIS server resolves the net service name into an Oracle Net address and returns this address to the client program or server. The client program then uses this address to connect to Oracle Database.

An NIS server runs a program called `ypserv`, which handles name requests. The `ypserv` program stores different types of data in special files called **map**s. For example, passwords are stored in a map called `passwd.byname`. Oracle Database service names are stored in a map called `tnsnames`.

When a user uses a connect string, NIS external naming uses an RPC call to contact the `ypserv` program, and passes the Oracle net service name and the name of the map. The `ypserv` program looks in the `tnsnames` map for the name, such as `payroll`, and its address for the net service name. The address is returned to the client, and the client program uses the address to contact the database server.

Note: The NIS external naming method is not available on all platforms. Use the `adapters` command to check availability of NIS external naming on your system. If available, then it is listed under Oracle Net naming methods, as follows:

```
$ adapters

Installed Oracle Net naming methods are:

Local Naming (tnsnames.ora)
Oracle Directory Naming
Oracle Host Naming
NIS Naming
```

See Oracle platform-specific documentation for additional information.

This section contains the following tasks:

- Task 1, "Configure NIS Servers to Support NIS External Naming"
- Task 2, "Configure the Clients"

Task 1 Configure NIS Servers to Support NIS External Naming

Before configuring servers to support NIS external naming, ensure that NIS is configured and running on the NIS servers that need to resolve Oracle Database net service names. Consult your NIS documentation for specifics. To complete this task, add the tnsnames map to the existing NIS maps, and then verify that the tnsnames map has been installed properly.

1. Create a `tnsnames.ora` file, as specified in "Configuring the Local Naming Method" on page 8-6.

   Note: Keep a copy of the `tnsnames.ora` file, preferably in `ORACLE_HOME/network/admin` directory. You may need to use this file again later to load net service names into the NIS map.

   ---

2. Convert the contents of the `tnsnames.ora` file to a `tnsnames` map using the `tns2nis` program similar to the following:

   ```
   tns2nis tnsnames.ora
   ```

   The `tns2nis` program reads the `tnsnames.ora` file from the current directory. If `tnsnames.ora` file is not located in the current directory, then you can use a full path name to specify its location—for example, `/etc/tnsnames.ora` or `ORACLE_HOME/network/admin/tnsnames.ora`.

   The `tnsnames` map is then written into the current working directory.

   Note: The `tns2nis` program is supplied with NIS External Naming.

   ---

3. Copy `tnsnames` to the NIS server.
4. Install the `tnsnames` map using `makedbm`, which is an NIS program.

Note: This step should be performed by the person in charge of NIS administration.

The `makedbm` program converts the `tnsnames` map into two files that the NIS server can read. The location of these files is operating system specific.

See Also: Oracle operating system-specific documentation for details

For example, to generate and install a `tnsnames` map on Linux, as the `root` user, enter the following at the command line:

```
# makedbm tnsnames /var/yp/'domainname'/tnsnames
```

5. Verify `tnsnames` has been installed properly using the following command:

   ```
   ypmatch net_service_name tnsnames
   ```

   For example, you might enter the following command:

   ```
   ypmatch example.com tnsnames
   ```

   This returns the length of the address in characters, followed by the address such as the following:

   ```
   99 (description=(address=(protocol=tcp) (host=garlic)(port=1999)))
      (connect_data=(service_name=dirprod)))
   ```

Task 2 Configure the Clients

To configure clients, configure NIS as the first method specified in the `NAMES.DIRECTORY_PATH` parameter in the `sqlnet.ora` file. This parameter specifies the order of naming methods Oracle Net can use to resolve connect identifiers to connect descriptors.

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **Naming**.
4. Click the **Methods** tab.
5. From the Available Methods list, select **NIS**, and then click the right-arrow button.
6. In the Selected Methods list, select **NIS**, and then use the **Promote** button to move the selection to the top of the list.
7. Select **Save Network Configuration** from the File menu.

   The `sqlnet.ora` file updates with the `NAMES.DIRECTORY_PATH` parameter, listing `nis` first:

   ```
   NAMES.DIRECTORY_PATH=(nis, hostname, tnsnames)
   ```

9

Configuring and Administering Oracle Net Listener

Oracle Net Listener is a separate process that runs on the database server computer. It receives incoming client connection requests and manages the traffic of these requests to the database server. This chapter describes how to configure the listener to accept client connections.

This chapter contains the following topics:

Note: In Oracle Database 11*g* Release 2 (11.2), the password feature is being deprecated. This does not cause a loss of security because authentication is enforced through local operating system authentication. Refer to *Oracle Database Net Services Reference* for more information.

See Also:

- Chapter 2, "Identifying and Accessing the Database" for a description of how the listener is used during an initial connection request
- Chapter 4, "Understanding the Communication Layers" for an architectural overview of the listener

Overview of Oracle Net Listener

Note: The release of the listener must be the same as or later than the latest release of all Oracle databases being serviced through the listener.

A listener is configured with one or more listening protocol addresses, information about supported services, and parameters that control its run-time behavior. The listener configuration is stored in a configuration file named `listener.ora`.

Because all of the configuration parameters have default values, it is possible to start and use a listener with no configuration. This default listener has a name of `LISTENER`, supports no services on startup, and listens on the following TCP/IP protocol address:

```
(ADDRESS=(PROTOCOL=tcp)(HOST=host_name)(PORT=1521))
```

The listener forwards client requests to supported services. These services can be configured statically in the `listener.ora` file or they can be dynamically registered with the listener. This dynamic registration feature is called **service registration**. The registration is performed by the **PMON process**, an instance background process of each database instance that is configured in the database initialization parameter file. Dynamic service registration does not require any manual configuration in the `listener.ora` file.

Service registration offers the following benefits:

- Simplified configuration

  Service registration reduces the need for the `SID_LIST_`*`listener_name`* parameter setting, which specifies information about the databases served by the listener, in the `listener.ora` file.

  Note: The `SID_LIST_`*`listener_name`* parameter is required if you are using Oracle Enterprise Manager to manage the database.

- Connect-time failover

  Because the listener always monitors the state of the instances, service registration facilitates automatic failover of a client connect request to a different instance if one instance is down.

  Note: When services are configured statically, a listener starts a dedicated server when it receives a client request. If the instance is not up, then the server returns an `Oracle not available` error message.

- Connection load balancing

  Service registration enables the listener to forward client connect requests to the least-loaded instance and **dispatcher** or **dedicated server**. Service registration balances the load across the **service handler**s and nodes.

- High-availability for Oracle Real Application Clusters and Oracle Data Guard

See Also:

- "Understanding Oracle Net Architecture" on page 5-1
- "Configuring Service Registration" on page 9-9
- "Configuring Address List Parameters" on page 13-3
- "Configuring Connection Load Balancing" on page 13-7

Configuring Oracle Net Listener During Installation

Oracle Universal Installer launches **Oracle Net Configuration Assistant** during installation. Oracle Net Configuration Assistant configures the listening protocol address and service information for Oracle Database.

During an Enterprise Edition or Standard Edition installation on the database server, Oracle Net Configuration Assistant automatically configures a listener with a name of `LISTENER` that has a TCP/IP listening protocol address for Oracle Database. During a Custom installation, Oracle Net Configuration Assistant prompts for the listener name and protocol address.

A listening IPC protocol address for **external procedure** calls is automatically configured, regardless of the installation type. Oracle Net Configuration Assistant also automatically configures service information for the external procedures in the `listener.ora` file.

If you are using the IPC protocol, then you can improve performance by specifying the maximum number of concurrent IPC connection requests to match your expected connection requests.

Example 9–1 shows a sample `listener.ora` file. The `LISTENER` entry defines the listening protocol address for a listener named `LISTENER`, and the `SID_LIST_LISTENER` entry provides information about the external services statically supported by the listener `LISTENER`.

Example 9–1 Example listener.ora File

```
LISTENER=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc) (queuesize=50))))
SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (SID_NAME=plsextproc)
      (ORACLE_HOME=/oracle11g)
      (PROGRAM=extproc)))
```

> **See Also:** *Oracle Database Net Services Reference* for additional information about identifying listeners by unique names and creating multiple listener entries in the `listener.ora` file

Customizing Oracle Net Listener Configuration

If the default or installed configuration is not adequate for a particular environment, then you can use **Oracle Net Manager** to customize the `listener.ora` configuration.

This section contains the following configuration topics:

- Configuring Listening Protocol Addresses
- Handling Large Volumes of Concurrent Connection Requests
- Configuring Static Service Information
- Managing Oracle Net Listener Security

Configuring Listening Protocol Addresses

Oracle Enterprise Manager and Oracle Net Manager can be used to configure protocol support for the listener.

The Oracle Net Listener endpoint address configuration accepts both IPv6 addresses and host names that resolve to IPv6 addresses, as explained in "IPv6 Interface and Address Configurations" on page 4-7. This technique can create listening endpoints that service IPv6 clients.

Using Oracle Enterprise Manager to Configure Listening Protocol Addresses

To configure protocol addresses for the listener using Oracle Enterprise Manager, do the following:

1. In the General section of the Database Home page, click the link next to Listener.

   The Listener page appears.

2. Click **Edit**.

   The Edit Listener page appears. You may be prompted to log in to the database server.

3. In the Addresses section, configure protocol support:

   a. Click **Add**.

      The Add Address page appears.

   b. From the Protocol list, select the protocol on which the listener is configured to listen.

      For TCP/IP, if the computer has more than one IP address and you want the listener to listen on all available IP addresses, then select **TCP/IP** or **TCP/IP with SSL** and enter the host name of the computer in the Host field.

   c. In Port, enter the port number.

      When configuring the listener to listen on TCP/IP, enter the default port of 1521. Otherwise, you must configure the `LOCAL_LISTENER` parameter in the initialization parameter file and the non-default port number must be specified for use by any naming method.

   d. In Host, enter the host address.

   e. Optionally, in the Advanced Parameters section, specify the I/O buffer space limit for send and receive operations of sessions in the Total Send Buffer Size and Total Receive Buffer Size fields.

   f. Click **OK**.

      The protocol address is added to the Addresses section.

4. Repeat Step 3 for additional protocols.

See Also:

- *Oracle Database Net Services Reference* for additional information about protocol addresses and TCP/IP privileged ports
- "Configuring I/O Buffer Space" on page 14-3 for additional information

Using Oracle Net Manager to Configure Listening Protocol Addresses

To configure protocol addresses for the listener using Oracle Net Manager, do the following:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, expand **Local**, and then select **Listeners**.
3. Select the listener.
4. From the list in the right pane, select **Listener Locations**.
5. Select the protocol from the Protocol list.
6. Enter the host name for the listener in the Host field.
7. Enter the port number in the Port field.
8. If you want to set send and receive buffer sizes, then click **Show Advanced**, and then enter the sizes in the appropriate fields.
9. Select **Save Network Configuration** from the File menu to save the changes.

Handling Large Volumes of Concurrent Connection Requests

If you expect the listener to handle large volumes of concurrent connection requests, then you can specify a listener queue size for its TCP/IP or IPC listening endpoints.

To specify the listener queue size, do the following:

- Specify the `QUEUESIZE` parameter at the end of the protocol address with its value set to the expected number of concurrent requests.

  The following example sets the queue size to 20:

```
LISTENER=
 (DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)(QUEUESIZE=20)))
```

  Note: The default number of concurrent connection requests is operating system-specific. The defaults for TCP/IP on the Linux operating system and Microsoft Windows follow:

  - Linux operating system: 128
  - Microsoft Windows XP Professional SP2: 10
  - Microsoft Windows 2003 Server Enterprise Edition: 200

Configuring Static Service Information

The listener uses the dynamic service information about the database and instance before using statically configured information in the `listener.ora` file. Configuration of static service information is necessary if you require remote database startup from a tool other than Oracle Enterprise Manager, or you have Oracle Database releases earlier than Oracle8*i*.

Table 9–1 describes static service settings in the `listener.ora` file.

Table 9–1 Static Service Settings in listener.ora

Oracle Net Manager Field	listener.ora File Parameter	Description
SID	SID_NAME	The **Oracle system identifier (SID)** of the instance. You can obtain the SID value from the `INSTANCE_NAME` parameter in the initialization parameter file.
Service Name	GLOBAL_DBNAME	The database service. While processing a client connection request, the listener tries to match the value of this parameter with the value of the `SERVICE_NAME` parameter in the client connect descriptor. If the client connect descriptor uses the `SID` parameter, then the listener does not attempt to map the values. This parameter is primarily intended for configurations with Oracle8 databases (where dynamic service registration is not supported for dedicated servers). This parameter may also be required for use with Oracle8*i* and later database services by some configurations. The value for this parameter is typically obtained from the combination of the `DB_NAME` and `DB_DOMAIN` parameters (`DB_NAME.DB_DOMAIN`) in the initialization parameter file, but the value can also contain any valid name used by clients to identify the service.
Oracle Home Directory	ORACLE_HOME	The Oracle home location of the instance. Without this setting, the listener assumes its Oracle home for the instance. On Linux and UNIX, this setting is optional. On Microsoft Windows, this setting is ignored. The Oracle home specified by the `ORACLE_HOME` parameter in `HKEY_LOCAL_MACHINE\SOFTWARE\ORACLE\HOME`*`ID`* of the Microsoft Windows registry is used.

Important: If you are using **connect-time failover** or **Transparent Application Failover (TAF)**, such as in an Oracle Real Application Clusters environment, then do not set the `GLOBAL_DBNAME` parameter.

To statically configure the listener, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.
2. Select **Listeners** from the Administer list, and then select the Oracle home that contains the configuration files.
3. Click **Go**. You may be prompted to log in to the database server.

   The Listeners page appears.
4. Select a listener, and then click **Edit**.

   The Edit Listener page appears.
5. Click the **Static Database Registration** tab, and then click **Add**.

   The Add Database Service page appears. Enter the required information in the fields.
6. Click **OK**.

Note: You can also configure static service information with Oracle Net Manager. See **Statically Configure Database Service Information** in the online Help for additional information.

The following example shows an excerpt of a `listener.ora` file statically configured for a database service called `sales.us.example.com`:

```
SID_LIST_listener=
(SID_LIST=
 (SID_DESC=
  (GLOBAL_DBNAME=sales.us.example.com)
  (SID_NAME=sales)
  (ORACLE_HOME=/u01/app/oracle/11g)))
```

See Also:

- "Configuring Service Registration" on page 9-9 for additional information about configuring dynamic service registration Oracle Databases
- Chapter 13, "Enabling Advanced Features of Oracle Net Services" for additional information about statically configuring the listener for external procedures and Heterogeneous Services
- *Oracle Enterprise Manager Advanced Configuration* for additional information about Oracle Enterprise Manager

Managing Oracle Net Listener Security

By default, Oracle Net Listener permits only local administration for security reasons. As a policy, the listener can be administered only by the user who started it. This is enforced through local operating system authentication. For example, if `user1` starts the listener, then only `user1` can administer it. Any other user trying to administer the listener gets an error. The super user is the only exception.

Oracle recommends that you perform listener administration in the default mode (secure by means of local operating system authentication), and access the system remotely using a remote login. Oracle Enterprise Manager can also be used for remote administration.

Configuring and Changing the Oracle Net Listener Password

Local administration of the listener is secure by default through the local operating system. Therefore configuring a password is neither required nor recommended for secure local administration. However, a password can be configured for the listener to provide security for administrative operations, such as starting or stopping the listener, viewing a list of supported services, or saving changes to the Listener Control configuration.

Note: If the `PASSWORDS_`*`listener_name`* parameter is set to an unencrypted password, then you must manually remove it from the `listener.ora` file before changing it. If the unencrypted password is not removed, then you are unable to set an encrypted password.

You can use the Listener Control utility (`lsnrctl`) or Oracle Enterprise Manager to configure or change the Oracle Net Listener password.

- To set a new encrypted password using `lsnrctl`, do the following:

```
LSNRCTL> SET PASSWORD
Password: password
The command completed successfully
```

- To change an encrypted password using `lsnrctl`, do the following:

```
LSNRCTL> CHANGE_PASSWORD
Old password: old_password
New password: new_secure_password
Reenter new password: new_secure_password
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=tpc)(HOST=sales-server)(PORT=1521)))
Password changed for LISTENER
The command completed successfully

LSNRCTL> SAVE_CONFIG
Connecting to
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521)))
Saved LISTENER configuration parameters.
Listener Parameter File   /oracle/network/admin/listener.ora
Old Parameter File   /oracle/network/admin/listener.bak
The command completed successfully
```

- To set or change an encrypted password with Oracle Enterprise Manager, do the following:

  1. Access the Net Services Administration page in Oracle Enterprise Manager.

     See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

  2. Select **Listeners** from the Administer list, and then select the Oracle home that contains the location of the configuration files.
  3. Click **Go**. You may be prompted to log in to the database server.

     The Listeners page appears.
  4. Select a listener, and then click **Edit**.

     The Edit Listeners page appears.
  5. Click the **Authentication** tab.
  6. Click **Require a password for listener operations**.
  7. Click **OK**.
  8. Restart the listener.

See Also:

- *Oracle Database Security Guide* for additional information about minimum requirements for passwords
- *Oracle Database Net Services Reference* for additional information about the `CHANGE_PASSWORD` command
- "Configure Password Authentication for the Listener" in the online Help for additional information.

Configuring Service Registration

Service registration allows processes, such as an Oracle database, to identify their available services to the listener, which then acts as a port mapper for those services. The listener uses the dynamic service information about the database and instance received through service registration before using statically configured information in the `listener.ora` file.

Dynamic service registration is configured in the database initialization file. It does not require any configuration in the `listener.ora` file. However, listener configuration must be set to listen on the ports named in the database initialization file, and must not have parameters set that prevent automatic registration, such as COST parameters.

This section contains the following configuration topics related to service registration:

- Setting Initialization Parameters for Service Registration
- Registering Information with a Local Listener
- Registering Information with a Remote Listener
- Registering Information with All Listeners in a Network
- Configuring a Naming Method

Setting Initialization Parameters for Service Registration

To ensure service registration works properly, the initialization parameter file should contain the following parameters:

- `SERVICE_NAMES` for the database service name
- `INSTANCE_NAME` for the instance name
- `LOCAL_LISTENER` for the local listener
- `REMOTE_LISTENER` for the remote listener, if any

For example:

```
SERVICE_NAMES=sales.us.example.com
INSTANCE_NAME=sales
```

The value for the `SERVICE_NAMES` parameter defaults to the **global database name**, a name comprising the `DB_NAME` and `DB_DOMAIN` parameters in the initialization parameter file. The value for the `INSTANCE_NAME` parameter defaults to the SID.

See Also: *Oracle Database Reference* for additional information about the `SERVICE_NAMES` and `INSTANCE_NAME` parameters

Registering Information with a Local Listener

By default, the PMON process registers service information with its local listener on the default local address of TCP/IP, port 1521. If the listener configuration is synchronized with the database configuration, then PMON can register service information with a nondefault local listener or a remote listener on another node. Synchronization occurs when the protocol address of the listener is specified in the `listener.ora` file and the location of the listener is specified in the initialization parameter file.

To have PMON register with a local listener that does not use TCP/IP, port 1521, configure the `LOCAL_LISTENER` parameter in the initialization parameter file to locate the local listener.

For a shared server environment, you can use the `LISTENER` attribute of the `DISPATCHERS` parameter in the initialization parameter file to register the dispatchers with a nondefault local listener. Because the `LOCAL_LISTENER` parameter and the `LISTENER` attribute enable PMON to register dispatcher information with the listener, it is not necessary to specify both the parameter and the attribute if the listener values are the same.

`LOCAL_LISTENER` is a comma-delimited list parameter. If a comma appears in the string, then the entire string must be surrounded by double quotation marks. Set the `LOCAL_LISTENER` parameter as follows:

```
ALTER SYSTEM SET LOCAL LISTENER=["]listener_address["][,...];
```

For example, if the listener address `"ab,cd"` is entered, then it resolves to one listener address. If the address is entered as `ab,cd`, then it resolves to two listener addresses, `ab` and `cd`.

For shared server connections, set the `LISTENER` attribute as follows:

```
ALTER SYSTEM SET DISPATCHERS="(PROTOCOL=tcp)(LISTENER=listener_address)";
```

In the preceding command, *listener_address* is resolved to the listener protocol addresses through a naming method, such as a `tnsnames.ora` file on the database server.

Notes:

- To dynamically update the `LOCAL_LISTENER` parameter, use the SQL statement `ALTER SYSTEM`, as follows:

  ```
  ALTER SYSTEM SET LOCAL_LISTENER=["]listener_address["][,...]
  ```

  If you set the parameter to null using the following statement, then the default local address of TCP/IP, port 1521 is assumed:

  ```
  ALTER SYSTEM SET LOCAL_LISTENER=''
  ```

- The `LISTENER` attribute overrides the `LOCAL_LISTENER` parameter. As a result, the SQL statement `ALTER SYSTEM SET LOCAL_LISTENER` does not affect the setting of this attribute.

In Example 9–2, a database resides on host `sales1-server`. A listener on this host named `listener_sales1` is configured to listen on port 1421 instead of port 1521.

Example 9–2 Registering a Local Listener in a Dedicated Server Environment

1. On the host where the local listener resides, configure the `listener.ora` file with the protocol address of the listener using Oracle Net Manager.
2. On the database, set the `LOCAL_LISTENER` parameter in the database initialization parameter file to the alias of the local listener. For example:

   ```
   ALTER SYSTEM SET LOCAL_LISTENER=listener_sales1;
   ```

   If the database is configured for shared server connections, then you could set the `LISTENER` attribute as follows:

   ```
   ALTER SYSTEM SET DISPATCHERS="(PROTOCOL=tcp)(LISTENER=listener_sales1)";
   ```

3. Resolve the listener name alias for the `LOCAL_LISTENER` setting through a `tnsnames.ora` file on the database host using a text editor, as follows:

   ```
   listener_sales1=
    (DESCRIPTION =
     (ADDRESS = (PROTOCOL=tcp)(HOST=sales-server)(PORT=1421)))
   ```

   Notes:

   - If you are registering a local listener and use Oracle Connection Manager, then do not include `(DESCRIPTION =` or its closing parenthesis.
   - A net service name entry can be created for the protocol address without the `CONNECT_DATA` section of the connect descriptor.

   See Also:

   - See the *Oracle Database SQL Reference* for additional information about the `ALTER SYSTEM` statement.
   - "Configuring Listening Protocol Addresses" on page 9-4
   - "Configuring a Naming Method" on page 9-15

Registering Information with a Remote Listener

A **remote listener** is a listener residing on one computer that redirects connections to a database instance on another computer. Remote listeners are typically used in an Oracle Real Application Clusters (Oracle RAC) environment. You can configure registration to remote listeners, such as with Oracle RAC, for dedicated or shared server environments.

In a dedicated server environment, you must enable the PMON background process to register with a remote listener. You do this by configuring the `REMOTE_LISTENER` parameter, which is a comma-delimited list parameter, in the initialization parameter file. The syntax of `REMOTE_LISTENER` is as follows:

```
ALTER SYSTEM SET REMOTE_LISTENER=["]listener_address["][,...];
```

In the preceding command, *listener_address* is resolved to the listener protocol addresses through a naming method such as a `tnsnames.ora` file on the database host. If a comma appears in the listener address, then the entire string must be surrounded by quotation marks.

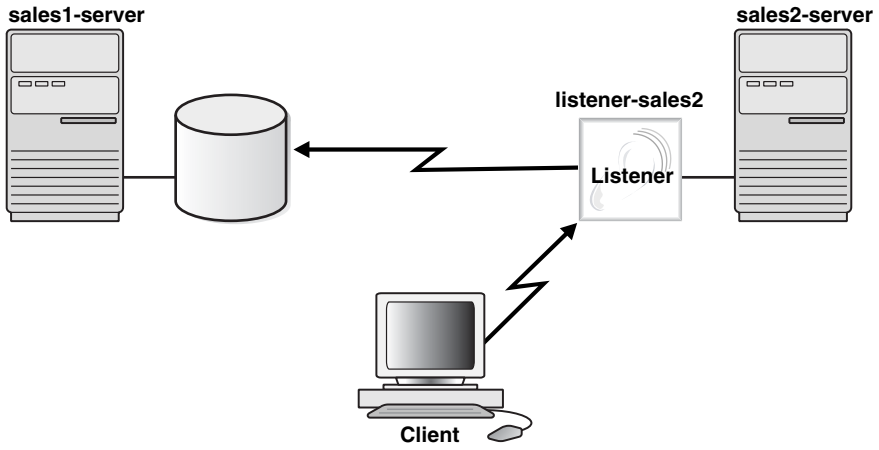
In a shared server environment, you can use the same registration technique as for a dedicated server environment. Alternatively, you can set the LISTENER attribute of the DISPATCHERS parameter in the initialization parameter file to register the dispatchers with any listener. The syntax of the LISTENER attribute is as follows:

```
ALTER SYSTEM SET DISPATCHERS="(PROTOCOL=tcp)(LISTENER=listener_address)";
```

> **Note:** The LISTENER attribute overrides the REMOTE_LISTENER initialization parameter. Because the REMOTE_LISTENER initialization parameter and the LISTENER attribute enable PMON to register dispatcher information with the listener, you do not need specify both the parameter and the attribute if the listener values are the same.

For example, assume that a remote listener named listener-sales2 listens on port 1521 on host sales2-server, and a database resides on host sales1-server. You want the listener on sales2-server to redirect connection requests to this database. Figure 9–1 illustrates this scenario.

Figure 9–1 Remote Listener



> **See Also:** *Oracle Database SQL Reference* for additional information about the ALTER SYSTEM SET statement

Example 9–3 shows how to register a remote listener in a dedicated server environment. In the example, the remote listener is sales2-server.

Example 9–3 Registering a Remote Listener in a Dedicated Server Environment

1. On the host where the remote listener resides, use Oracle Net Manager to configure the listener.ora file with the protocol addresses of the remote listener.
2. On the database to which you want requests to be redirected, set the REMOTE_LISTENER parameter in the database initialization parameter file to the alias of the remote listener, for example:

   ```
   ALTER SYSTEM SET REMOTE_LISTENER=listener_sales2;
   ```

   For shared server connections, set the DISPATCHER parameter in the initialization file for the database on host sales1-server as follows:

```
ALTER SYSTEM SET DISPATCHERS="(PROTOCOL=tcp)(LISTENER=listeners_sales2)";
```

Note: To statically update the REMOTE_LISTENER initialization parameter, use a text editor to de-register the information with the remote listener with which it had previously registered information.

3. Resolve the listener name alias for the REMOTE_LISTENER setting through a tnsnames.ora file on the database host. For example:

```
listener_sales2=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
```

See Also:

- "Configuring a Naming Method" on page 9-15
- "Configuring Listening Protocol Addresses" on page 9-4
- *Oracle Database Reference* to learn about the REMOTE_LISTENER initialization parameter
- *Oracle Real Application Clusters Administration and Deployment Guide* to learn how to configure remote listeners (also called SCAN listeners) in an Oracle RAC environment

Registering Information with All Listeners in a Network

A network may contain multiple local and remote listeners. By default, all listeners are cross-registered with each other. By specifying a set of listeners in the LISTENER_NETWORKS initialization parameter, you can designate a subset of local listeners with a subset of remote listeners. Listeners specified by the LISTENER_NETWORKS parameter should not be specified by the LOCAL_LISTENER and REMOTE_LISTENER parameters.

The syntax of LISTENER_NETWORKS is as follows:

```
LISTENER_NETWORKS = '((NAME=network_name)
                      (LOCAL_LISTENER=["]listener_address[,...]["])
                      [(REMOTE_LISTENER=["]listener_address[,...]["])])'
```

In the preceding syntax, *listener_address* is resolved according to the rules of LOCAL_LISTENER and REMOTE_LISTENER.

Example 9–4 Using Two Networks on a Subnet

Assume there are two distinct networks, network1 and network2. On network1, there is a local listener named local1, and a remote listener named remote1. On network2, there is a local listener named local2, and a remote listener named remote2. The following syntax sets up registration so that the listeners only redirect connections to listeners on the same network.

```
LISTENER_NETWORKS = '((NAME=network1) (LOCAL_LISTENER=local1) (REMOTE_
LISTENER=remote1))',
LISTENER_NETWORKS = '((NAME=network2) (LOCAL_LISTENER=local2) (REMOTE_
LISTENER=remote2))'
```

In the preceding example, local1 is registered only with remote1, and remote1 only redirect connections to local1. The listener local2 is registered only with remote2, and remote2 only redirect connections to local2.

Example 9–5 Configuring Multiple Listeners

Assume that multiple listeners are listening on a network named sales-network. The following conditions are true:

- A database configured for dedicated server connections resides on host sales1-server. It is the only database in the network.
- A local listener resides on sales1-server and listens on nondefault port 1421.
- A remote listener named resides on host sales2-server and listens on port 1521.
- Another remote listener resides on host sales3-server and listens on port 1521.

To register information with all listeners in a dedicated server environment, do the following:

1. On the hosts where the remote listeners reside (in this example, sales2-server and sales3-server), configure the listener.ora file with the protocol addresses of the remote listener.
2. On the database to which you want requests to be redirected, set the REMOTE_LISTENER parameter in the database initialization parameter file to the alias of the remote listeners, and the LOCAL_LISTENER parameter to the alias of the local listener.

   Set the parameters in the initialization file for the database on host sales1-server as follows:

   ```
   REMOTE_LISTENER="listener_sales2,listener_sales3"
   LOCAL_LISTENER=listener_sales1
   ```

3. Resolve the listener name alias for the LOCAL_LISTENER and REMOTE_LISTENER setting through a tnsnames.ora file on the database host.

   In the tnsnames.ora on sales1-server, resolve the local listener alias and remote listener aliases listener_sales1, listener_sales2, and listener_sales3 as follows:

   ```
   listener_sales1=
    (DESCRIPTION=
     (ADDRESS=(PROTOCOL=TCP)(HOST=sales1-server)(PORT=1421)))

   listener_sales2=
    (DESCRIPTION=
     (ADDRESS=(PROTOCOL=TCP)(HOST=sales2-server)(PORT=1521)))

   listener_sales3=
    (DESCRIPTION=
     (ADDRESS=(PROTOCOL=TCP)(HOST=sales3-server)(PORT=1521)))

   listener_sales_local=
    (DESCRIPTION=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1421)))

   listener_sales_remote=
   (DESCRIPTION_LIST=
     (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
   ```

```
(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales3-server)(PORT=1521))))
```

See Also:

- "Configuring a Naming Method" on page 9-15
- "Configuring Listening Protocol Addresses" on page 9-4
- *Oracle Database Reference* for additional information about the REMOTE_LISTENER initialization parameter

Configuring a Naming Method

The listener name alias specified for the LOCAL_LISTENER initialization parameter, REMOTE_LISTENER initialization parameter, or LISTENER attribute can be resolved through a tnsnames.ora file. A net service name entry can be created for the protocol address without the CONNECT_DATA section of the connect descriptor. Oracle Enterprise Manager and Oracle Net Manager cannot configure a tnsnames.ora file without the CONNECT_DATA information. To use listener name aliases, Oracle recommends you modify the tnsnames.ora file using a text editor.

For example, if LOCAL_LISTENER is set to listener_sales1 and listener_sales1 uses TCP/IP on port 1421, then the entry in the tnsnames.ora file would be:

```
listener_sales1=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1421))
```

Notes:

- Multiple addresses are supported, but connect-time failover and client load balancing features are not supported.
- If the listener alias specified in the LOCAL_LISTENER parameter is invalid or not resolved, then the PMON process does not allow the database to start. The following errors occur:

```
ORA-00119: invalid specification for system parameter
           LOCAL_LISTENER
ORA-00132: syntax error or unresolved network name '%s'
```

See Also: Chapter 13, "Enabling Advanced Features of Oracle Net Services" for additional information about multiple address configuration

Administering the Listener

After the listener is configured, you can administer it with the Listener Control utility, Oracle Enterprise Manager, and the Server Control utility (SVRCTL). This section describes some of the following administrative tasks for the listener. It contains the following topics:

- Starting and Stopping a Listener
- Managing a Listener in an Oracle Restart Configuration
- Determining the Current Status of a Listener
- Monitoring Services of a Listener

- Monitoring Listener Log Files

See Also:

- *Oracle Database Net Services Reference* for a complete list of the Listener Control utility commands
- Oracle Enterprise Manager online Help

Starting and Stopping a Listener

To stop or start a listener, use one of the following methods:

- Using the Listener Control Utility to Start or Stop a Listener
- Using Oracle Enterprise Manager to Start or Stop a Listener

Note: You can configure the listener to start automatically whenever the computer it is running on is restarted. See your operating system-specific documentation for details about establishing auto-restart.

Using the Listener Control Utility to Start or Stop a Listener

To start the listener from the command line, enter:

```
lsnrctl START [listener_name]
```

In the preceding command, *listener_name* is the name of the listener defined in the `listener.ora` file. It is not necessary to identify the listener if you are using the default listener, named `LISTENER`.

In addition to starting the listener, the Listener Control utility verifies connectivity to the listener.

To stop a listener from the command line, enter:

```
lsnrctl STOP [listener_name]
```

In the preceding command, *listener_name* is the name of the listener defined in the `listener.ora` file. It is not necessary to identify the listener if you are using the default listener, named `LISTENER`.

Using Oracle Enterprise Manager to Start or Stop a Listener

To start or stop a listener from Oracle Enterprise Manager, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

2. Select **Listeners** from the Administer list, and then select the Oracle home that contains the location of the configuration files.
3. Click **Go**.

   The Listeners page appears.
4. Select a listener.
5. From the Actions list, select **Start/Stop**.

The Start/Stop page appears.

6. Depending upon the current status of the selected listener, select either Stop or Start, and then click **OK**.

Managing a Listener in an Oracle Restart Configuration

The Oracle Restart feature enhances availability for the processes and applications in a single-instance database environment. You can add selected components to the Oracle Restart configuration. The Oracle Restart agents monitor the health of added components by periodically running check operations and restarting the components when necessary.

You can add the listener as a component to the Oracle Restart configuration. The listener is automatically started by Oracle Restart when it fails or is not running. For example, if you restart the database instance after a planned restart of the computer, then Oracle Restart restarts the listener. Server Control (SRVCTL) is a command-line interface that you can use to manage listeners in an Oracle Restart configuration.

To view all configured listeners, use the following command:

```
srvctl config listener
```

> **See Also:** *Oracle Database Administrator's Guide* to learn how to configure Oracle Restart and for SRVCTL syntax and semantics

Adding or Removing a Listener Using SRVCTL

Adding a listener means to add an entry for the listener to the grid infrastructure, enabling the agent to monitor this component. Similarly, removing a listener means removing an entry for a listener. In both cases you enter the command `srvctl` at the operating system command line.

- Enter `srvctl add listener` to add the listener.

  For example, the following command adds an entry for `listener_sales1` to the grid infrastructure:

  ```
  % srvctl add listener -l listener_sales1
  ```

- Enter `srvctl remove listener` to remove the listener.

  For example, the following command removes the entry for `listener_sales1` from the grid infrastructure:

  ```
  % srvctl remove listener -l listener_sales1
  ```

Starting or Stopping a Listener Using SRVCTL

The SRVCTL utility enables you to stop and start the listener. If you do not specify the `-l` parameter, then SRVCTL starts and stops the default listener.

- Enter `srvctl start listener` to start a listener.

  For example, the first command starts the default listener, and the second command starts `listener1` and `listener2`:

  ```
  % srvctl start listener
  % srvctl start listener -l listener1,listener2
  ```

- Enter `srvctl stop listener` to stop a listener.

For example, the first command stops the default listener, and the second command stops `listener1` and `listener2`:

```
% srvctl stop listener
% srvctl stop listener -l listener1,listener2
```

Determining the Current Status of a Listener

To show the current status of a listener, use either the `STATUS` command of the Listener Control utility or Oracle Enterprise Manager. The status output provides basic status information about a listener, a summary of listener configuration settings, the listening protocol addresses, and a summary of services registered with the listener.

Using Listener Control to Show Status

To show the status the listener from the command line, enter:

```
lsnrctl STATUS [listener_name]
```

In the preceding command, *listener_name* is the name of the listener defined in the `listener.ora` file. It is not necessary to identify the listener if you are using the default listener, named `LISTENER`.

Example 9–6 shows example output of the `STATUS` command.

Example 9–6 Listener Control Utility's STATUS Command Output

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)))
STATUS of the LISTENER
------------------------
Alias                     LISTENER
Version                   TNSLSNR for Linux: Version 11.2.0.0.2
Start Date                15-NOV-2009 20:22:00
Uptime                    0 days 0 hr. 5 min. 22 sec
Trace Level               support
Security                  OFF
SNMP                      OFF
Listener Parameter File   /oracle/admin/listener.ora
Listener Log File         /oracle/network/log/listener.log
Listener Trace File       /oracle/network/trace/listener.trc
Listening Endpoints Summary...
  (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)))
  (DESCRIPTION=(ADDRESS=(PROTOCOL=tcps)(HOST=sales-server)(PORT=2484)))

Services Summary...
Service "sales.us.example.com" has 1 instance(s).
  Instance "sales", status READY, has 3 handler(s) for this service...
Service "hr.us.example.com" has 1 instance(s).
  Instance "hr", status READY, has 2 handler(s) for this service...
The command completed successfully
```

The `STATUS` command generates output with the sections described in Table 9–2.

Table 9–2 Listener Control Utility STATUS Command

Output Section	Description
STATUS of the LISTENER	Status of the listener, which can be one of the following: ■ Alias of the listener ■ Version of listener ■ Start time and up time ■ Trace level ■ Whether the listener can respond to queries from an SNMP-based network management system ■ `listener.ora` file being used ■ Logging and tracing configuration settings ■ Whether a password is set in `listener.ora` file
Listening Endpoints Summary	The protocol addresses the listener is configured to listen on
Services Summary	A summary of the services registered with the listener and the service handlers allocated to each service
Service	The registered service
Instance	The name of the instance associated with the service along with its status and number of service handlers associated with the service Status can be one of the following: ■ A `READY` status means that the instance can accept connections. ■ A `BLOCKED` status means that the instance cannot accept connections. ■ A `READY/SECONDARY` status means that this is a secondary instance in an Oracle Real Application Clusters primary/secondary configuration and is ready to accept connections. ■ An `UNKNOWN` status means that the instance is registered statically in the `listener.ora` file rather than dynamically with service registration. Therefore, the status is not known.

Using Oracle Enterprise Manager to Show Status

To show the status of a listener from Oracle Enterprise Manager, do the following:

1. Access the Net Services Administration page in Oracle Enterprise Manager.

   See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

2. Select **Listeners** from the Administer list, and then select the Oracle home that contains the location of the configuration files.
3. Click **Go**. You may be prompted to log in to the database server.

   The Listeners page appears.
4. Select a listener.
5. From the Actions list, select **Show Listener Control Status**.
6. Click **Go**.

   The Listener Control Status page appears.

7. After viewing the content, click the listener link at the top of the page.

Monitoring Services of a Listener

The SERVICES command of the Listener Control utility provides detailed information about the services and instances registered with a listener and the service handlers allocated to each instance. To show information about the services and instances from the command line, enter:

```
lsnrctl SERVICES [listener_name]
```

Example 9–7 shows example output of the SERVICES command.

Example 9–7 Listener Control Utility's SERVICES Command Output

```
Connecting to (DESCRIPTION=(ADDRESS=(PROTOCOL=ipc)(KEY=net)))
Services Summary...
Service "sales.us.example.com" has 1 instance(s).
  Instance "sales", status READY, has 3 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
      "D000" established:0 refused:0 current:0 max:10000 state:ready
         DISPATCHER <machine: sales-server, pid: 1689>
         (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=52414))
      "D001" established:0 refused:0 current:0 max:10000 state:ready
         DISPATCHER <machine: sales-server, pid: 1691>
         (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=52415))
Service "hr.us.example.com" has 1 instance(s).
  Instance "hr", status READY, has 2 handler(s) for this service...
    Handler(s):
      "DEDICATED" established:0 refused:0 state:ready
         LOCAL SERVER
      "D000" established:0 refused:0 current:0 max:10000 state:ready
         DISPATCHER <machine: sales-server, pid: 11326>
         (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=58361))
The command completed successfully
```

This output shows that two database services, sales.us.example.com and hr.us.example.com, are registered with the listener.

Client connection requests to sales.us.example.com are handled by two dispatchers named D000 and D001 and one dedicated server. All handlers have a status of ready, indicating that they are ready to receive connections.

Client connection requests to hr.us.example.com are handled by one dispatcher named D001 and one dedicated server.

The SERVICES command generates output with the following information as described in Table 9–3.

Table 9–3 Listener Control Utility SERVICES Command

Output Section	Description
Services	The registered service.

Table 9–3 (Cont.) Listener Control Utility SERVICES Command

Output Section	Description
Instance	The name of the instance associated with the service The status field indicates if the instance can accept connections. ■ A `READY` status means that the instance can accept connections. ■ A `BLOCKED` status means that the instance cannot accept connections. ■ A `READY/SECONDARY` status means that the is a secondary instance in an Oracle Real Application Clusters primary/secondary configuration and is ready to accept connections. ■ A `RESTRICTED` status means that the instance is in restricted mode. The listener blocks all connections to this instance. ■ An `UNKNOWN` status means that the instance is registered statically in the `listener.ora` file rather than dynamically with service registration. Therefore, the status is non known.
Handlers	The name of the service handler. Dispatchers are named `D000` through `D999`. Dedicated servers have the name of `DEDICATED`. This section also identifies the following about the service handler: ■ `established`: The number of client connections this service handler has established. ■ `refused`: The number of client connections it has refused. ■ `current`: The number of client connections it is handling, that is, its current load. ■ `max`: The maximum number of connections for the service handler, that is, its maximum load. ■ `state`: The state of the handler: - A `READY` state means that the service handler can accept new connections. - A `BLOCKED` state means that the service handler cannot accept new connections.

Monitoring Listener Log Files

When you notice any of the following conditions, review the listener log file for error information:

- Long connection establishment times
- Connectivity problems and refusals
- Unexpected shutdown of the listener that could indicate a denial-of-service attack

See Also: "Analyzing Listener Log Files" on page 16-27

10

Configuring Oracle Connection Manager

Oracle Connection Manager is a **proxy server** that forwards connection requests to databases or other proxy servers. It operates on the session level. It usually resides on a computer separate from the database server and client computers. Oracle Connection Manager is available for installation with Oracle Database 11*g* Enterprise Edition. It is a custom installation option on the Client disk.

The primary functions of Oracle Connection Manager are:

- Access control: To use rule-based configuration to filter out user-specified client requests and accept others.
- Session multiplexing: To funnel multiple client sessions through a network connection to a **shared server** destination.

This chapter describes how to configure **Oracle Connection Manager** features. This chapter contains the following topics:

See Also:

- Chapter 1, "Introducing Oracle Net Services" for an introduction to Oracle Connection Manager concepts
- Chapter 4, "Understanding the Communication Layers" for an architectural overview of Oracle Connection Manager

Configuring Oracle Connection Manager

In order to configure Oracle Connection Manager you must configure the proxy server, database, and clients. The following tasks describe the general procedure:

Task 1 Configure the cman.ora file on the Oracle Connection Manager computer

The `cman.ora` file specifies the listening endpoint for the server, access control rules, and Oracle Connection Manager performance parameters.

"Configuring the cman.ora file for the Oracle Connection Manager Host" on page 10-2 explains how to perform this task.

Task 2 Configure the clients with the protocol address of the Oracle Connection Manager listener

"Configuring Clients for Oracle Connection Manager" on page 10-5 explains how to perform this task.

Task 3 Configure the database server for session multiplexing. This task is optional

"Configuring the Oracle Database Server for Oracle Connection Manager" on page 10-6 explains how to perform this task.

Configuring the cman.ora file for the Oracle Connection Manager Host

You configure the computer that hosts Oracle Connection Manager by setting parameters in the `cman.ora` file. This file resides on the computer that hosts Oracle Connection Manager, and is located in the `ORACLE_HOME/network/admin` directory. Oracle Connection Manager will not start if the `cman.ora` file does not exist. This file includes the following components:

- Listening endpoint
- Access control rule list
- Parameter list

Each Oracle Connection Manager configuration is encapsulated within a single name-value (NV) string, which consists of the preceding components.

One computer can host any number of Oracle Connection Managers, each with its own entry in the `cman.ora` file. When defining more than one Oracle Connection Manager in the file, you can assign a default by giving only one a fully qualified host name.

You make changes to the `cman.ora` file manually. You can specify multiple rules for both client and Oracle Connection Manager Control utility (CMCTL) connections. The following guidelines apply when making changes:

- You must enter at least one rule for client connections and one rule for CMCTL connections. Omitting a rule results in the rejection of all connections for the rule type omitted.
- Oracle Connection Manager does not support wildcards for partial IP addresses. If you use a wildcard, then use it in place of a full IP address. The IP address of the client may be, for example, (`SRC=*`).
- Oracle Connection Manager supports only the `/nn` notation for subnet addresses. In Example 10–1, in the first rule, `/27` represents a subnet mask that comprises 27 left-most bits. Only the first 27 bits in the client's IP address are compared with the IP address in the rule.

Note: Oracle Connection Manager supports IPv6 addressing. See "Configuring Oracle Connection Manager as a Bridge for IPv4 and IPv6" on page 10-7.

To set parameters in the `cman.ora` file, do the following:

1. Navigate to the `cman.ora` file in the `ORACLE_HOME/network/admin` directory.
2. Open the `cman.ora` file with a text editor.

Example 10–1 shows a `cman.ora` file that contains a configuration entry for an Oracle Connection Manager called `CMAN1`.

Example 10–1 Sample cman.ora File

```
CMAN1=
  (CONFIGURATION=
    (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1521))
    (RULE_LIST=
      (RULE=(SRC=192.168.2.32/27)(DST=sales-server)(SRV=*)(ACT=accept)
        (ACTION_LIST=(AUT=on)(MCT=120)(MIT=30)))
      (RULE=(SRC=192.168.2.32)(DST=proxysvr)(SRV=cmon)(ACT=accept)))
    (PARAMETER_LIST=
      (MAX_GATEWAY_PROCESSES=8)
      (MIN_GATEWAY_PROCESSSES=3)))
```

3. Configure the listening endpoint (`ADDRESS`).

   The listening endpoint specifies the protocol address for the Oracle Connection Manager listener. CMON, the Oracle Connection Manager monitoring process, uses this address to register information about gateway processes with the listener. The database uses the address to register service information at the Oracle Connection Manager node.

   The Oracle Connection Manager listener always listens on the TCP/IP protocol.

   ```
   (ADDRESS=(PROTOCOL=tcp)(HOST=proxysvr)(PORT=1521))
   ```

   Note: Oracle Connection Manager can connect to the database using protocols such as TCP/IP (version 4 and version 6). The protocol TCPS is not supported.

4. Configure the access control rule list (`RULE_LIST`).

   The access control rule list specifies which connections are accepted, rejected, or dropped by the listener. Example 10–1 shows the following rules:

   - In the first rule in the example, the following parameters are set:
     - `src=192.168.2.32/27` is for client connections. It designates the IP address of the client, or source.
     - `DST=sales-server` designates the destination host name. The parameter `ACT` specifies the action, that is, accept, reject, or drop. The parameter `ACTION_LIST` sets attributes for a connection if it is accepted, enabling you to override default parameter settings on a connection-by-connection basis.
   - In the second rule, the following parameters are set:
     - `SRC=192.168.2.32` and `DST=proxysvr` represent the same server, indicating that Oracle Connection Manager and CMCTL must reside on the same computer.

   See Also: "Enabling Access Control" on page 10-4

5. Configure the parameter list (`PARAMETER_LIST`).

   The parameter list sets attributes for an Oracle Connection Manager. Parameters take the following forms:

- If global, then it applies to all Oracle Connection Manager connections unless a rule-level parameter overrides it. To change a global parameter default setting, enter it into the PARAMETER_LIST with an allowable value.
- If a rule-level parameter is enabled in the ACTION_LIST section of the RULE_LIST, then it applies only to connections specified by the rule. It overrides its global counterpart.

Enabling Access Control

Use the RULE_LIST parameter to control client access to designated database servers in a TCP/IP environment. By entering filtering rules under this parameter, you can allow or restrict specific clients access to a database server.

To configure access control, do the following:

1. Open the cman.ora file with a text editor.
2. Update the RULE_LIST parameter using the following format:

```
(RULE_LIST=
 (RULE=(SRC=source_host)
       (DST=destination_host)
       (SRV=service)
       (ACT=accept | reject | drop)))
```

Table 10–1 describes the parameters.

Table 10–1 Rule-Level Parameters

Parameter	Description
SRC	The source host name or IP address of the client. The IP address can be a subnet, such as 192.168.2.62/24.
DST	The destination host name or IP address of the database server. The IP address can be a subnet, such as 192.168.2.62/24.
SRV	The service name of the Oracle Database obtained from the SERVICE_NAME parameter in the initialization parameter file (init.ora).
ACT	To accept, reject, or drop incoming requests based on the preceding three parameters.

You can define multiple rules in the RULE_LIST. The action (ACT) in the first matched RULE is applied to the connection request. If no rules are defined, then all connections are rejected.

In the following example, client computer client1-pc is denied access to the service sales.us.example.com, but client 192.168.2.45 is granted access to the service db1.

```
(RULE_LIST=
  (RULE=(SRC=client1-pc)(DST=sales-server)(SRV=sales.us.example.com)(ACT=reject))
  (RULE=(SRC=192.168.2.45)(DST=192.168.2.200)(SRV=db1)(ACT=accept)))
```

See Also: *Oracle Database Net Services Reference* for additional information about Oracle Connection Manager parameters

Configuring Clients for Oracle Connection Manager

To route clients to the database server through Oracle Connection Manager, configure the `tnsnames.ora` file with a **connect descriptor** that specifies the protocol address of Oracle Connection Manager. This address enables clients to connect to the Oracle Connection Manager computer. The connect descriptor looks similar to the following:

```
sales=
  (DESCRIPTION=
    (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=cman-pc)
      (PORT=1521))
    (CONNECT_DATA=
      (SERVICE_NAME=example.com)))
```

To configure a protocol address for Oracle Connection Manager, do the following:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Service Naming** from Directory or Local menus.
3. Click the plus sign (+) on the toolbar, or select **Create** from the Edit menu.

   The Welcome page of the Net Service Name wizard appears.
4. Enter a name in the Net Service Name field.
5. Click **Next**.

   The Protocol page appears.
6. Select the TCP/IP protocol for Oracle Connection Manager.
7. Click **Next**.

   The Protocol Settings page appears.
8. Specify the Oracle Connection Manager port and protocol. The default port number for Oracle Connection Manager is 1521, and the protocol is TCP/IP.

   See Also: *Oracle Database Net Services Reference* for protocol parameter settings

9. Click **Next**.

   The Service page appears.
10. Enter a service name in the **Service Name** field, and then select the connection type.

    See Also: "About Connect Descriptors" on page 2-5 for additional information about setting the service name string

11. Click **Next**.

Note: Do not click **Test**, because a connection cannot be tested at this point.

---

12. Click **Finish** to save your configuration and dismiss the Net Service Name wizard.

    The new net service name and the Oracle Connection Manager protocol address is added to the Service Naming folder.

Configuring the Oracle Database Server for Oracle Connection Manager

Configuring the database server is a two-part process that involves registering database information remotely with Oracle Connection Manager and, optionally, configuring the server for multiplexing.

This section contains the following topics:

- Configuring Service Registration
- Enabling Session Multiplexing

Configuring Service Registration

To enable the database server to communicate with Oracle Connection Manager, the initialization parameter file `(init.ora)` must contain a descriptor that specifies the listening address of Oracle Connection Manager, and the `tnsnames.ora` file must include the service name entry. The following procedure describes how to configure service registration:

1. Resolve the alias to a service name entry in the `tnsnames.ora` file as follows:

```
cman_listener_address =
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS+(PROTOCL=tcp) (HOST=proxy_server_name)( PORT=1521))))
```

   For example, the alias `listeners_cman` would be resolved to the following entry in the `tnsnames.ora` file:

```
listener_cman=
 (DESCRIPTION=
  (ADDRESS_LIST=
   (ADDRESS=(PROTOCOL=tcp)(HOST=proxyserver1)(PORT=1521))))
```

2. Specify an alias for Oracle Connection Manager in the `init.ora` file as follows. This alias is the one specified in the `tnsnames.ora` file in step 1.

```
REMOTE_LISTENER=cman_listener_address
```

   The alias must be specified because this address is TCP, port 1521 but it is not the default local listening address of TCP, port 1521 of the database server.

   For example, the alias for the Oracle Connection Manager listener running on host, `proxyserver1`, specified in step 1, might look like the following in the `init.ora` file:

```
REMOTE_LISTENER=listener_cman
```

3. After the initialization parameter file is configured with the listening address of Oracle Connection Manager, the **PMON process** can register database information with the Oracle Connection Manager listener. Use the following command to register the change:

```
SQL> ALTER SYSTEM REGISTER
```

See Also: "Registering Information with a Remote Listener" on page 9-11

Enabling Session Multiplexing

To enable Oracle Connection Manager to take advantage of session multiplexing, set the `DISPATCHERS` parameter in the initialization parameter file (`init.ora`) with the attributes `PROTOCOL` and `MULTIPLEX`, similar to the following:

```
DISPATCHERS="(PROTOCOL=tcp)(MULTIPLEX=on)"
```

Table 10–2 lists the parameters to set different levels of multiplexing.

Table 10–2 Session Multiplexing Parameters

Attribute	Description
PROTOCOL	The network protocol for which the dispatcher generates a listening endpoint.
MULTIPLEX	Used to enable session multiplexing If `1`, `on`, `yes`, `true`, or `both` is specified, then multiplexing is enabled for both incoming and outgoing network sessions. If `in` is specified, then multiplexing is enabled for incoming network sessions from the client. If `out` is specified, then multiplexing is enabled for outgoing network sessions. If `0`, `off`, `no`, or `false` is specified, then multiplexing is disabled for both incoming and outgoing network sessions.

Note: You can configure the `DISPATCHERS` parameter using the **Database Configuration Assistant**.

See Also:

- Chapter 11, "Configuring Dispatchers" for additional information about configuring shared servers
- *Oracle Database Net Services Reference* for a complete list of parameters and their default and allowed values

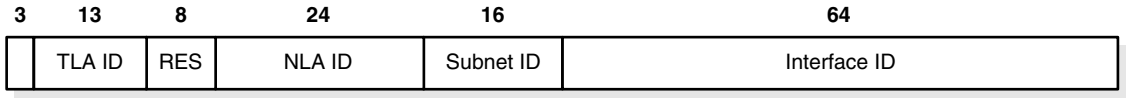
Configuring Oracle Connection Manager as a Bridge for IPv4 and IPv6

In some database connection environments, a client and database may use different versions of the IP protocol so that complete connectivity does not exist. In this case at least two hops in the connection use different versions of the IP protocol. For example, a request passes from an IPv4 source to an IPv6 destination, from an IPv6 source to an IPv4 destination, or from IPv6 to IPv6 through an IPv4 network.

You can use Oracle Connection Manager as a network bridge between **IPv4** and **IPv6**. To serve as a bridge, Oracle Connection Manager must run on a dual-stack host configured with at least one IPv4 interface and at least one IPv6 interface.

Use the Oracle Connection Manager filtering feature to filter based on an IPv6 address. You can base rules on complete or partial IP addresses. Figure 10–1 shows the format of an IPv6 address.

Figure 10–1 IPv6 Address Format



The numbers at the top of the diagram indicate the number of bits in the address. Each hexadecimal character in an IPv6 address represents 4 bits. Bits 4-16 are the Top-Level Aggregation Identifier (TLA ID) portion of the address. Bits 25-49 are the Next-Level Aggregation Identifiers (NLA ID).

For example, in the address `2001:0DB8::203:BAFF:FE0F:C74B`, the binary representation of the first four hexademical characters (`2001`) is as follows:

```
0010000000000001
```

Thus, the first 3 bits in the address are `001`. The TLA ID portion of the address is `0000000000001`.

To create a rules filter for IPv6 address, do the following:

1. Navigate to the `cman.ora` file located in the `ORACLE_HOME/network/admin` directory.
2. Open the `cman.ora` file with a text editor.
3. Create a `RULE` in the `RULE_LIST` based on IPv6 address format.

   For example, assume that the source host is an IPv6-only host with address `2001:0DB8::203:BAFF:FE0F:C74B`, whereas the destination is an IPv4-only host named `SALESL1593`. You configure Oracle Connection Manager as an IPv6-to-IPv4 bridge by creating one of the following rules:

Type of Rule	Description	Example

Filter based on subnet ID	Filtering is based on the 64 bits up to and including the subnet ID	`(RULE = (SRC = 2001:0DB8::203:BAFF:FE0F:C74B/`**`64`**`)`
		`(DST = SALESL1593)`
		`(SRV = SALES)`
		`(ACT = ACCEPT)`
		`(ACTION_LIST = (AUT=ON)(MOCT=10)(MIT=30)(CONN_STATE=YES))`
		`)`

Filter based on NLA ID	Filtering is based on the 48 bits up to and including the NLA ID	`(RULE = (SRC = 2001:0DB8::203:BAFF:FE0F:C74B/`**`48`**`)`
		`(DST = SALESL1593)`
		`(SRV = SALES)`
		`(ACT = ACCEPT)`
		`(ACTION_LIST = (AUT=ON)(MOCT=10)(MIT=30)(CONN_STATE=YES))`
		`)`

Type of Rule	Description	Example
Filter based on TLA ID	Filtering is based on the 16 bits up to and including the TLA ID	(RULE = (SRC = 2001:0DB8::203:BAFF:FE0F:C74B/**16**) (DST = SALESL1593) (SRC = SALES) (ACT = ACCEPT) (ACTION_LIST = (AUT=ON)(MOCT=10)(MIT=30)(CONN_ STATE=YES)))
Filter based on number of bits	Filtering is based on the first 60 bits of the address	(RULE = (SRC = 2001:0DB8::203:BAFF:FE0F:C74B/**60**) (DST = SALESL1593) (SRC = SALES) (ACT = ACCEPT) (ACTION_LIST = (AUT=ON)(MOCT=10)(MIT=30)(CONN_ STATE=YES)))

See Also:

- "TCP/IP Protocol" on page 4-6 to learn more about IPv6
- "About IPv6 Addresses in Connect Descriptors" on page 2-6
- "Configuring Listening Protocol Addresses" on page 9-4

Using the Oracle Connection Manager Control Utility to Administer Oracle Connection Manager

The Oracle Connection Manager Control utility enables you to administer an Oracle Connection Manager. When you issue commands from the operating system, the basic syntax for this utility is as follows:

```
cmctl [command] [argument1 . . . argumentN] [-c instance_name]
```

In the preceding command, `-c` specifies the Oracle Connection Manager instance. You are prompted for the password if one has been set.

Caution: There is an option to specify the password on the command line. However, this exposes the password on the screen, and is a potential security risk. Oracle recommends not using the password option (`-p`) on the command line.

For example, the following command starts the listener, Connection Manager Administration (CMADMIN), and gateway processes for an instance named `cman1`:

```
cmctl STARTUP -c cman1
```

You can also issue Oracle Connection Manager utility commands at the `CMCTL` program prompt. To obtain the prompt, enter `cmctl` with no arguments at the operating system command line. When you run `CMCTL`, the utility is started, and you can enter the necessary commands from the program prompt. The following is an example of the program prompt:

```
cmctl
CMCTL> STARTUP
```

Note: Before issuing the `STARTUP` command, do the following:

- Create the `cman.ora` file. A sample file is located in the `ORACLE_HOME/network/admin/samples` directory after installation of Oracle Connection Manager.
- Run the `ADMINISTER` command to choose an instance to start.

See Also:

- "Understanding Oracle Connection Manager Architecture" on page 5-6 for an overview of the Oracle Connection Manager processes
- *Oracle Database Net Services Reference* for a complete description of Oracle Connection Manager Control utility commands

11

Configuring Dispatchers

The **shared server** architecture enables a database server to allow many user processes to share very few server processes, so the number of users that can be supported is increased. With the shared server architecture, many user processes connect to a **dispatcher**. The dispatcher directs multiple incoming network session requests to a common queue. An idle shared server process from a shared pool of server processes picks up a request from the queue. This means a small pool of server processes can serve a large number of clients. Dispatchers reduce server processes. This is useful when a system is overloaded or has limited memory.

This chapter contains the following topics:

> **See Also:** *Oracle Database Administrator's Guide* for additional information about shared server configuration

Configuring Dispatchers

Shared memory resources for dispatchers, virtual circuits, and shared servers are preconfigured to allow the enabling of shared servers at run time. Database administrators can start dispatchers and shared servers with the SQL `ALTER SYSTEM` statement without having to restart the instance. A dispatcher is started automatically on the TCP/IP protocol when shared server mode is turned on and the dispatchers parameter has not been set. The default configuration for the dispatcher is equivalent to the following `DISPATCHERS` parameter setting in the database initialization parameter file.

```
dispatchers="(PROTOCOL=tcp)"
```

In configurations in which client load is causing a strain on memory and other system resources, database administrators can alleviate load issues by starting shared server resources.

Use the following views to check configurations and monitor dispatchers:

- V$QUEUE: This view contains information about the shared server message queues. This view is only available to the SYS user, and users who have SELECT ANY TABLE system privilege, such as SYSTEM.
- V$DISPATCHER: This view provides information about the dispatcher processes, including name, network address, status, various usage statistics, and index number.
- V$DISPATCHER_CONFIG: This view provides configuration information about the dispatchers.
- V$DISPATCHER_RATE: This view provides rate statistics for the dispatcher processes.

> **See Also:** *Oracle Database Performance Tuning Guide* and *Oracle Database Reference* for additional information these views

Configure the DISPATCHERS parameter if either of the following conditions apply:

- You need to configure a protocol other than TCP/IP
- You want to configure one or more of the optional dispatcher attributes, such as multiplexing.

You can specify the following attributes for the DISPATCHERS parameter. The PROTOCOL attribute is required, and the others are optional. The ADDRESS attribute is used when you need to set a specify port number, such as when using a firewall.

- ADDRESS
- CONNECTIONS
- DESCRIPTION
- DISPATCHERS
- LISTENER
- MULTIPLEX
- POOL
- PROTOCOL
- SERVICE
- SESSIONS
- TICKS

You alter dispatchers configurations with the SQL statement ALTER SYSTEM, or with **Database Configuration Assistant**. After setting this parameter, you do not need to restart the instance.

> **See Also:**
>
> - *Oracle Database Administrator's Guide* for additional information about shared server configuration
> - *Oracle Database Reference* for additional information about configuring the DISPATCHERS parameter and supported attributes
> - *Oracle Database SQL Reference* for additional information about the ALTER SYSTEM statement

Grouping Services by Dispatcher

An Oracle database can be represented by multiple service names. A pool of dispatchers can be allocated exclusively for clients requesting a particular service. This way, the mission critical requests may be given more resources and in effect increase their priority.

For example, the following initialization parameter file sample shows two dispatchers. The first dispatcher services requests for clients requesting `sales.us.example.com`. The other dispatcher services requests only for clients requesting `adminsales.us.example.com`.

```
SERVICE_NAMES=sales.us.example.com
INSTANCE_NAME=sales
DISPATCHERS="(PROTOCOL=tcp)"
DISPATCHERS="(PROTOCOL=tcp)(SERVICE=adminsales.us.example.com)"
```

Enabling Connection Pooling

Connection pooling is a resource utilization feature that enables you to reduce the number of physical network connections to a dispatcher. This is achieved by sharing or pooling a set of connections among the client processes.

To configure connection pooling, set the `DISPATCHERS` parameter in the initialization parameter file with the `POOL` attribute and the following optional attributes:

- `CONNECTIONS`
- `SESSIONS`
- `TICKS`

Consider a system that can support 200 connections for each dispatcher process and has the following:

- 2500 users concurrently connected through TCP/IP
- 1000 sessions concurrently connected through TCP/IP with SSL

To determine the number of dispatchers, use the following formula:

```
CEIL ( maximum concurrent sessions / connections for each dispatcher )
```

In this case, the `DISPATCHERS` attribute for TCP/IP should be set to a minimum of thirteen dispatchers and TCP/IP with SSL should be set to five dispatchers:

```
DISPATCHERS="(PROTOCOL=tcp)(DISPATCHERS=13)"
DISPATCHERS="(PROTOCOL=tcps)(DISPATCHERS=5)"
```

The number of dispatchers can be reduced by implementing connection pooling. Connection pooling allows each dispatcher to handle more sessions. For example, suppose that clients are idle most of the time and one dispatcher can route requests and responses for 2500 TCP/IP sessions, or 1000 TCP/IP with SSL sessions. The following configuration reduces the number of dispatchers required for each protocol and allows the dispatchers to be more fully utilized:

```
DISPATCHERS="(PROTOCOL=tcp)(DISPATCHERS=2)(POOL=on)(TICK=1)
(CONNECTIONS=200)(SESSIONS=2500)"

DISPATCHERS="(PROTOCOL=tcps)(DISPATCHERS=1)(POOL=on)(TICK=1)
(CONNECTIONS=200)(SESSIONS=1000)"
```

In the preceding example, the DISPATCHERS attribute does not have to be specified for the TCP/IP with SSL sessions because the default for DISPATCHERS is 1. The default for CONNECTIONS is the maximum number of connections for each process, which is operating system dependent.

See Also: "Connection Pooling" on page 1-8

Enabling Session Multiplexing

Session multiplexing, available with Oracle Connection Manager, enables multiple client sessions to funnel through a single protocol connection. For example, several user processes can connect to one dispatcher by way of a single connection from Oracle Connection Manager.

Oracle Connection Manager allows communication by users to the dispatcher by way of a shared connection. At any one time, users might need the connection, while other user processes linked to the dispatcher by way of the connection manager process are idle. Session multiplexing is beneficial because it maximizes use of the dispatcher process connections.

Session multiplexing is also useful for database link connections between dispatchers. The limit on the number of sessions for each dispatcher is operating system specific.

To enable session multiplexing, set the attribute MULTIPLEX in the DISPATCHERS parameter to on or an equivalent value.

```
DISPATCHERS="(PROTOCOL=tcp)(MULTIPLEX=on)"
```

See Also: "Enabling Session Multiplexing" on page 10-7 for configuration details and "Enabling Connection Pooling" on page 11-3

Configuring Clients for Environments Using Both Shared Server and Dedicated Server

If a shared server is configured on the server side, and a client connection request arrives when no dispatchers are registered, then the request is processed by a dedicated server process. If you want a particular client always to use a dispatcher, then configure (SERVER=shared) in the CONNECT_DATA section of the connect descriptor. For example:

```
sales=
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (SERVER=shared)))
```

When the (SERVER=shared) attribute is configured and a dispatcher is not available, then the client connection request is rejected, and a message is sent to the client.

If the database is configured for a shared server and a particular client requires a dedicated server, then you can configure the client to use a dedicated server in one of the following ways:

- You can configure a net service name with a connect descriptor that contains (SERVER=dedicated) in the CONNECT_DATA section. For example:

  ```
  sales=
  (DESCRIPTION=
  ```

```
(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
(CONNECT_DATA=
   (SERVICE_NAME=sales.us.example.com)
   (SERVER=dedicated))
```

- You can configure the client profile file, `sqlnet.ora`, with `USE_DEDICATED_SERVER=on`. This adds `(SERVER=dedicated)` to the `CONNECT_DATA` section of the connect descriptor the client uses.

Note: If `USE_DEDICATED_SERVER` is set to `ON`, then existing `(SERVER=`*`value`*`)` entries in connect descriptors are overwritten with `(SERVER=dedicated)`.

See Also:

- "Configuring Advanced Connect Data Parameters" on page 13-5 to set the `SERVER` parameter
- "Routing Connection Requests to a Process" on page 12-4 to set the `USE_DEDICATED_SERVER` parameter
- *Oracle Call Interface Programmer's Guide* and *Oracle Database Administrator's Guide* for additional information about enabling and configuring database resident connection pooling

12

Configuring Profiles

This chapter describes how to configure client and server configuration parameters in **profile**s. A profile is a collection of parameters that specifies preferences for enabling and configuring Oracle Net features on the client or database server. A profile is stored and implemented through the `sqlnet.ora` file.

This chapter contains the following topics:

Overview of Profile Configuration

You can use a profile to:

- Specify the client domain to append to unqualified names
- Prioritize **naming method**s
- Enable logging and tracing features
- Route connections through specific processes
- Configure parameters for an **external procedure**
- Configure **Oracle Advanced Security**
- Use protocol-specific parameters to restrict access to the database

Configuring the Profile During Installation

Oracle Universal Installer launches **Oracle Net Configuration Assistant** after software installation on the client and server. Oracle Net Configuration Assistant configures the order of the naming methods that the computer uses to resolve a **connect identifier** to a **connect descriptor**

Configuration with the Oracle Net Configuration Assistant during installation results in an entry in the `sqlnet.ora` file similar to the following:

```
NAMES.DIRECTORY_PATH=(ezconnect,tnsnames)
```

NAMES.DIRECTORY_PATH specifies the priority order of the naming methods to use to resolve connect identifiers. If the installed configuration is not adequate, then use **Oracle Net Manager** to change the sqlnet.ora configuration.

Configuring Client Attributes for Names Resolution

The following sections describe available client configuration options:

- Specifying a Default Domain for Clients
- Prioritizing Naming Methods
- Routing Connection Requests to a Process

Specifying a Default Domain for Clients

In environments where the client often requests names from a specific domain, it is appropriate to set a default domain in the client sqlnet.ora file with the NAMES.DEFAULT_DOMAIN parameter. This parameter is available to local and external naming methods.

When a default domain is set, it is automatically appended to any unqualified net service name given in the connect string, and then compared to net service names stored in a tnsnames.ora file.

For example, if the client tnsnames.ora file contains a net service name of sales.us.example.com, and the default domain is us.example.com, then the user can enter the following connect string:

```
CONNECT scott@sales
Enter password: password
```

In the preceding example, sales gets searched as sales.us.example.com.

If the connect string includes the domain extension, such as in CONNECT scott@sales.us.example.com, then the domain is not appended.

If a net service name in a tnsnames.ora file is not domain qualified and the NAMES.DEFAULT_DOMAIN parameter is set, then the net service name must be entered with a period (.) at the end of the name. For example, if the domain is set to us.example.com and the client tnsnames.ora file contains a net service name of sales2, then the user would enter the following connect string:

```
CONNECT scott@sales2.
Enter password: password
```

In the preceding example, the client would connect to sales2, not sales2.us.example.com.

The following procedure describes how to specify a default domain:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **Naming**.
4. Click the **Methods** tab.

5. In the Default Domain field, enter the domain.
6. Select **Save Network Configuration** from the File menu.

   The `sqlnet.ora` file should contain an entry that looks similar to the following:

   ```
   NAMES.DEFAULT_DOMAIN=us.example.com
   ```

Prioritizing Naming Methods

After naming methods are configured, as described in Chapter 8, "Configuring Naming Methods", they must be prioritized. Naming methods to resolve a connect identifier are tried in the order they appear in the list. If the first naming method in the list cannot resolve the connect identifier, then the second method in the list is used, and so on.

The following procedure describes how to specify the order of naming methods:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **Naming**.
4. Click the **Methods** tab.

   Table 12–1 describes the naming method values listed in the Methods tab.

Table 12–1 Naming Method Values

Naming Method Value	Description
TNSNAMES	Resolve a **net service name** through the `tnsnames.ora` file on the client. **See Also:** "Configuring the Local Naming Method" on page 8-6
LDAP	Resolve a database service name, net service name, or **net service alias** through a **directory server**. **See Also:** "Configuring the Directory Naming Method" on page 8-12
EZCONNECT	Enable clients to use a TCP/IP connect identifier, consisting of a host name and optional port and service name, or resolve a host name alias through an existing names resolution service or centrally maintained set of `/etc/hosts` files. **See Also:** "Using the Easy Connect Naming Method" on page 8-1
NIS	Resolve service information through an existing network information service (NIS).

5. Select naming methods from the Available Methods list, and then click the right-arrow button.

   The selected naming methods move to the Selected Methods list.

6. Order the naming methods according to the order in which you want Oracle Net to try to resolve the net service name or database service name. Select a naming method in the Selected Methods list, and then click **Promote** or **Demote** to move the selection up or down in the list.

7. Select **Save Network Configuration** from the File menu.

   The `sqlnet.ora` file updates with the `NAMES.DIRECTORY_PATH` parameter:

   ```
   NAMES.DIRECTORY_PATH=(ldap, tnsnames)
   ```

Routing Connection Requests to a Process

Clients and servers can be configured so connection requests are directed to a specific process. The following procedure describes how to route connection requests to a process:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **General**.
4. Click the **Routing** tab.
5. Select the preferred way for routing connections.

   Note: To configure all connections to use a particular server, you select the **Always Use Dedicated Server** option in Oracle Net Manager. This sets the `USE_DEDICATED_SERVER` parameter in the `sqlnet.ora` file to force the listener to spawn a dedicated server for all network sessions from the client. The result is a dedicated server connection, even if a shared server is configured.

6. Choose **File** > **Save Network Configuration**.

   See Also: Table 12–3, " Advanced Settings in sqlnet.ora" for a description of the fields and options

Configuring Database Access Control

You can configure the `sqlnet.ora` file to allow access to some clients and deny access to others. Table 12–2 describes the available settings.

Table 12–2 Access Control Settings in sqlnet.ora

Oracle Net Manager Field/Option	sqlnet.ora File Parameter	Description
Check TCP/IP client access rights	TCP.VALIDNODE_ CHECKING	Specify whether to screen access to the database. If this field is selected, then Oracle Net Manager checks the parameters `TCP.EXCLUDED_NODES` and `TCP.VALIDNODE_CHECKING` to determine which clients to allow access to the database. If this field is deselected, then Oracle Net Manager does not screen clients.
Clients denied access	TCP.EXCLUDED_NODES	Specify which clients using the TCP/IP protocol are denied access to the database.
Clients allowed access	TCP.INVITED_NODES	Specify which clients using the TCP/IP protocol are allowed access to the database.

If the TCP.INVITED_NODES parameter does not include the listener node, then the Listener Control utility cannot connect to the listener. This will prevent start, stop and administration commands from being performed on the listener.

If there are invalid host names or IP addresses listed in the TCP.INVITED_NODES parameter or the TCP.EXCLUDED_NODES parameter, then the Listener Control utility cannot contact the listener.

The following procedure describes how to configure database access control:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

1. In the navigator pane, select **Profile** from the Local menu.
2. From the list in the right pane, select **General**.
3. Click the **Access Rights** tab.
4. Select the **Check TCP/IP client access rights** option.
5. In the Clients allowed to access fields and Clients excluded from access, enter either a host name or an IP address for a client that you want to include or exclude, using commas to delimit entries placed on the same line.

Configuring Advanced Profile Information

Table 12–3 describes the advanced `sqlnet.ora` file settings that you can set.

Table 12–3 Advanced Settings in sqlnet.ora

Oracle Net Manager Field/Option	sqlnet.ora File Parameter	Description
Send operation Time Out	SQLNET.SEND_ TIMEOUT	Specify the time, in seconds, for a database server to complete a send operation to clients to complete after connection establishment. Setting this parameter is recommended for environments in which clients shut down on occasion or abnormally. If the database server cannot complete a send operation in the time specified, then it logs `ORA-12535: TNS:operation timed out` and `ORA-12608: TNS: Send timeout occurred` errors in the `sqlnet.log` file. Without this parameter, the database server continues to send responses to clients that are unable to receive data. You can also set this parameter on the client side to specify the time, in seconds, for a client to complete send operations to the database server after connection establishment. Without this parameter, the client may continue to send requests to a database server already overwhelmed with requests.
Receive operation Time Out	SQLNET.RECV_ TIMEOUT	Specify the time, in seconds, for a database server to wait for client data after connection establishment. A client must send some data within the specified time interval. Setting this parameter is recommended for environments in which clients shut down on occasion or abnormally. If a client does not send any data in the time specified, then the database server logs `ORA-12535: TNS:operation timed out` and `ORA-12609: TNS: Receive timeout occurred` errors to the `sqlnet.log` file. Without this parameter, the database server continues to wait for data from clients that may be down or are experiencing difficulties. You can also use this setting on the client side to specify the time, in seconds, for a client to wait for response data from the database server after connection establishment. Without this parameter, the client may wait for a long period of time for a response from a database server overwhelmed with requests.
Connection Time Out	SQLNET.INBOUND_ CONNECT_TIMEOUT	Specify the time, in seconds, for a client to connect with the database server and provide the necessary authentication information.
Total Send Buffer Size	SEND_BUF_SIZE	Specify the buffer space limit for send operations of sessions.
Total Receive Buffer Size	RECV_BUF_SIZE	Specify the buffer space limit for receive operations of sessions.

Table 12–3 (Cont.) Advanced Settings in sqlnet.ora

Oracle Net Manager Field/Option	sqlnet.ora File Parameter	Description
TNS Time Out Value	SQLNET.EXPIRE_TIME	Specify the time interval, in minutes, to send a probe to verify that client/server connections are active. Setting a value greater than 0 ensures that connections are not left open indefinitely, due to an abnormal client termination. If the probe finds a terminated connection, or a connection that is no longer in use, then it returns an error, causing the server process to exit. This setting is intended for the database server, which typically handles multiple connections at any one time. Limitations on using this terminated connection detection feature are: ■ It is not allowed on bequeathed connections. ■ Though very small, a probe packet generates additional traffic that may downgrade network performance. ■ Depending on which operating system is in use, the server may need to perform additional processing to distinguish the connection probing event from other events that occur. This can also result in downgrading network performance.
Client Registration ID	SQLNET.CLIENT_ REGISTRATION	Specify a unique identifier for a client. This identifier is passed to the listener with any connection request. The identifier can be any string up to 128 characters long.

Table 12–3 (Cont.) Advanced Settings in sqlnet.ora

Oracle Net Manager Field/Option	sqlnet.ora File Parameter	Description
Logon Authentication Protocol Version	SQLNET.ALLOWED_LOGON_VERSION	Define the minimum Oracle Database client release that is allowed to attempt connections to database instances under the control of the given code tree. Each connection attempt is tested. If the client or server does not meet the minimum release specified by its partner, then authentication fails with an `ORA-28040` error. Supported values include: ■ `11` for Oracle Database 11*g* authentication protocols (recommended for strongest protection) ■ `10` for Oracle Database 10*g* authentication protocols ■ `9` for Oracle9*i* authentication protocols ■ `8` for Oracle8*i* authentication protocols The default value is `8`. Note the following implications of setting the value to `11`: ■ To take advantage of the password protections introduced in Oracle Database 11*g*, users must change their passwords. ■ Releases of OCI clients before Oracle Database 10*g* and all versions of JDBC thin clients cannot authenticate to the Oracle database using password-based authentication.
Turn Off UNIX Signal Handling	BEQUEATH_DETACH	Turn on or off UNIX signal handling. Because the client application spawns a server process internally through the Bequeath protocol as a child process, then the client application becomes responsible for cleaning up the child process when it completes. When the server process completes its connection responsibilities, it becomes a terminated process. Signal handlers are responsible for cleaning up these terminated processes. Setting this parameter configures the client profile to pass this process to the UNIX initialization process by disabling signal handlers.
Disable Out-of-Band Break	DISABLE_OOB	Turn on or off out-of-band breaks. If deselected or set to `off`, then Oracle Net can send and receive break messages using urgent data requests provided by the underlying protocol. Once enabled, this feature applies to all protocols used by this client. If selected or set to `on`, then it disables the ability to send and receive break messages using urgent data requests of the underlying protocol.

See Also:

- "Limiting Resource Consumption by Unauthorized Users" on page 14-7 for complete information about configuring this setting
- "Configuring I/O Buffer Space" on page 14-3 for complete information about configuring this setting
- *Oracle Database Advanced Security Administrator's Guide*
- Oracle operating system-specific documentation to determine if the protocol supports urgent data requests. TCP/IP is an example of a protocol that supports this feature.

The following procedure describes how to set advanced features in the `sqlnet.ora` file:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **General**.
4. Click the **Advanced** tab.
5. Enter the values for the fields or options you want to set.
6. Select **Save Network Configuration** from the File menu.

Configuring External Naming Methods

The `sqlnet.ora` file is used to configure required client parameters needed for Network Information Service (NIS) external naming. The following procedure describes how to configure the NIS parameter in the `sqlnet.ora` file:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the File menu.
3. From the list in the right pane, select **Naming**.
4. Click the External tab.
5. Enter `NAMES.NIS.META_MAP` in the Meta Map field.
6. Select **Save Network Configuration** from the File menu.

Configuring Oracle Advanced Security

Oracle Advanced Security enables data encryption and integrity checking, enhanced authentication, and single sign-on. Oracle Advanced Security also provides centralized user management on LDAP-compliant directory servers and certificate-based single sign-on. This functionality relies on the **Secure Sockets Layer (SSL)**.

The following procedure describes how to configure a client or server to use Oracle Advanced Security features:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Profile** from the Local menu.
3. From the list in the right pane, select **Oracle Advanced Security**.

   Each Oracle Advanced Security tab page enables you to configure a separate set of parameters.

4. Select or edit options as applicable.

5. Select **Save Network Configuration** from the File menu.

See Also:

- Choose the Help button on the particular tab page
- Oracle Advanced Security procedural topics in the Oracle Net Manager online help. To access these topics in the online help, select Oracle Advanced Security, and then select the How To option
- *Oracle Database Advanced Security Administrator's Guide* for additional information about configuration

13

Enabling Advanced Features of Oracle Net Services

This chapter describes how to configure the advanced features of Oracle Net Services, including advanced connect data parameters, **load balancing**, **failover**, and connections to non-database services.

This chapter contains the following topics:

Configuring Advanced Network Address and Connect Data Information

A database service can be accessed by several routes and protocol addresses. You configure which routes to use by setting the list of protocol addresses. You configure the order addresses are used by specifying the address parameters. This section contains the following topics:

Creating a List of Listener Protocol Addresses

A database service may be accessed by more than one network route, or protocol address. In the following example, `sales.us.example.com` can connect to `sales.us.example.com` using listeners on either `sales1-server` or `sales2-server`.

```
sales.us.example.com=
 (DESCRIPTION=
  (ADDRESS_LIST=
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

To add a network protocol address to an existing net service name or database service, use either Oracle Enterprise Manager or Oracle Net Manager.

- Using Oracle Enterprise Manager to Add a Network Protocol
- Using Oracle Net Manager to Add a Network Protocol

Using Oracle Enterprise Manager to Add a Network Protocol

The following procedure describes how to add a network protocol to an existing net service name or database service using Oracle Enterprise Manager:

1. Access the Directory Naming or Local Naming page in Oracle Enterprise Manager:
   a. Access the Net Services Administration page in Oracle Enterprise Manager.

      See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

   b. Select **Local Naming** or **Directory Naming** from the Administer list, and then select the Oracle home for the directory server or the location of the local configuration files.
   c. Click **Go**.

      The Directory Naming or Local Naming pages appear.
2. Select the directory service or net service name.

   For Directory Naming, perform a search of the net service name in the Simple Search section, select the net service or database service from the Results list, and then click **Edit**. For Local Naming, select a net service from the list, and then click **Edit**.
3. In the Addresses section, click **Add**.

   The Add Address page appears.
4. From the Protocol list, select the protocol on which the listener is configured to listen. This protocol must also be installed on the client.
5. Enter the appropriate parameter information for the selected protocol in the fields provided.

   See Also: *Oracle Database Net Services Reference* for protocol parameter settings
6. Optionally, in the Advanced Parameters section, specify the I/O buffer space limit for send and receive operations of sessions in the Total Send Buffer Size and Total Receive Buffer Size fields.

   See Also: "Configuring I/O Buffer Space" on page 14-3 for additional information about buffer space
7. Click **OK**.

   The protocol address is added to the Addresses section.
8. Click **OK** to update the address information.

Using Oracle Net Manager to Add a Network Protocol

The following procedure describes how to add a network protocol to an existing net service name or database service using Oracle Net Manager:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, select **Service Naming** from the Directory or Local menus.
3. Select either the net service name or a database service.

   The right pane displays the current destination service and address list.
4. In the Address Configuration box, click the plus sign (+) to add a new address.

   A new Address tab appears:

   a. Select the protocol and enter appropriate address information.

      See Also: *Oracle Database Net Services Reference* for details about protocol address parameters

   b. Optionally, on the Address tab, click **Advanced** to specify the I/O buffer space limit for send and receive operations of sessions in the Total Send Buffer Size and Total Receive Buffer Size fields.

      See Also: "Configuring I/O Buffer Space" on page 14-3 for additional information about buffer space

   c. Order the protocol addresses according to where they should be in the protocol address list with the left-arrow and right-arrow buttons. Unless multiple address options are configured, the first address in the list is contacted.
5. Select **Save Network Configuration** from the File menu.

Configuring Address List Parameters

When a database service is accessible by multiple listener protocol addresses, specify the order in which the addresses are to be used, such as chosen randomly or tried sequentially. Table 13–1 lists the parameters used with multiple protocol addresses.

Table 13–1 Address List Parameters

Parameter	Description
FAILOVER	At connect time, instructs Oracle Net to fail over to a different listener if the first listener fails when set to `on`. The number of addresses in the list determines how many addresses are tried. When set to `off`, instructs Oracle Net to try one address. Connect-time failover is turned `on` by default for multiple address lists (`ADDRESS_LIST`), connect descriptors (`DESCRIPTION`), and multiple connect descriptors (`DESCRIPTION_LIST`). When using a connect descriptor with a `SERVICE_NAME`, ensure that the value is not a `GLOBAL_DBNAME` in any `SID_DESC` entry, or a `SID_NAME` in any `SID_DESC` entry without a `GLOBAL_DBNAME` set.

Table 13–1 (Cont.) Address List Parameters

Parameter	Description
LOAD_BALANCE	When set to `on`, instructs Oracle Net to progress through the list of protocol addresses in a random sequence, balancing the load on the various listeners. When set to `off`, instructs Oracle Net to try the addresses sequentially until one succeeds. Client load balancing is turned `on` by default for multiple connect descriptors (`DESCRIPTION_LIST`).
SOURCE_ROUTE	When set to `on`, instructs Oracle Net to use each address in the order presented until the destination is reached. This parameter is required for reaching the destination using a specific route, that is, by specific computers. This parameter is used to enable connections to **Oracle Connection Manager**.

Note: You cannot set source routing with connect-time failover or client load balancing. Source routing connects to each address in the list sequentially whereas connect-time failover and client load balancing select a single address from a list.

See Also: "Configuring Clients for Oracle Connection Manager" on page 10-5 for additional information about configuring clients for source routing

The following procedure describes how to configure address list parameters:

1. Perform the procedure in "Creating a List of Listener Protocol Addresses" on page 13-1.
2. Use Oracle Enterprise Manager or Oracle Net Manager to configure address list options.
   - For Oracle Enterprise Manager, select the appropriate option in the Connect-time Failover and Client Load Balancing section.
   - For Oracle Net Manager, click **Advanced** in the Address Configuration box. The Address List Options dialog box appears. Select the appropriate option.

   Table 13–2 describes the address list options.

Table 13–2 Address List Options Dialog Box

Option	Parameter Setting
Try each address, in order, until one succeeds.	FAILOVER=on
Try each address, randomly, until one succeeds.	LOAD_BALANCE=on FAILOVER=on
Try one address, selected at random.	LOAD_BALANCE=on
Use each address in order until destination reached.	SOURCE_ROUTE=on
Use only the first address.	LOAD_BALANCE=off FAILOVER=off SOURCE_ROUTE=off

The following example shows a `tnsnames.ora` file configured for client load balancing:

```
sales.us.example.com=
 (DESCRIPTION=
  (ADDRESS_LIST=
  (LOAD_BALANCE=on)
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
  (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

The following example shows a `tnsnames.ora` file configured for connect-time failover:

```
sales.us.example.com=
 (DESCRIPTION=
  (ADDRESS_LIST=
   (LOAD_BALANCE=off)
   (FAILOVER=ON)
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
  (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

Configuring Advanced Connect Data Parameters

The `CONNECT_DATA` section of a connect descriptor defines the destination database service. In the following example, `SERVICE_NAME` defines a service called `sales.us.example.com`:

```
sales.us.example.com=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
   (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)))
```

In addition to the service name, you can optionally configure the connect data information with the parameters described in Table 13–3.

Table 13–3 Advanced Connect Data Settings

Oracle Enterprise Manager/Oracle Net Manager Option	tnsnames.ora File Parameter	Description
Instance Name	INSTANCE_NAME	The database instance to access. The instance name can be obtained from the `INSTANCE_NAME` parameter in the initialization parameter file.
Session Data Unit Size	SDU	The transfer rate of data packets being sent across the network. You can specify the **session data unit (SDU)** size to change the performance characteristics having to do with the packets sent across the network.
Use for Heterogeneous Services	HS	If you want an Oracle database server to access a third-party system through **Heterogeneous Services**, then set this option to `on`.
Oracle RDB Database	RDB_DATABASE	The file name of the Oracle Rdb database.
Type of Service	TYPE_OF_SERVICE	The type of service to use for the Oracle Rdb database.
Global Database Name	GLOBAL_NAME	Oracle Rdb database identifier.

In the following example, the transfer rate for data packets is set:

```
sales.us.example.com=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
   (CONNECT_DATA=
    (SERVICE_NAME=sales.us.example.com)
    (SDU=8192)))
```

Use Oracle Enterprise Manager or Oracle Net Manager to configure advanced CONNECT_DATA parameters for either a net service name or a database service.

Using Oracle Enterprise Manager to Configure Advanced Connect Descriptor Parameters

The following procedure describes how to configure advanced connect descriptor parameters using Oracle Enterprise Manager:

1. Access the Directory Naming or Local Naming page in Oracle Enterprise Manager, as follows:
   a. Access the Net Services Administration page in Oracle Enterprise Manager.

      See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

   b. Select **Local Naming** or **Directory Naming** from the **Administer** list, and then select the Oracle home for the directory server or the location of the local configuration files.
   c. Click **Go**.

      The Directory Naming or Local Naming pages appear.
2. Select the directory service or net service name.

   For **Directory Naming**, search the net service name in the **Simple Search** section by selecting the net service or database service from the **Results** list, and then clicking **Edit**. For **Local Naming**, select a net service from the list, and then click **Edit**.
3. Click the **Advanced** tab.
4. Enter fields or select options as appropriate, and then click **OK**.
5. Click **OK** to update the connect data information.

Using Oracle Net Manager to Configure Advanced Connect Descriptor Parameters

The following procedure describes how to configure advanced connect descriptor parameters using Oracle Net Manager:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2
2. In the navigator pane, select **Service Naming** from Directory or Local menus.
3. Select either the net service name or a database service.

   The right pane displays the current destination service and address list.
4. In the Service Identification box, click **Advanced**.

The Advanced Service Options dialog box appears.

5. Enter fields or select options as appropriate, and then click **OK**.
6. If you are making these changes to the Local folder, then select **Save Network Configuration** from the File menu. Changes to the Directory folder are saved automatically.

Configuring Connection Load Balancing

The **connection load balancing** feature improves connection performance by balancing the number of active connections among multiple **dispatcher**s. In an **Oracle Real Application Clusters (Oracle RAC)** environment, connection pool load balancing also can balance the number of active connections among multiple instances.

Because the **PMON process** can register with remote listeners, a listener can always be aware of all instances and dispatchers, regardless of their location. Depending on the load information, a listener decides which instance and, if shared server is configured, which dispatcher to send the incoming client request.

In a **shared server** configuration, a listener selects a dispatcher in the following order:

1. Least loaded node.
2. Least loaded instance.
3. Least loaded dispatcher for that instance.

In a **dedicated server** configuration, a listener selects an instance in the following order:

1. Least loaded node.
2. Least loaded instance.

If a database service has multiple instances on multiple nodes, then the listener selects the least loaded instance on the least loaded node. If shared server is configured, then the least loaded dispatcher of the selected instance is chosen.

An Oracle Real Application Clusters environment requires that the dispatchers on each instance be cross-registered with the other listeners on the other nodes. This is achieved by the use of the `LISTENER` attribute of the `DISPATCHERS` parameter.

Note: For optimum connection pool load balancing results, the instances that belong to the same database service should be on equivalent hardware and software configurations.

See Also:

- "Registering Information with a Remote Listener" on page 9-11 for complete information about cross-registration
- *Oracle Database Reference* for complete information about the `SERVICE_NAMES` and `INSTANCE_NAME` parameters
- Chapter 11, "Configuring Dispatchers" for complete information about the `LISTENER` attribute

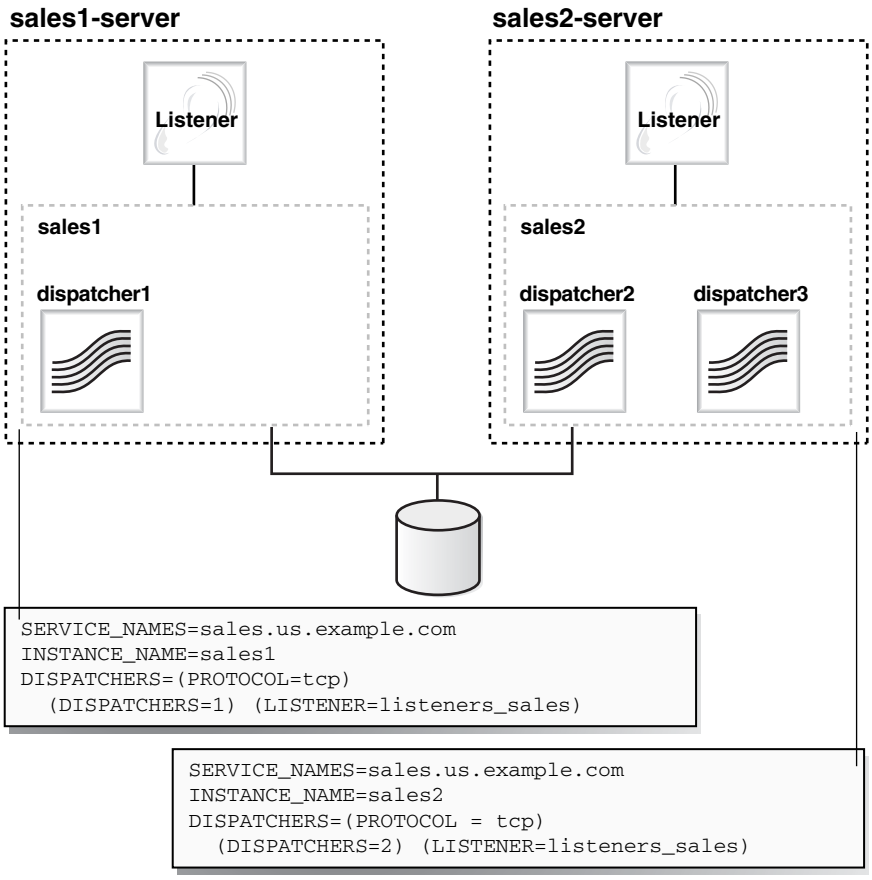
Example of Connection Load Balancing for Shared Server Configuration

Figure 13–1 shows an Oracle Real Application Clusters shared server database with two instances, `sales1` and `sales2`, of the same service, `sales.us.example.com`. The instances `sales1` and `sales2` reside on computers `sales1-server` and `sales2-server`, respectively. `sales1` has one dispatcher and `sales2` has two dispatchers. Listeners named `listener` run on nodes 1 and 2. The `listener` attribute in the `DISPATCHERS` parameter has been configured to allow for service registration of information to both listeners.

In this example, the following load information is registered:

- The one minute load average for each instance is 600 for `sales1` and 400 for `sales2`.
- The number of connections to each instance is 200 for `sales1` and 300 for `sales2`.
- The number of dispatcher connections to each instance is 200 for `dispatcher1`, 100 for `dispatcher2`, and 200 for `dispatcher3`.
- The load average on `sales2-server` (400) is less than the load average on `sales1-server` (600). This can happen if more processing is required on `sales1-server`. The number of connections to `sales1` (200) is the same as that of its only dispatcher, `dispatcher1`.
- The number of connections on `sales2` (300) is the sum of the connections on its two dispatchers, `dispatcher2` (100) and `dispatcher3` (200). Therefore, `sales2` has more connections than `sales1`. In this example, `sales2-server` is the least loaded node, `sales2` is the least loaded instance, and `dispatcher2` is the least loaded dispatcher.

Figure 13–1 Load Balancing Environment for a Shared Server Configuration



The `listeners_sales` value in `(LISTENER=listeners_sales)` can be then resolved through a local `tnsnames.ora` file on both servers as follows:

```
listeners_sales=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
```

Based on the environment, the following actions occur. The numbered actions correspond to the arrows shown in Figure 13–2 on page 13-10:

1. PMON processes for instances `sales1` and `sales2` register with both listeners. The listeners are updated on the load of the instances and dispatchers dynamically.
2. The client sends a connect request. A connect descriptor is configured to try each protocol address randomly until one succeeds:

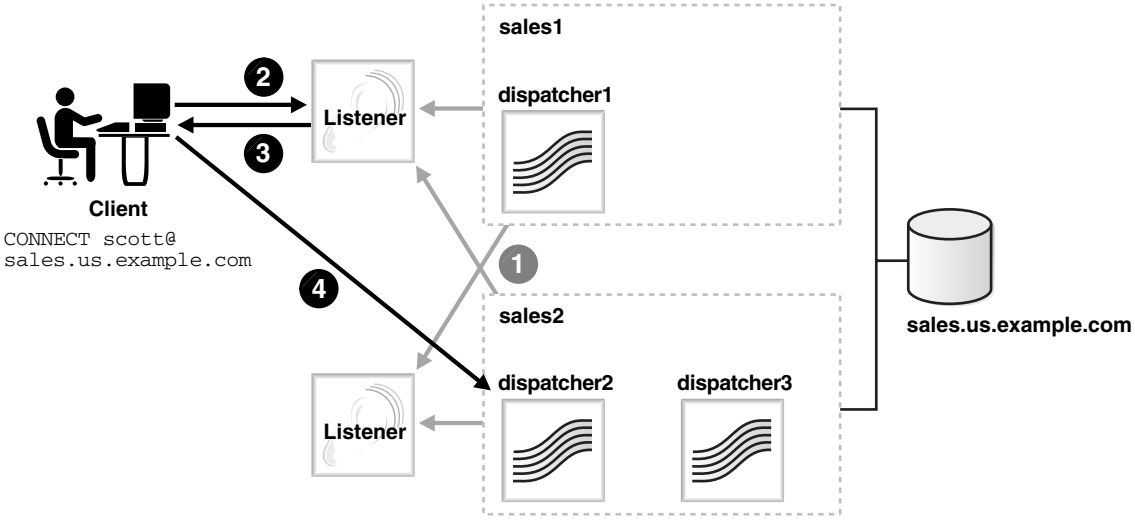
   ```
   sales.us.example.com=
    (DESCRIPTION=
     (LOAD_BALANCE=on)
     (FAILOVER=on)
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521))
     (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
   ```

   The listener on `sales1-server` was randomly chosen to receive the client connect request.

The listener on `sales1-server` compares the load of the instances `sales1` and `sales2`. The comparison takes into account the load on nodes `sales1-server` and `sales2-server`, respectively. Because `sales2-server` is less loaded than `sales1-server`, the listener selects `sales2-server` over `sales1-server`.

3. The listener compares the load on dispatchers `dispatcher2` and `dispatcher3`. Because `dispatcher2` is less loaded than `dispatcher3`, the listener redirects the client connect request to `dispatcher2`.
4. The client connects directly to `dispatcher2`.

Figure 13–2 Load Balancing Example for a Shared Server Configuration



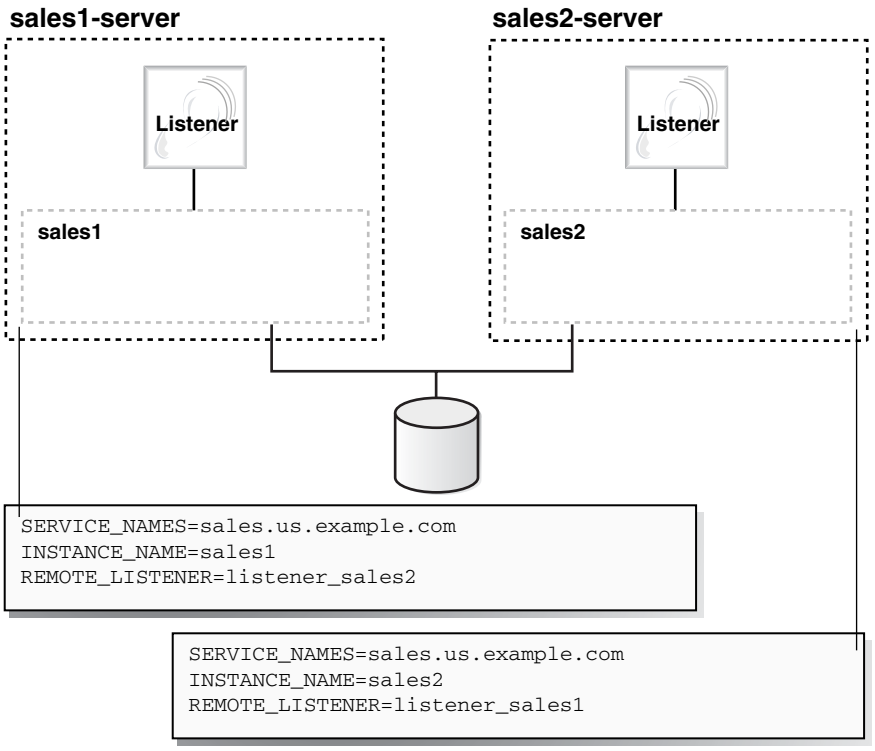
Example of Connection Load Balancing for Dedicated Server Configuration

Figure 13–3 shows an Oracle Real Application Clusters dedicated server database with two instances, `sales1` and `sales2`, of the same service, `sales.us.example.com`. The instances `sales1` and `sales2` reside on computers `sales1-server` and `sales2-server`, respectively. Listeners named `listener` run on nodes 1 and 2. The `REMOTE_LISTENER` initialization parameter has been configured to allow for service registration of information to both listeners.

In this example, the following load information is registered:

- `sales1-server` has a node load average of 450 per minute.
- `sales2-server` has a node load average of 200 per minute.
- `sales1` has 200 connections.
- `sales2` has 150 connections.

Figure 13–3 Load Balancing Environment for a Dedicated Server Configuration



The `listener_sales1` value in `(REMOTE_LISTENER=listener_sales1)` can then be resolved through a local `tnsnames.ora` file on the `sales2-server` as follows:

```
listener_sales1=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521)))
```

The `listener_sales2` value in `(REMOTE_LISTENER=listener_sales2)` can then be resolved through a local `tnsnames.ora` file on the `sales1-server` as follows:

```
listener_sales2=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)))
```

Based on the environment, the following actions occur. The numbered actions correspond to the arrows shown in Figure 13–4:

1. PMON processes for instances `sales1` and `sales2` register with both listeners. The listeners are updated on the load of the instances dynamically.

   Based on the preceding information, `sales2-server` is the least loaded node and `sales2` is the least loaded instance.

2. The client sends a connect request.

   A connect descriptor is configured to try each protocol address randomly until one succeeds:

   ```
   sales.us.example.com=
    (DESCRIPTION=
     (LOAD_BALANCE=on)
     (FAILOVER=on)
   ```

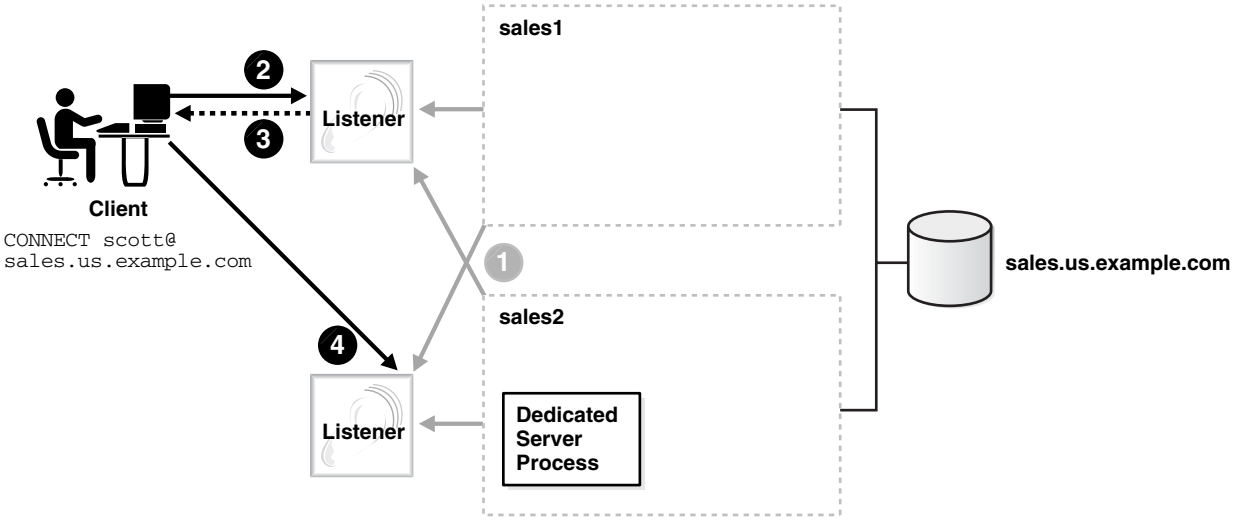
```
(ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521))
(ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)))
```

The listener on `sales1-server` was randomly chosen to receive the client connect request.

The listener on `sales1-server` compares the load of the instances `sales1` and `sales2`. The comparison takes into account the load on nodes `sales1-server` and `sales2-server`, respectively. Because `sales2-server` is less loaded than `sales1-server`, the listener selects `sales2-server` over `sales1-server`.

3. The listener on `sales1-server` redirects the client connect request to the listener on `sales2-server`.
4. The client connects to the listener on `sale2-server`. The listener starts a dedicated server process, and the dedicated server process inherits the connection request from the listener.

Figure 13–4 Load Balancing Example for a Dedicated Server Configuration



Configuring Transparent Application Failover

Transparent Application Failover (TAF) instructs Oracle Net to fail over a failed connection to a different listener. This enables the user to continue to work using the new connection as if the original connection had never failed.

TAF involves manual configuration of a net service name that includes the `FAILOVER_MODE` parameter included in the `CONNECT_DATA` section of the connect descriptor.

This sections contains the following topics:

- About Transparent Application Failover
- What Transparent Application Failover Restores
- About FAILOVER_MODE Parameters
- Implementing Transparent Application Failover
- Verifying Transparent Application Failover

About Transparent Application Failover

TAF is a client-side feature that allows clients to reconnect to surviving databases in the event of a failure of a database instance. Notifications are used by the server to trigger TAF callbacks on the client-side.

TAF is configured using either client-side specified Transparent Network Substrate (TNS) connect string or using server-side service attributes. If both methods are used to configure TAF, then the server-side service attributes supersede the client-side settings. Server-side service attributes are the preferred way to set up TAF.

TAF can operate in one of two modes, Session Failover and Select Failover. Session Failover re-creates lost connections and sessions. Select Failover replays queries that were in progress.

When there is a failure, callback functions are initiated on the client-side using Oracle Call Interface (OCI) callbacks. This works with standard OCI connections as well as Connection Pool and Session Pool connections.

TAF operates with Oracle Data Guard to provide automatic failover. TAF works with the following database configurations to effectively mask a database failure:

- Oracle Real Application Clusters
- Replicated systems
- Standby databases
- Single instance Oracle database

See Also:

- *Oracle Real Application Clusters Installation and Configuration Guide*
- *Oracle Real Application Clusters Administration and Deployment Guide*
- *Oracle Call Interface Programmer's Guide* for more details on callbacks, connection pools, and session pools

What Transparent Application Failover Restores

TAF automatically restores some or all of the following elements associated with active database connections. Other elements may need to be embedded in the application code to enable TAF to recover the connection.

- Client-server database connections: TAF automatically reestablishes the connection using the same connect string or an alternate connect string that you specify when configuring failover.
- Users' database sessions: TAF automatically logs a user in with the same user ID as was used before the failure. If multiple users were using the connection, then TAF automatically logs them in as they attempt to process database commands. Unfortunately, TAF cannot automatically restore other session properties. These properties can be restored by invoking a callback function.
- Completed commands: If a command was completed at the time of connection failure, and it changed the state of the database, then TAF does not resend the command. If TAF reconnects in response to a command that may have changed the database, then TAF issues an error message to the application.
- Open cursors used for fetching: TAF allows applications that began fetching rows from a cursor before failover to continue fetching rows after failover. This is called

select failover. It is accomplished by re-running a SELECT statement using the same snapshot, discarding those rows already fetched and retrieving those rows that were not fetched initially. TAF verifies that the discarded rows are those that were returned initially, or it returns an error message.

- Active transactions: Any active transactions are rolled back at the time of failure because TAF cannot preserve active transactions after failover. The application instead receives an error message until a ROLLBACK is submitted.
- Server-side program variables: Server-side program variables, such as PL/SQL package states, are lost during failures, and TAF cannot recover them. They can be initialized by making a call from the failover callback.

See Also: *Oracle Call Interface Programmer's Guide*

About FAILOVER_MODE Parameters

The FAILOVER_MODE parameter must be included in the CONNECT_DATA section of a connect descriptor. FAILOVER_MODE can contain the parameters described in Table 13–4.

Table 13–4 Additional Parameters of the FAILOVER_MODE Parameter

FAILOVER_MODE Parameters	Description
BACKUP	A different net service name for backup connections. A backup should be specified when using preconnect to pre-establish connections.
DELAY	The amount of time in seconds to wait between connect attempts. If RETRIES is specified, then DELAY defaults to one second. If a callback function is registered, then this parameter is ignored.
METHOD	Setting for fast failover from the primary node to the backup node: ■ basic: Set to establish connections at failover time. This option requires almost no work on the backup server until failover time. ■ preconnect: Set to pre-established connections. This provides faster failover but requires that the backup instance be able to support all connections from every supported instance.
RETRIES	The number of times to attempt to connect after a failover. If DELAY is specified, then RETRIES defaults to five retry attempts. If a callback function is registered, then this parameter is ignored.
TYPE	The type of failover. Three types of Oracle Net failover functionality are available by default to Oracle Call Interface (OCI) applications: ■ session: Set to failover the session. If a user's connection is lost, then a new session is automatically created for the user on the backup. This type of failover does not attempt to recover select operations. ■ select: Set to enable users with open cursors to continue fetching on them after failure. However, this mode involves overhead on the client side in normal select operations. ■ none: This is the default. No failover functionality is used. This can also be explicitly specified to prevent failover from happening.

Note: Oracle Net Manager does not provide support for TAF parameters. These parameters must be set manually.

Implementing Transparent Application Failover

> **Important:** Do not set the GLOBAL_DBNAME parameter in the SID_LIST_*listener_name* section of the listener.ora file. A statically configured global database name disables TAF.

Depending on the FAILOVER_MODE parameters, you can implement TAF in several ways. Oracle recommends the following methods:

- TAF with Connect-Time Failover and Client Load Balancing
- TAF Retrying a Connection
- TAF Pre-establishing a Connection

TAF with Connect-Time Failover and Client Load Balancing

Implement TAF with connect-time failover and client load balancing for multiple addresses. In the following example, Oracle Net connects randomly to one of the protocol addresses on sales1-server or sales2-server. If the instance fails after the connection, then the TAF application fails over to the other node's listener, reserving any SELECT statements in progress.

```
sales.us.example.com=
 (DESCRIPTION=
  (LOAD_BALANCE=on)
  (FAILOVER=on)
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (FAILOVER_MODE=
       (TYPE=select)
       (METHOD=basic)))
```

TAF Retrying a Connection

TAF also provides the ability to automatically retry connecting if the first connection attempt fails with the RETRIES and DELAY parameters. In the following example, Oracle Net tries to reconnect to the listener on sales1-server. If the failover connection fails, then Oracle Net waits 15 seconds before trying to reconnect again. Oracle Net attempts to reconnect up to 20 times.

```
sales.us.example.com=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (FAILOVER_MODE=
       (TYPE=select)
       (METHOD=basic)
       (RETRIES=20)
```

```
      (DELAY=15))))
```

TAF Pre-establishing a Connection

A backup connection can be pre-established. The initial and backup connections must be explicitly specified. In the following example, clients that use net service name `sales1.us.example.com` to connect to the listener on `sales1-server` are also preconnected to `sales2-server`. If `sales1-server` fails after the connection, then Oracle Net fails over to `sales2-server`, preserving any `SELECT` statements in progress. Similarly, Oracle Net preconnects to `sales1-server` for those clients that use `sales2.us.example.com` to connect to the listener on `sales2-server`.

```
sales1.us.example.com=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_NAME=sales1)
     (FAILOVER_MODE=
       (BACKUP=sales2.us.example.com)
       (TYPE=select)
       (METHOD=preconnect))))
sales2.us.example.com=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_NAME=sales2)
     (FAILOVER_MODE=
       (BACKUP=sales1.us.example.com)
       (TYPE=select)
       (METHOD=preconnect))))
```

Verifying Transparent Application Failover

You can query the `FAILOVER_TYPE`, `FAILOVER_METHOD`, and `FAILED_OVER` columns in the `V$SESSION` view to verify that TAF is correctly configured. To view the columns, use a query similar to the following:

```
SELECT MACHINE, FAILOVER_TYPE, FAILOVER_METHOD, FAILED_OVER, COUNT(*)
FROM V$SESSION
GROUP BY MACHINE, FAILOVER_TYPE, FAILOVER_METHOD, FAILED_OVER;
```

The output before failover looks similar to the following:

```
MACHINE              FAILOVER_TYPE FAILOVER_M FAI   COUNT(*)
-------------------- ------------- ---------- --- ----------
sales1               NONE          NONE       NO          11
sales2               SELECT        PRECONNECT NO           1
```

The output after failover looks similar to the following:

```
MACHINE              FAILOVER_TYPE FAILOVER_M FAI   COUNT(*)
-------------------- ------------- ---------- --- ----------
sales2               NONE          NONE       NO          10
```

```
sales2              SELECT        PRECONNECT YES          1
```

Note: You can monitor each step of TAF using an appropriately configured `OCI TAF CALLBACK` function.

See Also:

- *Oracle Call Interface Programmer's Guide*
- *Oracle Database Reference* for additional information about the `V$SESSION` view

Specifying the Instance Role for Primary and Secondary Instance Configurations

The `INSTANCE_ROLE` parameter is an optional parameter for the `CONNECT_DATA` section of a connect descriptor. It enables you to specify a connection to the primary or secondary instance of Oracle Real Application Clusters configurations.

This parameter is useful when:

- You want to explicitly connect to a primary or secondary instance. The default is the primary instance.
- You want to use TAF to preconnect to a secondary instance.

The `INSTANCE_ROLE` parameter supports the following values:

- `primary`: Specifies a connection to the primary instance.
- `secondary`: Specifies a connection to the secondary instance.
- `any`: Specifies a connection to whichever instance has the lowest load, regardless of primary or secondary instance role.

Connection to Instance Role Type

In the following example of the `tnsnames.ora` file, net service name `sales_primary` enables connections to the primary instance, and net service name `sales_secondary` enables connections to the secondary instance.

```
sales_primary=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
       (PORT=1521))
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales2-server)
       (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_ROLE=primary)))
sales_secondary=
 (DESCRIPTION=
  (ADDRESS=
       (PROTOCOL=tcp)
       (HOST=sales1-server)
```

```
      (PORT=1521))
  (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=sales2-server)
      (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_ROLE=secondary)))
```

Connection To a Specific Instance

There are times when Oracle Enterprise Manager and other system management products need to connect to a specific instance regardless of its role to perform administrative tasks. For these types of connections, configure (INSTANCE_NAME=*instance_name*) and (INSTANCE_ROLE=any) to connect to the instance regardless of its role.

In the following example, net service name sales1 enables connections to the instance on sales1-server and sales2 enables connections to the instance on sales2-server. (SERVER=dedicated) is specified to force a dedicated server connection.

```
sales1=
 (DESCRIPTION=
  (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=sales1-server)
      (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_ROLE=any)
     (INSTANCE_NAME=sales1)
     (SERVER=dedicated)))
sales2=
 (DESCRIPTION=
  (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=sales2-server)
      (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_ROLE=any)
     (INSTANCE_NAME=sales2)
     (SERVER=dedicated)))
```

Note: Failover is incompatible with the preceding settings.

TAF Pre-Establishing a Connection

If TAF is configured, then a backup connection can be pre-established to the secondary instance. The initial and backup connections must be explicitly specified. In the following example, Oracle Net connects to the listener on sales1-server and preconnects to sales2-server, the secondary instance. If sales1-server fails after the connection, then the TAF application fails over to sales2-server, the secondary instance, preserving any SELECT statements in progress.

```
sales1.example.com=
 (DESCRIPTION=
  (ADDRESS=
      (PROTOCOL=tcp)
```

```
      (HOST=sales1-server)
      (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_ROLE=primary)
     (FAILOVER_MODE=
       (BACKUP=sales2.example.com)
       (TYPE=select)
       (METHOD=preconnect)))
sales2.example.com=
 (DESCRIPTION=
  (ADDRESS=
      (PROTOCOL=tcp)
      (HOST=sales2-server)
      (PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com)
     (INSTANCE_ROLE=secondary)))
```

Configuring Connections to Third-party Database Services

The following topics describe how to configure connections to third-party database services:

- Default Configuration for External Procedures
- Configuring Oracle Net Services for Oracle Heterogeneous Services
- Configuring Oracle Net Services for an Oracle Rdb Database

Default Configuration for External Procedures

An **external procedure** is a procedure called from another program, written in a different language. An example would be a PL/SQL program calling one or more C routines that are required to perform special-purpose processing.

When an application calls an external procedure, Oracle Database starts an external procedure agent named `extproc`. Using the network connection established by Oracle Database, the application passes the following information to the agent:

- DLL or shared library name
- External procedure name
- Any parameters

The agent then loads the DLL or the shared library, and runs the external procedure and passes back to the application any values returned by the external procedure. The agent must reside on the same computer as the application making the external procedure call.

When you use the default configuration for external procedures, the `extproc` agent is spawned directly by Oracle Database. There are no configuration changes required for either `listener.ora` or `tnsnames.ora`. However, you must define the environment variables to be used by external procedures in the `extproc.ora` file located in the `ORACLE_HOME/hs/admin` directory. If the default configuration for external procedures is not used, then the parameters listed in Table 13–5 must be set.

Table 13–5 External Procedures Settings in listener.ora

Oracle Enterprise Manager Field	listener.ora Parameter	Description
Program Name	PROGRAM	The name of the external procedure agent executable. **Note:** On Microsoft Windows, the executable must reside in the `ORACLE_HOME\bin` directory.

Environment Variables	ENVS	The environment variables to be used by external procedures in the `extproc.ora` file located in the `ORACLE_HOME/hs/admin` directory.
		
		Note: When `extproc.ora` is in use, it precedes the same environment variables of `ENVS` in `listener.ora`.
		
		Syntax: `SET` *`name=value`*
		
		Example: `SET EXTPROC_DLLS=ANY`
		
		Specify the `EXTPROC_DLLS` environment variable to restrict the DLLs that `extproc` is allowed to load. Without the `EXTPROC_DLLS` environment variable, `extproc` loads DLLs from `ORACLE_HOME/lib` on UNIX operating systems and `ORACLE_HOME\bin` on Microsoft Windows.
		
		Set `EXTPROC_DLLS` to one of the following values:
		
		■ Colon-separated list of DLLs
		
		Syntax: `"`*`DLL`*`:`*`DLL`*`"`
		
		This value allows `extproc` to load the specified DLLs and the DLLs from `ORACLE_HOME/lib` on UNIX operating systems and `ORACLE_HOME\bin` on Microsoft Windows. You must enter the complete directory path and file name of the DLLs.
		
		■ `ONLY` (Recommended for maximum security)
		
		Syntax: `"ONLY:`*`DLL`*`:`*`DLL`*`"`
		
		This value allows `extproc` to load only the specified DLLs. You must enter the complete directory path and file name of the DLLs.
		
		■ `ANY`
		
		Syntax: `"ANY"`
		
		Description: This value allows `extproc` to load any DLL. `ANY` disables DLL checking.
		
		Examples:
		
		`"EXTPROC_DLLS=/home/xyz/mylib.so:/home/abc/urlib.so,`
		`LD_LIBRARY_PATH=/private/xpm/lib:/private/mylibs,`
		`MYPATH=/usr/ucb:/usr/local/packages,APL_ENV_FILE=/apl/conf/env.txt"`
		
		`"EXTPROC_DLLS=ONLY:/home/xyz/mylib.so:/home/abc/urlib.so,`
		`LD_LIBRARY_PATH=/private/xpm/lib:/private/mylibs,`
		`MYPATH=/usr/ucb:/usr/local/packages,APL_ENV_FILE=/apl/conf/env.txt"`
		
		`"EXTPROC_DLLS=ANY,LD_LIBRARY_PATH=/private/xpm/lib:/private/mylibs,`
		`MYPATH=/usr/ucb:/usr/local/packages,APL_ENV_FILE=/apl/conf/env.txt"`

Oracle Home Directory	ORACLE_HOME	The Oracle home location of the agent.
SID	SID_NAME	A system identifier for the external procedure agent by any name.

Note: The default configuration for external procedures does not require a network listener to work with Oracle Database and the `extproc` agent. The `extproc` agent is spawned directly by Oracle Database and eliminates the risks that `extproc` might be spawned by Oracle Listener, unexpectedly. This default configuration is recommended for maximum security.

You can change the default configuration for external procedures and have the `extproc` agent spawned by Oracle Listener. To do this, you must perform additional network configuration steps.

Having the `extproc` agent spawned by Oracle Listener is necessary if you use:

- Multi-threaded Agent
- Oracle Database in MTS mode on Microsoft Windows
- The `AGENT` clause of the `LIBRARY` specification or the `AGENT IN` clause of the `PROCEDURE` specification such that you can redirect external procedures to a different `extproc` agent.

Configuring Oracle Net Services for External Procedures

You can change the default configuration for external procedures and have the `extproc` agent spawned by the listener similar to earlier releases of Oracle Database.

Example 13–1 shows a sample configuration in the `listener.ora` file.

Example 13–1 listener.ora File with an External Procedure

```
LISTENER=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=tcp)(HOST=sale-server)(PORT=1521))
      (ADDRESS=(PROTOCOL=ipc)(KEY=extproc)))
SID_LIST_LISTENER=
  (SID_LIST=
    (SID_DESC=
      (GLOBAL_DBNAME=sales.us.example.com)
      (ORACLE_HOME=/oracle)
      (SID_NAME=sales))
    (SID_DESC=
      (SID_NAME=plsextproc)
      (ORACLE_HOME=/oracle)
      (PROGRAM=extproc))
```

Example 13–2 shows a sample configuration in the `tnsnames.ora` file.

Example 13–2 tnsnames.ora File with an External Procedure

```
EXTPROC_CONNECTION_DATA=
 (DESCRIPTION=
   (ADDRESS=(PROTOCOL=ipc)(KEY=extproc))
   (CONNECT_DATA=
    (SID=plsextproc)))
```

Modifying the Default Configuration for External Procedures: To modify the default configuration for external procedures, configure and run a separate or existing listener

to serve external procedures. The following procedure describes how to modify the default configuration:

1. Configure an existing listener to serve external procedures using Oracle Net Configuration Assistant. For most installation types, this listener is named `LISTENER`, as follows:
   a. Access the Oracle Net Administration page in Oracle Enterprise Manager.
   b. Select **Listeners** from the Administer list, and then select the Oracle home that contains the location of the configuration files.
   c. Click **Go**.

      The Listeners page appears.
   d. Select the existing listener created by Oracle Net Configuration Assistant, and then click **Edit**.

      The Edit Listeners page appears.
   e. In the Addresses section, select the protocol address for external procedures, and then click **Add**.
   f. Click the **Other Services** tab.
   g. Select the row representing the service information for external procedures, and then click **Add**.
2. Add service information about `extproc` in the `listener.ora` file, including the parameters described in Table 13–5.

Creating a New Listener to Run External Procedures: To configure and run a separate listener to serve external procedures, remove the external procedure entries for a different listener using Oracle Net Configuration Assistant. The following procedure describes how to create a new listener:

1. Create a listener to exclusively handle external procedures, as follows:
   a. Navigate to the Listeners page.
   b. Click **Create**.

      The Create Listener page appears.
   c. In the Listener Name field, enter a unique listener name, such as `LISTENEREXTPROC`, in the Listener Name field.
2. In the Addresses section, configure an IPC protocol address, as follows:
   a. Click **Add**.

      The Add Address page appears.
   b. From the Protocol list, select **IPC**.
   c. In the Key field, enter a key value of `extproc`.
   d. Click **OK**.

      See Also: "Configuring Listening Protocol Addresses" on page 9-4 for additional information about configuring listener protocol addresses
3. Add service information about `extproc` in the `listener.ora` file, including the parameters described in Table 13–5, as follows:

a. Click the **Other Services** tab.

b. Click **Add**.

The Create Other Service page appears.

c. Enter the following values in the fields:

- `extproc` in the Program Name field.
- The Oracle home where the `extproc` executable resides in the Oracle Home Directory field.
- System identifier, such as `extproc`, in the SID field.

d. In the Environment Variables section, click **Add Another Row**.

e. Enter the `EXTPROC_DLLS` environment variable in the Name field and the directory path and file name of the DLLs in the Value field.

f. Click **OK**.

The Create Listener page appears.

g. Click **OK** to add the listener.

The listener is added to the Listeners page.

The `listener.ora` file updates with information for external procedures, as shown in the following output:

```
LISTENEREXTPROC=
 (DESCRIPTION=
  (ADDRESS=
     (PROTOCOL=ipc)(KEY=extproc)))
SID_LIST_LISTENEREXTPROC=
  (SID_LIST=
    (SID_DESC=
     (PROGRAM=extproc)
     (ENVS="EXTPROC_DLLS=ONLY:/home/xyz/mylib.so:/home/abc/urlib.so,
      LD_LIBRARY_PATH=/private/xpm/lib:/private/mylibs,
      MYPATH=/usr/ucb:/usr/local/packages,APL_ENV_FILE=/apl/conf/env.txt")
     (SID_NAME=extproc)
     (ORACLE_HOME=/oracle)))
```

4. Start the listener for external procedures from a user account with lower privileges than the `oracle` user.

Ensure that this user account does not have general access to the files owned by the `oracle` user. Specifically, this user should not have permission to read or write to database files or to the Oracle server address space. In addition, this user should have read access to the `listener.ora` file, but must not have write access to it.

Running the listener with lower privileges also prevents using the Listener Control `SET` commands to alter the configuration of this listener in the `listener.ora` file. For this reason, Oracle recommends that you complete `listener.ora` file configuration before running the listener.

See Also:

- "Starting Oracle Net Listener and the Oracle Database Server" on page 6-2 for instructions on using the Listener Control utility `START` command to start the listener
- *Oracle Database Application Developer's Guide - Fundamentals* for instruction on enabling external procedure calls

Configuring Oracle Net Services for Oracle Heterogeneous Services

Heterogeneous Services is an integrated component within the Oracle database server, and provides the generic technology for accessing third-party systems from the Oracle database server. Heterogeneous Services enables you to:

- Use Oracle SQL to transparently access data stored in third-party systems as if the data resides within an Oracle database server.
- Use Oracle procedure calls to transparently access third-party systems, services, or application programming interfaces (APIs), from your Oracle distributed environment.

While Heterogeneous Services provides the generic technology in the Oracle database server, a Heterogeneous Services agent is required to access a particular third-party system.

To initiate a connection to the third-party system, the Oracle database server starts an agent process through the listener on the gateway. The following procedure describes how to configure the Oracle database server to be able to connect to the agents:

1. Configure the listener on the gateway to listen for incoming requests from the Oracle database server and spawn Heterogeneous Services agents by configuring the following parameters in the `listener.ora` file:
   - `PROGRAM`: The name of the agent executable
   - `ORACLE_HOME`: The Oracle home location of the agent executable
   - `SID_NAME`: The Oracle System Identifier (SID)
2. Configure the `PROGRAM`, `ORACLE_HOME`, and `SID` parameters in Oracle Enterprise Manager.

   Access the Oracle Net Administration page in Oracle Enterprise Manager.

   See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1
3. Select **Listeners** from the Administer list, and then select the Oracle home that contains the location of the configuration files.
4. Click **Go**.

   The Listeners page appears.
5. Select the listener created by Oracle Net Configuration Assistant, and then click **Edit**.

   The Edit Listeners page appears.
6. In the Addresses section, select the protocol address for external procedures, and then click **Remove**.
7. Click the **Other Services** tab.

8. Click **Add**.

   The Create Other Service page appears.

9. Enter the program name in the Program Name field that will be run to create a gateway, the Oracle home where the agent executable resides in the Oracle Home Directory field, and the SID or service name of the third-party system in the SID field.

10. Click **OK**.

    The Edit Listener page appears.

11. Click **OK** to modify the listener.

    The Listeners page appears.

    The `listener.ora` file updates information about the Heterogeneous Services, as shown in the following:

```
SID_LIST_LISTENER=
 (SID_LIST=
  (SID_DESC=
   (SID_NAME=sybasegw)
   (ORACLE_HOME=/oracle11g)
   (PROGRAM=tg4sybs))
```

12. On the computer where the Oracle database resides, set up a net service name to connect to the listener on the gateway. The connect descriptor must include the `HS=ok` clause to make sure the connection uses Heterogeneous Services, as follows:

    a. Create a net service name that can be used for connections from the Oracle database server to a third-party system.

       See Also: Task 1, "Configure Net Services Names" on page 8-7 for local naming instructions and Task 2, "Create Net Service Names in the Directory" on page 8-13 for directory naming instructions

    b. Use either Oracle Enterprise Manager or Oracle Net Manager to configure `HS=ok`.

       – For Oracle Enterprise Manager, click the **Advanced** tab in the Create Net Service Name page, and then click the **Use for Heterogeneous Services**.

       – For Oracle Net Manager, click **Advanced** in the Service Identification box. The Advanced Service Options dialog box appears. Click **Use for Heterogeneous Services**.

    c. Click **OK** to confirm the change.

       The `tnsnames.ora` file updates with the new net service name configured for Heterogeneous Services, as shown in the following:

```
sybase_gtw=
 (DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=gate-server)(PORT=1521))
  (CONNECT_DATA=
    (SERVICE_NAME=sybasegw)
  )
    (HS=ok))
 )
```

See Also: *Oracle Database Heterogeneous Connectivity Administrator's Guide*

Configuring Oracle Net Services for an Oracle Rdb Database

Oracle Rdb is a database for Digital 64-bit operating systems. Because Oracle Rdb has its own listener, the client interacts with Rdb in the same manner as it does with an Oracle database.

To initiate a connection to an Oracle Rdb, set up a net service name to connect to the Oracle Rdb database using the parameters described in Table 13–6.

Table 13–6 Oracle RDB Database Settings in a Connect Descriptor

Oracle Enterprise Manager Field	tnsnames.ora Parameter	Description
Rdb Database	RDB_DATABASE	The file name of an Oracle Rdb database.
Type of Service	TYPE_OF_SERVICE	The type of service to use for an Oracle Rdb database. It is used by Rdb interface tools. This feature should only be used if the application supports both Oracle Rdb and Oracle database services, and you want the application to load balance between the two.
Global Database Name	GLOBAL_NAME	The Oracle Rdb database. Optional.

The following procedure describes how to configure a client for an Oracle Rdb database:

1. Create a net service name that can be used for connections from the Oracle server to a third-party system.

   See Also: Task 1, "Configure Net Services Names" on page 8-7 for local naming instructions and Task 2, "Create Net Service Names in the Directory" on page 8-13 for directory naming instructions

2. Use either Oracle Enterprise Manager or Oracle Net Manager to set the Oracle Rdb parameters.
   - For Oracle Enterprise Manager, select Local Naming from the Oracle Net Services administration page, then **CREATE LIKE**, and then click the **Advanced** tab on the Create Net Service Name page.
   - For Oracle Net Manager, click **Advanced** in the Service Identification section. The Advanced Service Options dialog box appears.
3. Enter the file name of an Oracle Rdb database in the Rdb Database field.
4. Optionally, enter the global database name in the Global Database Name field, and, if needed, specify the type of service in the Type of Service field, and then click **OK.**

   The `tnsnames.ora` file updates with the new net service name configured for the Oracle Rdb database, similar to the following:

```
alpha5=
  (DESCRIPTION=
    (ADDRESS=...)
    (CONNECT_DATA=
      (SERVICE_NAME=generic)
      (RDB_DATABASE=[.mf]mf_personnel.rdb)
     (GLOBAL_NAME=alpha5)))
```

In the following example, TYPE_OF_SERVICE is used to load balance between an Oracle Rdb database service and an Oracle database service:

```
alpha5=
 (DESCRIPTION_LIST=
  (DESCRIPTION=
    (ADDRESS=...)
    (CONNECT_DATA=
     (SERVICE_NAME=generic)
     (RDB_DATABASE=[.mf]mf_personnel.rdb)
     (GLOBAL_NAME=alpha5))
  (DESCRIPTION=
   (ADDRESS=...)
    (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com))
   (TYPE_OF_SERVICE=oracle_database))
```

14

Optimizing Performance

This chapter describes how to optimize connection performance. This chapter contains the following topics:

Configuring Session Data Unit

Under typical database configuration, Oracle Net encapsulates data into buffers the size of the **session data unit (SDU)** before sending the data across the network. Oracle Net sends each buffer when it is filled, flushed, or when an application tries to read data. Adjusting the size of the SDU buffers relative to the amount of data provided to Oracle Net to send at any one time can improve performance, network utilization, and memory consumption. When large amounts of data are being transmitted, increasing the SDU size can improve performance and network throughput.

The amount of data provided to Oracle Net to send at any one time is referred to as the message size. Oracle Net assumes by default that the message size will normally vary between 0 and 8192 bytes, and infrequently, be larger than 8192 bytes. If this assumption is true, then most of the time, the data is sent using one SDU buffer.

The SDU size can range from 512 bytes to 32767 bytes. The default SDU for the client and a dedicated server is 8192 bytes. The default SDU for a shared server is 32767 bytes.

The actual SDU size used is negotiated between the client and the server at connect time and is the smaller of the client and server values. Configuring an SDU size different from the default requires configuring the SDU on both the client and server computers, unless you are using shared servers. For shared servers, only the client value must be changed because the shared server defaults to the maximum value.

You should consider changing the SDU size when the predominant message size is smaller or larger than 8192. The SDU size should be 70 bytes larger than the predominant message size. If the predominant message size plus 70 bytes exceeds the maximum SDU, then the SDU should be set such that the message size is divided into the smallest number of equal parts where each part is 70 bytes less than the SDU size. To change the default, change the `DEFAULT_SDU_SIZE` parameter in the `sqlnet.ora` file.

For example, if the majority of the messages sent and received by the application are smaller than 8KB, taking into account the 70 bytes for overhead, then setting the SDU to 8KB will likely produce good results. If sufficient memory is available, then using the maximum value for the SDU minimizes the number of system calls and overhead for Oracle Net Services.

Note: Starting with Oracle Database 11*g*, Oracle Net Services optimized bulk data transfer for components, such as Oracle SecureFiles LOBs and Oracle Data Guard redo transport services. The SDU size limit, as specified in the network parameter files, does not apply to these bulk data transfers.

See Also:

- "Session Data Unit Size for Data Transfer Optimization" on page 1-11
- "Statistics Example" on page 16-58

Setting the SDU Size for the Database

To set the SDU size for the database server, configure the following files:

- `sqlnet.ora`

  Configure the `DEFAULT_SDU_SIZE` parameter in the `sqlnet.ora` file, as follows:

  ```
  DEFAULT_SDU_SIZE=32767
  ```

- Initialization parameter file

  If using shared server processes, then set the SDU size in the `DISPATCHERS` parameter in the initialization parameter file, as follows:

  ```
  DISPATCHERS="(DESCRIPTION=(ADDRESS=(PROTOCOL=tcp))(SDU=8192))"
  ```

- `listener.ora`

  If you have configured the listener with a list of targets in the `listener.ora` file, then the value for SDU in the `SID_LIST` element overrides the current setting in the `sqlnet.ora` file when using dedicated server processes.

  ```
  SID_LIST_listener_name=
    (SID_LIST=
      (SID_DESC=
       (SDU=8192)
       (SID_NAME=sales)))
  ```

  The smaller value of the SDU size and the value configured for the client take precedence.

Setting the SDU Size for the Client

To set the SDU size for the client, configure the following files:

- `sqlnet.ora`

  For global configuration on the client side, configure the `DEFAULT_SDU_SIZE` parameter in the `sqlnet.ora` file, as follows:

  ```
  DEFAULT_SDU_SIZE=32767
  ```

- tnsnames.ora

  For a particular connect descriptor, you can specify the SDU parameter in the DESCRIPTION parameter.

```
sales.us.example.com=
(DESCRIPTION=
    (SDU=11280)
      (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
    (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com))
)
```

The SDU size applies to all Oracle Net protocols for the particular connect descriptor.

Determining Bandwidth-delay Product

Bandwidth-delay product is the product of network bandwidth and the round trip time of data going over the network. A simple way to determine the round trip time, is to use a command such as ping from one host to another and use the response times returned by ping.

For example, if a network has a bandwidth of 100 Mbps and a round trip time of 5ms, then the send and receive buffers should be at least (100*10^6) * (5/10^3) bits or approximately 62.5 Kilobytes.

The following equation shows the relationships between the units and factors involved:

```
100,000,000 bits   1 byte   5 seconds
---------------- x ------ x --------- = 62,500 bytes
 1 second          8 bits     1000
```

Setting the SEND_BUF_SIZE and RECV_BUF_SIZE to at least the bandwidth-delay product insures that when large amounts of data are being sent that the network bandwidth will be optimally utilized.

Based on the preceding equation, the bandwidth-delay product of this network link is approximately 64KB. If the largest message used to transfer redo data between a primary database and a standby database is 1MB, then the value for the SEND_BUF_SIZE and RECV_BUF_SIZE parameters could be 1MB. However, if the average message is less, then a setting of 64KB should be sufficient to optimize use of the available bandwidth.

> **See Also:** For information about determining messages sizes, refer to "Statistics Example" on page 16-58

For most network protocols, ensure that the RECV_BUF_SIZE parameter at one end of the network connection, typically at the client, equals the value of the SEND_BUF_SIZE parameter at the other end, typically at the server.

Configuring I/O Buffer Space

Reliable network protocols, such as TCP/IP, buffer data into send and receive buffers while sending and receiving to or from lower and upper layer protocols. The sizes of these buffers affect network performance by influencing flow control decisions.

The RECV_BUF_SIZE and SEND_BUF_SIZE parameters specify sizes of socket buffers associated with an Oracle Net connection. To ensure the continuous flow of data and better utilization of network bandwidth, specify the I/O buffer space limit for receive and send operations of sessions with the RECV_BUF_SIZE and SEND_BUF_SIZE parameters. The RECV_BUF_SIZE and SEND_BUF_SIZE parameter values do not have to match, but should be set according to your environment.

For best performance, the size of the send and receive buffers should be set large enough to hold all the data that may be sent concurrently on the network connection. For a simple database connection, this typically maps to the OCI_PREFETCH_MEMORY size.

Use these parameters with caution as they affect network and system performance. The default values for these parameters are operating system-specific. Following are the defaults for the Linux operating system:

- SEND_BUF_SIZE: 262,144 bytes
- RECV_BUF_SIZE: 262,144 bytes

These parameters are supported for TCP, TCP/IP with SSL, and SDPs. Additional protocols may support these parameters on certain operating systems. The recommended values for these parameters are specified in the installation guide. Refer to operating system-specific documentation of Oracle Net for additional information.

Notes:

- The actual value of the SEND_BUF_SIZE and RECV_BUF_SIZE parameters may be less than the value specified because of limitations in the host operating system or due to memory constraints.
- It is important to consider the total number of concurrent connections that your system must support and the available memory resources. The total amount of memory consumed by these connections depends on the number of concurrent connections and the size of their respective buffers.

See Also: *Oracle Call Interface Programmer's Guide* for additional information about the OCI_PREFETCH_MEMORY parameter

Configuring I/O Buffer Space on the Client

To configure the client, set the buffer space size in the following locations in the specified file:

- Setting only the RECV_BUF_SIZE parameter is typically adequate. If the client is sending large requests, then also set the SEND_BUF_SIZE parameter. These parameters are set in the client's sqlnet.ora file.
- For a particular connect descriptor, you can override the current settings in the client sqlnet.ora file. You can specify the buffer space parameters for a particular protocol address or description in the tnsnames.ora file similar to the following:

```
sales.us.example.com=
 (DESCRIPTION=
   (ADDRESS_LIST=
     (ADDRESS=(PROTOCOL=tcp)(HOST=sales1-server)(PORT=1521)
```

```
          (SEND_BUF_SIZE=11784)
          (RECV_BUF_SIZE=11784))
        (ADDRESS=(PROTOCOL=tcp)(HOST=sales2-server)(PORT=1521)
          (SEND_BUF_SIZE=11784)
          (RECV_BUF_SIZE=11784))
      (CONNECT_DATA=
        (SERVICE_NAME=sales.us.example.com)))
  hr.us.example.com=
   (DESCRIPTION=
     (SEND_BUF_SIZE=8192)
     (RECV_BUF_SIZE=8192)
       (ADDRESS=(PROTOCOL=tcp)(HOST=hr1-server)(PORT=1521))
     (CONNECT_DATA=
       (SERVICE_NAME=hr.us.example.com)))
```

Configuring I/O Buffer Size on the Server

Because the database server writes data to clients, setting the `SEND_BUF_SIZE` parameter on the server-side is typically adequate. If the database server is receiving large requests, then also set the `RECV_BUF_SIZE` parameter.

To configure the database server, set the buffer space size in the `listener.ora` and `sqlnet.ora` files.

In the `listener.ora` file, specify the buffer space parameters for a particular protocol address or for a description. The following is an example of the settings:

```
LISTENER=
(DESCRIPTION=
(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
(SEND_BUF_SIZE=11784)
(RECV_BUF_SIZE=11784))
(ADDRESS=(PROTOCOL=ipc)(KEY=extproc)
(SEND_BUF_SIZE=11784)
(RECV_BUF_SIZE=11784)))
LISTENER2=
(DESCRIPTION=
(SEND_BUF_SIZE=8192)
(RECV_BUF_SIZE=16384)
(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)))
```

The following is an example of the settings in the `sqlnet.ora` file:

```
RECV_BUF_SIZE=65536
SEND_BUF_SIZE=65536
```

Setting the Buffer Size Parameter for Shared Server Processes

If using shared server processes, then you can override the current settings obtained from the server `sqlnet.ora` file by setting the buffer space parameters in the `DISPATCHERS` initialization parameter as follows:

```
DISPATCHERS="(ADDRESS=(PROTOCOL=tcp)(SEND_BUF_SIZE=65536))"
```

Configuring SDP Support for InfiniBand Connections

Oracle Net Services provides support for the Sockets Direct Protocol (**SDP**) for InfiniBand high-speed networks.

SDP is a standard communication protocol for clustered server environments. SDP is an interface between a network interface card and the application. By using SDP,

applications place most of the messaging burden upon the network interface card, freeing the CPU for other tasks. As a result, SDP decreases network **latency** and CPU utilization.

SDP is designed specifically for System Area Networks (SANs). A SAN is characterized by short-distance, high-performance communications between multiple server systems, such as Oracle Application Server or any other third-party middle-tier client and database servers clustered on one switch.

Note: Check with your individual vendor for their version compatibility with Oracle Database 11*g*.

Visit the Oracle Technology Network for additional information about SDP support at

`http://otn.oracle.com/membership`

The following sections describe how to set up Oracle Net support of SDP for middle tier and database server communication. It contains the following topics:

- Prerequisites for Using SDP
- Configuring SDP on the Server
- Configuring SDP on the Client

> **See Also:** "Understanding Performance" on page 1-10 for an overview of supported deployments

Prerequisites for Using SDP

Prior to configuring support for SDP, install the required hardware, and set up InfiniBand hardware and software compatible with OpenFabrics Enterprise Distribution (OFED) 1.4 from a designated vendor on both the application Web server and database server.

During installation of the InfiniBand software, identify the constant that defines SDP or the address family for the system. This can be obtained from the operating system or OFED documentation.

> **See Also:** Vendor documentation for installation information.

Configuring SDP on the Server

To configure the database server, configure an SDP address in the `listener.ora` file on the database server.

Note: If the SDP or address protocol family constant is not 27, the default value for Oracle Net Services, then define the constant in the `SDP.PF_INET_SDP` parameter in the `sqlnet.ora` file.

The following example shows an SDP endpoint that uses port number 1521 on the computer `sales-server`.

```
LISTENER=
  (DESCRIPTION=
    (ADDRESS_LIST=
      (ADDRESS=(PROTOCOL=sdp)(HOST=sales-server)(PORT=1521))
```

```
(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))
(ADDRESS=(PROTOCOL=ipc)(KEY=extproc))))
```

See Also: "Creating a List of Listener Protocol Addresses" on page 13-1

Configuring SDP on the Client

Note: If the SDP or address protocol family constant is not 27, the default value for Oracle Net Services, then define the constant in the `SDP.PF_INET_SDP` parameter in the `sqlnet.ora` file.

The following procedure describes how to configure the Oracle Application Server servers or third-party middle-tier client:

1. If configuring third-party middle-tier client, then upgrade the clients to use Oracle Database 11*g* Client software, as follows:
   a. Run Oracle Universal Installer.
   b. Select **Oracle Database 11*g* Client** from the Available Products page.
2. For both Oracle Application Server servers and third-party middle-tier client, create a net service name to connect to the database server:
   - For Oracle Application Server servers, specify a net service name that uses the same TCP/IP protocol address configured in the `tnsnames.ora` file. For example:

     ```
     sales=
      (DESCRIPTION=
        (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)))
        (CONNECT_DATA=
          (SERVICE_NAME=sales.us.example.com)))
     ```

   - For third-party middle-tier clients, specify a net service name that uses the same SDP address configured in the `tnsnames.ora` file.

     For example:

     ```
     sales=
      (DESCRIPTION=
        (ADDRESS=(PROTOCOL=sdp)(HOST=sales-server)))
        (CONNECT_DATA=
          (SERVICE_NAME=sales.us.example.com)))
     ```

See Also: Chapter 8, "Configuring Naming Methods" for additional information about creating connect descriptors

Limiting Resource Consumption by Unauthorized Users

Unauthorized access to the listener or database server can result in denial-of-service attacks, whereby an unauthorized client attempts to block authorized users' ability to access and use the system when needed. Malicious clients may attempt to flood the listener or database server with connect requests that have the sole purpose of consuming resources, such as connections, processes, or threads. To mitigate these types of attacks, configure limits that constrain the time in which resources can be held prior to authentication. Client attempts to exceed the configured limits result in

connection terminations and an audit trail containing the IP address of the client being logged.

To limit the resource consumption by unauthorized users and enable the audit trail, set time-limit values for the parameters described in Table 14–1. These parameters do not have default values.

Table 14–1 Connect-Timeout Parameters

Parameter	File	Description
INBOUND_CONNECT_TIMEOUT_*listener_name*	listener.ora	The time, in seconds, for the client to complete its connect request to the listener after the network connection had been established. If the listener does not receive the client request in the time specified, then it terminates the connection. In addition, the listener logs the IP address of the client and an ORA-12525: TNS:listener has not received client's request in time allowed error message to the `listener.log` file.
SQLNET.INBOUND_CONNECT_TIMEOUT	sqlnet.ora on the database server	The time, in seconds, for a client to connect with the database server and provide the necessary authentication information. If the client fails to establish a connection and complete authentication in the time specified, then the database server terminates the connection. In addition, the database server logs the IP address of the client and an ORA-12170: TNS:Connect timeout occurred error message to the `sqlnet.log` file. The client receives either an `ORA-12547: TNS:lost contact` or an ORA-12637: Packet receive failed error message.

When specifying values for these parameters, consider the following recommendations:

- Set both parameters to an initial low value.
- Set the value of the `INBOUND_CONNECT_TIMEOUT_`*`listener_name`* parameter to a lower value than the `SQLNET.INBOUND_CONNECT_TIMEOUT` parameter.

For example, you can set `INBOUND_CONNECT_TIMEOUT_`*`listener_name`* to 2 seconds and `INBOUND_CONNECT_TIMEOUT` parameter to 3 seconds. If clients are unable to complete connections within the specified time due to system or network delays that are normal for the particular environment, then increment the time as needed.

See Also:

- "Resolving the Most Common Error Messages for Oracle Net Services" on page 16-10 for a description of error message workarounds
- "Analyzing Listener Log Files" on page 16-27 for additional information about entries in the `listener.log` file
- "Configuring Advanced Profile Information" on page 12-5 for information about configuring these parameters
- "Resolving the Most Common Error Messages for Oracle Net Services" on page 16-10 for a description of error message workarounds

Part III

Testing and Troubleshooting Oracle Net Services

Part III describes how to establish connections, and identify and diagnose problems with Oracle Net Services.

This part contains the following chapters:

15

Testing Connections

After you have configured the network, you should connect and test each component to ensure that the network is functioning properly. Oracle Net Services provides tools to help you test the listener, database, and Oracle Connection Manager.

This chapter contains the following topics:

Testing the Network

The following is the recommended sequence for testing the network:

1. Start and test each listener. To start the listener, use the procedure described in "Starting Oracle Net Listener and the Oracle Database Server" on page 6-2. To test a listener, initiate a connection from a client to any active database controlled by that listener.
2. Start and test each Oracle Connection Manager, if included in your network. To start Oracle Connection Manager, use the procedure described in "Starting Oracle Connection Manager" on page 6-3.

   To test Oracle Connection Manager, initiate a connection from a client to any active database that has been registered with Oracle Connection Manager.
3. Test the server with a loopback test or Oracle Net Manager.

   A **loopback test** uses Oracle Net to go from the database server back to itself, bypassing the Interprocess Communication (IPC). Performing a successful loopback verifies that Oracle Net is functioning on the database server. The following procedure describes how to perform a loopback test using Oracle Net Manager:

   a. Start Oracle Net Manager.

      See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

   b. In the navigator, expand **Directory** or **Local**, and then select **Service Naming**.
   c. Select the net service name or database service.
   d. Choose **Command**, and then select **Test Net Service**.

Testing assumes the listener and database are running. If they are not, then see "Starting Oracle Net Listener and the Oracle Database Server" on page 6-2 to start components.

During testing, a Connection Test dialog box appears, providing status and test results. A successful test results in the following message:

```
The connection test was successful.
```

If the test was successful, then proceed to Step 5.

If the test was not successful, then use the error message to determine further action. For example, if the error message is the following:

```
Initializing first test to use userid: scott, password: tiger
Attempting to connect using userid:  scott
The test did not succeed.
ORA-28000: the account is locked

There may be an error in the fields entered,
or the server may not be ready for a connection.
```

Change the user name to an account known to be unlocked. To change the user name, click **Change Login**. You will be prompted for the password.

e. Click **Close** to close the Connect Test dialog box.

4. Test client with a connection.

To test several different clients in your network, initiate a connection to a database server from each of them using the following command:

```
CONNECT username@connect_identifier
```

Using the TNSPING Utility to Test Connectivity from the Client

The TNSPING utility determines whether the listener for a service on an Oracle Net network can be reached successfully.

If you can connect successfully from a client to a server (or a server to another server) using the TNSPING utility, then it displays an estimate of the round trip time (in milliseconds) it takes to reach the Oracle Net service.

If it fails, then it displays a message describing the error that occurred. This enables you to see the network error that is occurring without the overhead of a database connection.

Use the following command to test connectivity:

```
tnsping net_service_name count
```

In the preceding command, the following arguments are used:

- *net_service_name* must exist in `tnsnames.ora` file or the name service in use, such as NIS.
- *count* determines how many times the program attempts to reach the server. This argument is optional.

If the net service name specified is a database name, then TNSPING attempts to contact the corresponding listener. It does not actually determine whether the database is running. Use SQL*Plus to attempt a connection to the database.

Following are some examples of TNSPING.

> **Note:** Different platforms may have different interfaces, but the program accepts the same arguments. Invoke TNSPING for the display of the proper interface requirements.

Example 15–1 is an example of checking a listener for a database using a net service name of `sales` using the `TNSPING` command.

Example 15–1 Checking a Listener with TNSPING

```
TNSPING sales
```

This produces the following message:

```
TNS Ping Utility for Linux: Version 11.1.0.0.2 on 15-FEB-2009 14:46:28

Copyright (c) 1997, 2009 Oracle Corporation.  All rights reserved.

Used parameter files:
Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL =
TCP)(HOST = sales-server)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME =
sales.us.example.com)))
OK (10 msec)
```

To determine whether the listener for the `sales` database is available, and to specify that TNSPING try to connect eight times and then give up, use the following syntax:

```
tnsping sales 8
```

This command produces the following message:

```
TNS Ping Utility for Linux: Version 11.1.0.0.2 on 15-FEB-2009 14:48:28

Copyright (c) 1997, 2009 Oracle Corporation.  All rights reserved.

Used parameter files:

Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL =
TCP)(HOST = sales-server)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME =
sales.us.example.com)))
OK (10 msec)
OK (0 msec)
OK (10 msec)
OK (0 msec)
OK (10 msec)
OK (10 msec)
OK (10 msec)
OK (0 msec)
```

Example 15–2 is an example of TNSPING attempting to check using an invalid net service name.

Example 15–2 Checking an Invalid Net Service Name with TNSPING

```
tnsping badname
```

This attempt produces the following message:

```
TNS Ping Utility for Linux: Version 11.1.0.0.2 on 15-FEB-2009 14:50:28

Copyright (c) 1997, 2009 Oracle Corporation.  All rights reserved.

Used parameter files:

TNS-03505: Failed to resolve name
```

Example 15–3 is an example of output when using TNSPING to check a name that is valid, but that resolves to an address where no listener is located (for example, the listener may not be started).

Example 15–3 Checking Valid Net Service Name but No Listener with TNSPING

```
TNS Ping Utility for Linux: Version 11.1.0.0.2 on 15-FEB-2009 14:52:28

Copyright (c) 1997, 2009 Oracle Corporation.  All rights reserved.

Used parameter files:
Used TNSNAMES adapter to resolve the alias
Attempting to contact (DESCRIPTION = (ADDRESS_LIST = (ADDRESS = (PROTOCOL =
TCP)(HOST = sales-server)(PORT = 1521))) (CONNECT_DATA = (SERVICE_NAME =
sales.us.example.com)))
TNS-12541: TNS:no listener
```

Using the TRCROUTE Utility to Test Connectivity from the Client

The Trace Route Utility (TRCROUTE), in Linux and UNIX environments, enables administrators to discover the path or route a connection is taking from a client to a server. If TRCROUTE encounters a problem, then it returns an error stack to the client instead of a single error. These additional error messages make troubleshooting easier.

TRCROUTE is different from TNSPING in that it travels as a special type of connect packet, and is routed as such. As it travels toward its destination, the TRCROUTE connect packet collects the TNS addresses of every node it travels through. If an error occurs, then TRCROUTE collects error information that shows where the error occurred. The TRCROUTE displays the information collected on the client screen. You can redirect the TRCROUTE output to a file, and print it if you want.

The TRCROUTE uses minimal resources. It gathers information in the connect data of a special connect packet; standard connect packets are not affected.

The server is not affected by TRCROUTE. The listener receives and processes the TRCROUTE connect packet. It returns the information to the client by putting it into a refuse packet. The server does not need to start any new processes or deal with dummy connections.

To use the TRCROUTE utility, enter the following command:

```
trcroute net_service_name
```

Example 15–4 shows a successful trace route packet that traveled from a client to a listener.

Example 15–4 Successful Trace Route

```
trcroute sales
Trace Route Utility for Linux: Version 11.2.0.0.2 on 15-FEB-2009 14:35:05
```

```
Copyright (c) 1999, 2009 Oracle Corporation.  All rights reserved.

Route of TrcRoute:
------------------

Node: Client            Time and address of entry into node:
-------------------------------------------------------------
09-NOV-2008 21:48:48 ADDRESS= PROTOCOL=TCP  HOST=10.150.21.136  PORT=14001

Node: Server            Time and address of entry into node:
-------------------------------------------------------------
09-NOV-2008 21:48:05 ADDRESS= PROTOCOL=TCP  HOST=10.150.21.136  PORT=14001
```

Example 15–5 shows an unsuccessful trace route packet that could not reach the listener because the listener was not up.

Example 15–5 Trace Route with Error

```
Trace Route Utility for Linux: Version 11.2.0.0.2 on 15-FEB-2009 14:35:05

Copyright (c) 1999, 2009 Oracle Corporation.  All rights reserved.

Route of TrcRoute:
------------------

Node: Client            Time and address of entry into node:
-------------------------------------------------------------
25-FEB-2002 14:43:05 ADDRESS= PROTOCOL=TCP  HOST=sales-server  PORT=1521

TNS-12543: TNS:unable to connect to destination
TNS-12541: TNS:no listener
TNS-12560: TNS:protocol adapter error
TNS-03601: Failed in route information collection
```

16

Troubleshooting Oracle Net Services

Oracle Net Services provides methods for understanding, testing and resolving network problems. Oracle Database includes utilities, and log and trace files for testing and diagnosing network connection and problems. The TNSPING and TRCROUTE utilities test connectivity. The log and trace files keep track of the interaction between network components as errors occur. Evaluating this information will help you to diagnose and troubleshoot network problems.

This chapter describes common testing procedures and network errors, and outlines procedures for resolving problems. It also describes methods for logging and tracing error information to diagnose and troubleshoot more complex network problems. This chapter contains the following topics:

Understanding Automatic Diagnostic Repository

The **automatic diagnostic repository** (ADR) is a systemwide tracing and logging central repository. The repository is a file-based hierarchical data store for depositing diagnostic information, including network tracing and logging information.

The ADR home is the unit of the ADR directory that is assigned to an instance of an Oracle product. Each database instance has its own ADR home. Similarly, each listener, Oracle Connection Manager, and client instance has its own ADR home.

The location of an ADR home is given by the following path, which starts at the ADR base directory:

```
diag/product_type/product_id/instance_id
```

Table 16–1 lists the values of the path components for an Oracle Net Listener instance.

Table 16–1 ADR Home Path Components for an Oracle Net Listener Instance

Path Component	Value for Oracle Net Listener
product_type	tnslsnr
product_id	host name
instance_id	listener alias name

Figure 16–1 illustrates the directory hierarchy of the ADR for an Oracle Net Listener instance. Other ADR homes for other Oracle products or components (such as Automatic Storage Management (ASM) or Oracle Database) can exist within this hierarchy, under the same ADR base.

Figure 16–1 Directory Structure for an Oracle Net Listener Instance

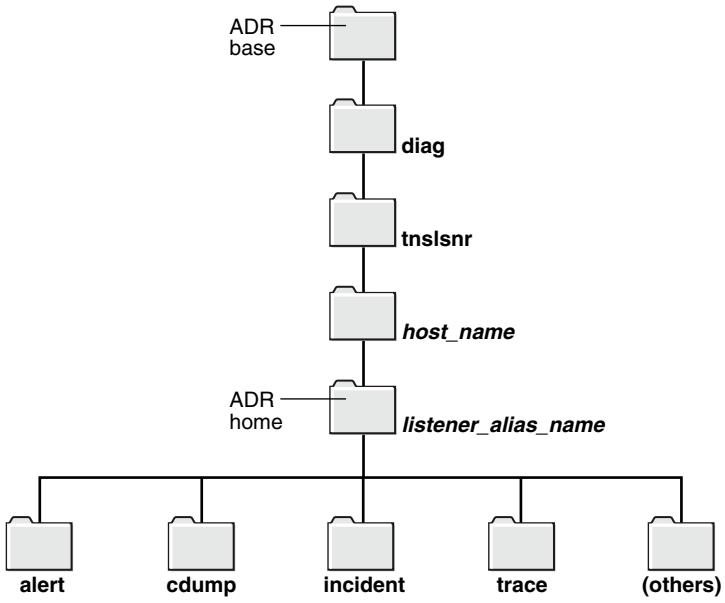


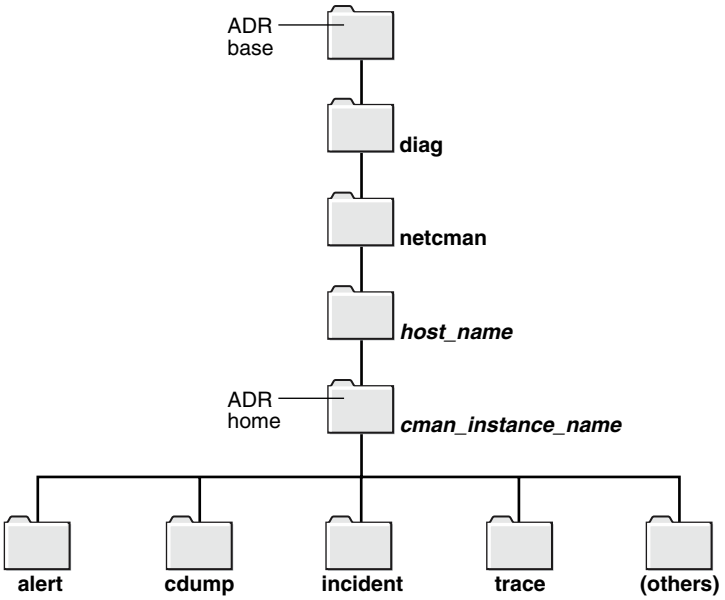
Table 16–2 lists the values of the path components for an Oracle Connection Manager instance.

Table 16–2 ADR Home Path Components for a Oracle Connection Manager Instance

Path Component	Value for Oracle Connection Manager
product_type	netcman
product_id	host name
instance_id	Oracle Connection Manager instance name

Figure 16–2 illustrates the directory hierarchy of the ADR for an Oracle Connection Manager instance. Other ADR homes for other Oracle products or components (such as ASM or Oracle Database) can exist within this hierarchy, under the same ADR base.

Figure 16–2 Directory Structure for a Oracle Connection Manager Instance



Within the ADR home directory are subdirectories where each instance, such as the database, listener, Oracle Connection Manager, or client, stores diagnostic data. Table 16–3 lists some of these subdirectories and their contents.

Table 16–3 ADR Home Subdirectories

Subdirectory Name	Contents
alert	The XML-formatted log named log.xml
cdump	Core files
incident	Multiple subdirectories, where each subdirectory is named for a particular incident, and where each contains dumps pertaining only to that incident
trace	Background and server process trace files, SQL trace files, and text version of the log.xml file in the alert directory
(others)	Other subdirectories of ADR home, which store incident packages, health monitor reports, and other information

The `ADR_BASE` is the physical location in which one or more ADR homes are placed. Conceptually, it is the root directory of ADR.

Non-ADR (meaning that the `DIAG_ADR_ENABLED` parameter is set to `OFF`) diagnostic and tracing methods are still current and applicable but the parameters are ignored if ADR is enabled. ADR is enabled by default.

Diagnostic parameters are found in the following configuration files:

- `sqlnet.ora` for clients.
- `listener.ora` for listeners.
- `cman.ora` for connection managers.

Table 16–4 compares usage of diagnostic parameters found in the `sqlnet.ora` file used in both ADR and non-ADR-based diagnostics.

Table 16–4 sqlnet.ora File Diagnostic Parameter Comparison

Parameter	DIAG_ADR_ ENABLED=OFF	DIAG_ADR_ ENABLED=ON
ADR_BASE	Disabled	Enabled
TRACE_LEVEL_CLIENT	Enabled	Enabled
TRACE_LEVEL_SERVER	Enabled	Enabled
TRACE_DIRECTORY_CLIENT	Enabled	Disabled
TRACE_FILE_CLIENT	Enabled	Disabled
TRACE_UNIQUE_CLIENT	Enabled	Disabled
LOG_DIRECTORY_CLIENT	Enabled	Disabled
LOG_FILE_CLIENT	Enabled	Disabled
LOG_DIRECTORY_SERVER	Enabled	Disabled
TRACE_DIRECTORY_SERVER	Enabled	Disabled
TRACE_FILE_SERVER	Enabled	Disabled

Table 16–5 compares usage of diagnostic parameters found in the `listener.ora` file used in both non-ADR and ADR-based diagnostics.

Table 16–5 listener.ora File Diagnostic Parameter Comparison

Parameter	DIAG_ADR_ ENABLED=OFF	DIAG_ADR_ ENABLED=ON
ADR_BASE_listener_name	Disabled	Enabled
LOGGING_listener_name	Enabled	Enabled
TRACE_LEVEL_listener_name	Enabled	Enabled
TRACE_TIMESTAMP_listener_name	Enabled	Enabled
LOG_DIRECTORY_CLIENT_listener_name	Enabled	Disabled
LOG_FILE_CLIENT_listener_name	Enabled	Disabled
TRACE_DIRECTORY_CLIENT_listener_name	Enabled	Disabled
TRACE_FILELEN_listener_name	Enabled	Disabled
TRACE_FILENO_listener_name	Enabled	Disabled

Table 16–6 compares usage of diagnostic parameters found in the `cman.ora` file used in both non-ADR and ADR-based diagnostics.

Table 16–6 cman.ora File Diagnostic Parameter Comparison

Parameter	DIAG_ADR_ ENABLED=OFF	DIAG_ADR_ ENABLED=ON
ADR_BASE	Disabled	Enabled
LOG_LEVEL	Enabled	Enabled
TRACE_LEVEL	Enabled	Enabled
TRACE_TIMESTAMP	Enabled	Enabled
LOG_DIRECTORY	Enabled	Disabled

Table 16–6 (Cont.) cman.ora File Diagnostic Parameter Comparison

Parameter	DIAG_ADR_ ENABLED=OFF	DIAG_ADR_ ENABLED=ON
TRACE_DIRECTORY	Enabled	Disabled
TRACE_FILELEN	Enabled	Disabled
TRACE_FILENO	Enabled	Disabled

See Also:

- *Oracle Call Interface Programmer's Guide* for information about the location of the client ADR Home
- *Oracle Database Administrator's Guide* for additional information about ADR
- *Oracle Database Net Services Reference* for descriptions of the following diagnostic parameters

ADRCI: ADR Command Interpreter

ADRCI is a command-line tool that is part of the fault diagnosability infrastructure introduced in Oracle Database 11*g*. ADRCI enables you to:

- View diagnostic data within ADR
- Package incident and problem information into a zip file for transmission to Oracle Support Services

Diagnostic data includes incident and problem descriptions, trace files, dumps, health monitor reports, alert log entries, and so on.

ADRCI has a rich command set, and can be used in interactive mode or within scripts. In addition, ADRCI can run scripts of ADRCI commands in the same way that SQL*Plus runs scripts with SQL and PL/SQL commands.

To view trace files using ADRCI, enter `ADRCI` at a command line. The following are common ADRCI commands to check a client:

Client Side

```
adrci>> SHOW ALERT
adrci>> SHOW BASE -product client
adrci>> SET BASE -product client
adrci>> SHOW TRACEFILE
adrci>> SHOW TRACE trace_file.trc
adrci>> SHOW SPOOL
```

In the preceding commands, `SHOW ALERT` will show the log.xml file in a text editor, such as VI. `SHOW BASE -product client` displays the value of `ADR_BASE` for the client. Use that value for *client* in the `SET BASE` command.

The following are common ADRCI commands to check a server:

Server Side

```
adrci>> SHOW BASE
adrci>> SHOW TRACEFILE
adrci>> SHOW TRACE trace_file.trc
```

Other ADRCI command options are available for a more targeted Oracle Net trace file analysis. Type `HELP` at the `adrci` prompt for help documentation.

> **See Also:** *Oracle Database Utilities* for additional information about ADRCI

Diagnosing Oracle Net Services

Any underlying fault, noticeable or not, is reported by Oracle Net Services with an error number or message. The error number and message provide useful information for diagnosing the problem, but may not always identify the actual problem. This section helps you determine which parts of Oracle Net Services do function properly rather than the parts which do not work. It also helps you to decide in which of the following categories the fault belongs:

- Oracle software
- Operating system layer
- Other network layers

Testing the various network layers progressively should, in most cases, uncover any problem.

This section contains the following topics:

- Diagnosing Server Problems
- Diagnosing Client Problems

Diagnosing Server Problems

To start diagnosing server problems, you should answer the following questions:

- Is any other system (workstation/server) able to connect to the server using Oracle Net?
- Has the server, database, or listener configuration remained the same for some time?

If you answered yes to any of the preceding questions, then go to "Diagnosing Client Problems" on page 16-7.

If you are unsure, or answered no to any of the preceding questions, then continue. Diagnosing Oracle Net Services on the server involves the following tasks:

- Task 1, "Verify the Database is Running"
- Task 2, "Perform a Loopback Test"

Task 1 Verify the Database is Running

To check that the database is up, log in to the database and connect with a valid username and password. For example:

```
SQLPLUS system
Enter password: password
```

A message appears, confirming that you are connected with the database. If you receive the following errors, then ask your Database Administrator to assist you:

- `ORA-1017: invalid U/P`
- `ORA-1034: Oracle not available`

Task 2 Perform a Loopback Test

A loopback test uses Oracle Net to go from the database server back to itself, bypassing the Interprocess Communication (IPC). Performing a successful loopback verifies that Oracle Net is functioning on the database server.

To perform a **loopback test** from the server to the database:

1. Ensure that the `listener.ora`, `tnsnames.ora`, and `sqlnet.ora` files exist in the correct locations, as described in "Using Localized Management" on page 3-1.
2. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

3. In the navigator, expand **Directory** or **Local** > **Service Naming**.
4. Select the net service name or database service.
5. Choose **Command** > **Test Net Service**.

   Testing assumes the listener and database are running. If they are not, then see "Starting Oracle Net Listener and the Oracle Database Server" on page 6-2 to start components.

   During testing, a Connection Test dialog box appears, providing status and test results. A successful test results in the following message:

   ```
   The connection test was successful.
   ```

   If the test was successful, then proceed to Step 6.

   If the test was not successful, then do the following:

   a. Ensure the database and listener are running, and then click **Test**.
   b. Click **Change Login** to change the username and password for the connection, and then click **Test**.

   - If the loopback test passes, then go to "Diagnosing Client Problems".
   - If the loopback test continues to fail, then contact Oracle Support Services.

6. Click **Close** to close the Connect Test dialog box.

Diagnosing Client Problems

Verify at least one of the following statements. This will help you decide if it is a client problem.

- The database server passed a loopback test, showing that the connection worked.
- Other computers connect also using Oracle Net Services to this same database.
- Connections from this workstation worked before making changes on this computer, such as the installation of a new product or modification to the network configuration.

The following procedure describes how to perform diagnostics on the client:

1. Check that you have installed the same protocol support that was installed on the database server.

   On Linux and UNIX platforms you can use the `ADAPTERS` utility to verify protocol support. On the database server, run the following command from the `ORACLE_`

HOME/bin directory to display the protocol support, naming methods, and security options linked with the oracle executable:

```
adapters ./oracle
```

The adapters utility displays output similar to the following:

```
Installed Oracle Net transport protocols are:

    IPC
    BEQ
    TCP/IP
    SSL
    RAW
    SDP/IB

Installed Oracle Net naming methods are:

    Local Naming (tnsnames.ora)
    Oracle Directory Naming
    Oracle Host Naming
    NIS Naming

Installed Oracle Advanced Security options are:

    RC4 40-bit encryption
    RC4 56-bit encryption
    RC4 128-bit encryption
    RC4 256-bit encryption
    DES40 40-bit encryption
    DES 56-bit encryption
    3DES 112-bit encryption
    3DES 168-bit encryption
    AES 128-bit encryption
    AES 192-bit encryption
    AES 256-bit encryption
    MD5 crypto-checksumming
    SHA crypto-checksumming (for FIPS)
    SHA-1 crypto-checksumming
    Kerberos v5 authentication
    RADIUS authentication
```

On the client, run the adapters command from the ORACLE_HOME/bin directory to display the configured Oracle protocol support, naming methods, and security options. The ADAPTERS utility displays output similar to the following:

```
Installed Oracle Net transport protocols are:

    IPC
    BEQ
    TCP/IP
    SSL
    RAW
    SDP/IB

Installed Oracle Net naming methods are:

    Local Naming (tnsnames.ora)
    Oracle Directory Naming
    Oracle Host Naming
    NIS Naming
```

```
Installed Oracle Advanced Security options are:

    RC4 40-bit encryption
    RC4 56-bit encryption
    RC4 128-bit encryption
    RC4 256-bit encryption
    DES40 40-bit encryption
    DES 56-bit encryption
    3DES 112-bit encryption
    3DES 168-bit encryption
    AES 128-bit encryption
    AES 192-bit encryption
    AES 256-bit encryption
    MD5 crypto-checksumming
    SHA-1 crypto-checksumming
    Kerberos v5 authentication
    RADIUS authentication
```

Note: RAW is an internal protocol used by Oracle Net.

See Also: *Oracle Database Administrator's Guide* for additional information about the `adapters` utility

2. Check base connectivity for underlying network transport. Oracle Net technology depends on the underlying network for a successful connection.

Protocol	Verify that you can...
TCP/IP	Use terminal emulation or file transfer utilities, (PING, FTP, TELNET) from the client to the database server.
Named Pipes	■ See other computers or servers on the Microsoft network. ■ Ensure that you are able to share drives within the network.

3. Ensure that the Oracle Net foundation layer and the appropriate Oracle protocol support are present by verifying that all Oracle Net Services software has been installed for the client.
4. Ensure that the client computer has the `tnsnames.ora` and the `sqlnet.ora` files in the correct locations.

See Also: "Using Localized Management" on page 3-1

If you have any other working client computers connecting to the selected Oracle Database, then back up your existing files and copy both the working `tnsnames.ora` and `sqlnet.ora` files from the working client computer to the non-working clients. This eliminates the possibility of errors in the files.

5. Test the Oracle Net foundation layer. You can test using the following command to connect to SQL*Plus:

```
sqlplus user/password@connect_string
```

> **Note:** Do not use the TNSPING utility. The TNSPING utility works like the TCP/IP PING utility and does not create and open a socket, nor does it connect with the listener. It ensures that the listener is present on the database server.

6. If the connection still fails, then do the following:
   - Use tracing, as described in section "Troubleshooting Network Problems Using Log and Trace Files" on page 16-20
   - Check the Oracle Support Web site for a specific diagnostics bulletin on the error received
   - Contact Oracle Support Services

Resolving the Most Common Error Messages for Oracle Net Services

Due to the complexity of network communications, network errors may originate from a variety of sources, for a variety of reasons. If an error occurs, then applications such as SQL*Plus, that depend on network services from Oracle Net Services, normally generate an error message.

A list of the most common network error messages follows:

- ORA-03113: TNS:end-of-file on communication channel
- ORA-12154: TNS:could not resolve the connect identifier specified
- ORA-12170: TNS:Connect timeout occurred
- TNS-12500/ORA-12500: TNS: listener failed to start a dedicated server process
- ORA-12514: TNS:listener does not currently know of service requested in connect descriptor
- ORA-12520: TNS:listener could not find available handler for requested type of server
- ORA-12521: TNS:listener does not currently know of instance requested in connect descripto
- ORA-12525: TNS:listener has not received client's request in time allowed
- ORA-12533: TNS:illegal ADDRESS parameters
- TNS-12540/ORA-12540: TNS:internal limit restriction exceeded and TNS-00510: Internal limit restriction exceeded
- TNS-12541/ORA-12541: TNS:no listener
- TNS-12549/ORA-12549: TNS:operating system resource quota exceeded and TNS-00519: Operating system resource quota exceeded
- TNS-12560/ORA-12560: TNS:protocol adapter error occurred

For information about the specific error messages, use the Oracle error tool `oerr`, by entering the following at any command line:

```
oerr code error_number
```

In the preceding command, *code* is the type of message, such as ORA and TNS, and *error_number* is the number associated with the error message.

See Also: *Oracle Database Error Messages* for a complete listing of error messages

ORA-03113: TNS:end-of-file on communication channel

Cause: An error has occurred on the database server.

Action: Check the `alert_sid.log` file on the server. An unexpected end of file was processed on the communication channel. This may be an indication that the communications link may have gone down at least temporarily, or it may indicate that the server has gone down.You may need to modify your retransmission count.

ORA-12154: TNS:could not resolve the connect identifier specified

Cause: A connection to a database or other service was requested using a connect identifier, and the connect identifier specified could not be resolved into a connect descriptor using one of the naming methods configured. For example, if the type of connect identifier used was a net service name then the net service name could not be found in a naming method repository, or the repository could not be located or reached.

Action: Perform the following steps:

1. Check the type of naming adapters listed in the `names.directory_path` parameter in the `sqlnet.ora` file. If none are configured, then use the `adapters` command to determine which adapters are in use. The following example shows the adapters:

```
$ adapters
              ...
     Installed Oracle Net naming methods are:

         Local Naming (tnsnames.ora)
         Oracle Directory Naming
         Oracle Host Naming
         NIS Naming
```

   The net service name given in the connect string should be defined for at least one of the naming methods.

2. Check the resolution path for each adapter for possible problems. For example, ensure that the name given in the connect string is correct and complete, using the full name of the net service if necessary.

- When using the local naming method (TNSNAMES), do the following:

1. Verify that the `tnsnames.ora` file exists and is in the correct location. The location is either the ORACLE_HOME/network/admin directory or the directory specified by the TNS_ADMIN environment variable.

2. Verify there is an entry in the tnsnames.ora file for the name given in the connect string. This net service name should match the name in the `tnsnames.ora` file exactly if the name is simple and there is not NAMES_DEFAULT_DOMAIN in the `sqlnet.ora` file, or the net service name is a fully-qualified name. If the net service name in the connect string is simple, then check the NAMES_DEFAULT_DIRECTORY parameter in the `sqlnet.ora` file. Its value is appended to the net service name given in the connect string. This fully-qualified name should be the entry in the `tnsnames.ora` file.

3. If you are connecting from a login dialog box, then verify that you are not placing an "@" symbol before your connect net service name.
4. Activate client tracing and repeat the operation.

- When using the directory naming method (LDAP), do the following:

1. Verify the ldap.ora file exists and is in the correct location. The following directories are searched for ldap.ora file in the order given. The ldap.ora found will be used.
   - The directory specified by the TNS_ADMIN environment variable.
   - The `ORACLE_HOME/network/admin` directory.
   - The directory specified by the LDAP_ADMIN environment variable.
   - The `ORACLE_HOME/ldap/admin` directory.
2. Verify that the parameters defined in the ldap.ora file are correct, as follows:
   - The DIRECTORY_SERVERS parameter defines the correct host and port for one or more valid LDAP servers.
   - The DEFAULT_ADMIN_CONTEXT parameter defines the location of the Oracle Context in this directory which should include the net service entry.

   If the `ldap.ora` file does not exist, then these parameters will be resolved using automatic discovery.
3. Verify that the LDAP server host and port are defined in DNS.
4. Verify that the directory has the default Oracle Context defined.
5. Use the ldapsearch utility or a directory administration tool to verify that the net service object exists in the Oracle Context at the location given by the value of the DEFAULT_ADMIN_CONTEXT parameter.

- When using Oracle host naming method (EZCONNECT or HOSTNAME), do the following:

1. Verify that the host name give is correct, and is defined in the local host name resolution service, such as local hosts file, DNS, and so on.

- When using NIS naming method (NIS), do the following:

1. Verify that the NIS file for tnsnames is properly set up.
2. Check that the net service name matches the tnsnames entry as described in the preceding local naming section.

See Also:

- "Example of Troubleshooting a TNS-12154 Error" on page 16-19 for additional information about troubleshooting the error
- "Using Localized Management" on page 3-1 for configuration file location information
- Chapter 8, "Configuring Naming Methods" for naming information

ORA-12170: TNS:Connect timeout occurred

Cause: The client failed to establish a connection and complete authentication in the time specified by the `SQLNET.INBOUND_CONNECT_TIMEOUT` parameter in

the sqlnet.ora file. This error may be a result of network or system delays, or it may indicate that a malicious client is trying to cause a denial-of-service attack on the database server.

Action: If the error occurred due to system or network delays that are normal for the particular environment, then perform the following steps:

1. Turn on tracing to determine which clients are timing out.

   See Also: "Tracing Error Information for Oracle Net Services" on page 16-34

2. Reconfigure the SQLNET.INBOUND_CONNECT_TIMEOUT, SQLNET.SEND_TIMEOUT, or SQLNET.RECV_TIMEOUT parameters in sqlnet.ora to a larger value.

If you suspect a malicious client, then perform the following steps:

1. Restrict access to the client. For example, you can configure parameters for access rights in the sqlnet.ora file.
2. Locate the IP address of the client in the sqlnet.log file on the database server to identify the source. Remember that an IP address can be forged.

   For example, the following sqlnet.log excerpt shows a client IP address of 192.168.2.35.

```
Fatal NI connect error 12170.

  VERSION INFORMATION:
TNS for Linux: Version 11.2.0.0.0
Oracle Bequeath NT Protocol Adapter for Linux: Version 11.2.0.0.0
TCP/IP NT Protocol Adapter for Linux: Version 11.2.0.0.0
  Time: 03-MAY-2009 13:51:12
  Tracing to file: /ora/trace/svr_13279.trc
  Tns error struct:
    nr err code: 0
    ns main err code: 12637
    TNS-12637: Packet receive failed
    ns secondary err code: 12604
    nt main err code: 0
    nt secondary err code: 0
    nt OS err code: 0
  Client address: (ADDRESS=(PROTOCOL=tcp)(HOST=192.168.2.35)(PORT=52996))
```

   If the time out occurs before the IP address can be retrieved by the database server, then enable listener tracing to determine the client that made the request.

   See Also:

   - "Limiting Resource Consumption by Unauthorized Users" on page 14-7 additional information about setting the SQLNET.INBOUND_CONNECT_TIMEOUT parameter
   - "Configuring Database Access Control" on page 12-4
   - Tracing Error Information for Oracle Net Services on page 16-34

TNS-12500/ORA-12500: TNS: listener failed to start a dedicated server process

Cause: The listener failed to start the Oracle program. Possible reasons include:

- The maximum number of processes allowed for a single user was exceeded
- The listener does not have execute permission on the Oracle program
- The associated Microsoft Windows service is not started

In some cases, these errors can be caused by the same conditions which cause the following errors:

- TNS-12549/ORA-12549
- TNS-00519
- TNS-12540/ORA-12540
- TNS-00510
- TNS-12560/ORA-12560

Action: Perform the appropriate action:

- Increase the number of processes by setting the `PROCESSES` parameter in the database initialization file to a larger value.
- Check the `listener.log` file for detailed error stack information.

ORA-12514: TNS:listener does not currently know of service requested in connect descriptor

Cause: The listener received a request to establish a connection to a database or other service. The connect descriptor received by the listener specified a service name for a service (usually a database service) that has either not yet dynamically registered with the listener or has not been statically configured for the listener. This may be a temporary condition such as after the listener has started, but before the database instance has registered with the listener.

Action: Perform the following steps:

1. Wait a moment, and then try to connect a second time.
2. Check which services are currently known by the listener by running the Listener Control utility `STATUS` or `SERVICES` command.

   See Also: "Determining the Current Status of a Listener" on page 9-18 and "Monitoring Services of a Listener" on page 9-20

3. Check that the `SERVICE_NAME` parameter in the connect descriptor specifies a service name known by the listener.
4. Check for an event in the `listener.log` file.

   See Also: "Analyzing Listener Log Files" on page 16-27

ORA-12520: TNS:listener could not find available handler for requested type of server

Cause: The type of service handler requested by the client is incorrect or not registered for the requested `SERVICE_NAME`/`INSTANCE_NAME`, or the database instance is not registered with the listener.

Action: If you suspect the problem is the wrong type of service handler, then perform the following steps:

1. If `(server=`*`value`*`)` is set in the connect descriptor, then ensure that the value is set to the appropriate service handler type for the database, that is,

dedicated for dedicated server or shared for dispatchers. You can use the Listener Control utility SERVICES command to see what service handlers are currently registered with the listener.

2. If USE_DEDICATED_SERVER is set to ON in the sqlnet.ora file, then ensure the database is configured to use dedicated servers. If it is not, then set this parameter to off.
3. Ensure that the database instance is running. If the instance not running, then start it so that it can register with the listener.

See Also: "Monitoring Services of a Listener" on page 9-20 for additional information about service handlers

ORA-12521: TNS:listener does not currently know of instance requested in connect descripto

Cause: The instance name in the connect descriptor is incorrect, or the database instance is not registered with the listener.

Action: Perform the following steps:

1. Ensure the service name specified in the connect descriptor is correct.
2. Ensure the database instance is running. If the instance not running, then start it so that it can register with the listener. You can use the Listener Control utility SERVICES command to see what instances are currently registered with the listener.

See Also: "Monitoring Services of a Listener" on page 9-20 for additional information about SERVICES command

ORA-12525: TNS:listener has not received client's request in time allowed

Cause: The client failed to complete its connect request in the time specified by the INBOUND_CONNECT_TIMEOUT_*listener_name* parameter in the listener.ora file. This error may be a result of network or system delays, or it may indicate that a malicious client is trying to cause a denial-of-service attack on the listener.

See Also: "Limiting Resource Consumption by Unauthorized Users" on page 14-7 for additional information about setting the INBOUND_CONNECT_TIMEOUT_*listener_name* parameter

Action: If the error occurred due to system or network delays that are normal for the particular environment, then reconfigure the INBOUND_CONNECT_TIMEOUT_*listener_name* parameter in listener.ora to a larger value.

If you suspect a malicious client, then perform the following steps:

1. Locate the IP address of the client in listener.log to identify the source. Remember that an IP address can be forged.

   For example, the following listener.log excerpt shows a client IP address of 192.168.2.35.

```
03-MAY-2009 16:42:35 * <unknown connect data> *
(ADDRESS=(PROTOCOL=tcp)(HOST=192.168.2.35)(PORT=53208)) * establish *
<unknown sid> * 12525
TNS-12525: TNS:listener has not received client's request in time
allowed
```

```
TNS-12604: TNS: Application timeout occurred
```

2. Restrict access to the client. For example, you can configure parameters for access rights in the `sqlnet.ora` file.

   See Also: "Configuring Database Access Control" on page 12-4

ORA-12533: TNS:illegal ADDRESS parameters

Cause: The protocol specific parameters in the `ADDRESS` section of the designated connect descriptor are incorrect.

Action: Correct the protocol address.

Note: This error is often caused by hand-editing of the `tnsnames.ora` file.

See Also: *Oracle Database Net Services Reference* for correct protocol syntax

TNS-12540/ORA-12540: TNS:internal limit restriction exceeded and TNS-00510: Internal limit restriction exceeded

Cause: An internal limit has been exceeded. Possible limits include:

- Number of open connections that Oracle Net can process simultaneously
- Number of memory buffers which can be used simultaneously
- Number of processes a particular database instance is allowed

The first two are examples of hard limits. The third is an example of a limit which can be increased by setting `PROCESSES` parameter in the database initialization file to a larger value. In this case, a TNS-12500/ORA-12500 error is also returned.

In some cases, these errors can be caused by the same conditions which cause TNS-12549/ORA-12549 and TNS-00519 errors.

Action: Wait for the open connections to close and retry. If the error persists, then check the `sqlnet.log` or `listener.log` file for detailed error stack information.

TNS-12541/ORA-12541: TNS:no listener

Cause: The connection request could not be completed because the listener is not running.

Action: Perform the following actions:

- Ensure that the supplied destination address matches one of the addresses used by the listener.
- Verify that the listener is running at the address specified by the request.
- Ensure the listener is listening on the host and port specified by the request.
- Verify the client is pointing to the listener.

TNS-12549/ORA-12549: TNS:operating system resource quota exceeded and TNS-00519: Operating system resource quota exceeded

Cause: A quota or hard limit imposed by the operating system has been exceeded.

Possible limits include:

- The maximum number of processes allowed for a single user
- The operating system is running low on paging space

Action: Perform the appropriate action:

- Increase the number of processes by setting the `PROCESSES` parameter in the database initialization file to a larger value.
- Check the `sqlnet.log` or `listener.log` file for detailed error stack information, such as an operating system error code to help identify which quota has been exceeded.

TNS-12560/ORA-12560: TNS:protocol adapter error occurred

Cause: There was an error when using a particular protocol. This error may be due to incorrect configuration of an `ADDRESS` parameter or may occur due to errors returned from the underlying protocol or operating system interface.

In some cases, these errors are caused by the same conditions which cause TNS-00510, TNS-00519, TNS-12540/ORA-12540, TNS-12549/ORA-12549 errors.

Action: This error occurs on Microsoft Windows systems only. Perform the following actions:

1. Select **Run** from the Microsoft Windows Start menu.
2. Enter `MSCONFIG` in the Open field.
3. Go to the Services tab.
4. Enable `OracleService`*`sid`* if it is disabled.
5. Restart the computer.
6. Check that Oracle Services has started.

Troubleshooting Directory Naming Errors

Directory naming issues associated with connectivity errors for database service or net service name entries in a directory server require analysis of the data. You can analyze the data contained within a directory server with the `ldifwrite` command line tool. The `ldifwrite` tool is an Oracle Internet Directory tool.

The `ldifwrite` tool can be used to convert all or part of the information residing in a directory server to **LDAP Data Interchange Format (LDIF)**. The `ldifwrite` tool performs a subtree search, including all entries following the specified **distinguished name (DN)**, including the DN itself.

The `ldifwrite` tool syntax is as follows:

```
ldifwrite -c net_service_name/database_service -b base_DN -f ldif_file
```

Table 16–7 lists `ldifwrite` tool arguments and descriptions for each.

Table 16–7 Idifwrite Arguments

Argument	Description
-c net_service_name/database_service	The net service name or database service name that connects to the directory server.
-b base_DN	The base of the subtree to be written out in LDIF format.
-f ldif_file	The input file name.

The following example writes all the directory naming entries under `dc=us,dc=example,dc=com` to the `output1.ldi` file:

```
ldifwrite -c ldap -b "dc=us,dc=example,dc=com" -f output.ldif
```

Note: Check the `ldap.ora` file to determine the base_DN value. It is the same as the `DEFAULT_ADMIN_CONTEXT` entry in the `ldap.ora` file.

Troubleshooting Tips for Oracle Net Services

The following suggestions may be useful when diagnosing network problems:

- Use the node or network address during configuration instead of the name of the server computer.

  This eliminates any internal lookup problems and make the connection slightly faster.

- If you are using TCP/IP addresses, then use the IP address rather than the host name.

  For example, change the (`HOST=`*`server_name`*) line in the `tnsnames.ora` file with the IP address, for example (`HOST=192.168.2.5`).

- Perform a loopback test.

  Perform a loopback test on the server as described in Task 2, "Perform a Loopback Test" on page 16-7. If the test passes, then use FTP to send the `tnsnames.ora` and `sqlnet.ora` files to the client.

- Check the systems between the client and the server.

  If it is a wide area network (WAN), then identify any intermediate systems that may not work correctly. If all computers are fine, then the problem may be a timing issue.

- Verify whether there is a timing issue.

  Timing issues are associated with an `ORA-12535` error in the client log files.

  To resolve this, try speeding up the connection by using exact addresses instead of names and increase the `INBOUND_CONNECT_TIMEOUT_`*`listener_name`* parameter in the `listener.ora` file. The default value for this parameter is 10 seconds.

- Determine which Oracle applications are failing.

  SQL*Plus may work, but CASE tools may not. If you determine the problem is a data volume issue, then try to transfer a large (5 MB) file with the base connectivity.

Questions to Consider When Troubleshooting Oracle Net Services

The following questions can help diagnose a problem:

- Do all computers have a problem, or is it just one?

  If one computer works and another does not, and the same software (Oracle and third-party products) is installed on each computer, then, if possible, swap out the network cables to see if the problem occurs on the second client. If it does occur,

then it indicates that the problem has something to do with the client/server connection and is not local to the client.

- What kind of connections exist between the client and the server, for example, X.25, ISDN, or leased line?

  Sniffers and LAN analyzers are useful for locating intermittent connection failures, and detecting time outs and resent packets. You can also see which side is waiting for a response.

Example of Troubleshooting a TNS-12154 Error

This section offers some solutions to the TNS-12154 error. The TNS-12154 error is encountered when SQL*Net cannot find the alias specified for a connection in the `tnsnames.ora` file or other naming adapter.

Before attempting to resolve the problem, it may be helpful to have a printout or view the `tnsnames.ora` file and the `sqlnet.ora` file. Looking at these files at the same time is helpful because references are made to both.

The `tnsnames.ora` and `sqlnet.ora` files are located in the default network administration directory on the client system.

Be sure that the `tnsnames.ora` file and the `sqlnet.ora` file resemble the following examples.

Example 16–1 shows an example of a `tnsnames.ora` file.

Example 16–1 tnsnames.ora Sample

```
DEV1.WORLD =
     (DESCRIPTION =
           (ADDRESS =
             (PROTOCOL = TCP)
             (HOST = 192.168.2.56)
             (PORT = 1521)
           )
       (CONNECT_DATA =
         (SERVICE_NAME = sales.example.com)
       )
     )
```

Example 16–2 shows an example of a `sqlnet.ora` file.

Example 16–2 sqlnet.ora Sample

```
TRACE_LEVEL_CLIENT = OFF
SQLNET.AUTHENTICATION_SERVICES = (NONE)
NAMES.DIRECTORY_PATH = (TNSNAMES)
AUTOMATIC_IPC = OFF
```

To begin the diagnostic process, determine which section of this document applies to the problem. In the sample files shown in Example 16–1 and Example 16–2, the alias in Example 16–1 is `DEV1.WORLD`. However, the `NAMES.DEFAULT_DOMAIN=WORLD` parameter does not exist in Example 16–2. To fix this problem, add the `NAMES.DEFAULT_DOMAIN=WORLD` parameter anywhere in the `sqlnet.ora` file. Save the file, and try the connection again.

If the TNS-12154 error still persists, then determine whether the files were transferred from the client to the server and check the configuration files to ensure that CTRL-M (`^M`) or CTRL-R (`^R`) characters were not inserted at the ends of any lines. Remove any such characters you may find.

If the characters do not exist, then verify whether the NAMES.DIRECTORY_PATH parameter exists in the sqlnet.ora file and make sure the value in parentheses is TNSNAMES, as follows:

```
NAMES.DIRECTORY_PATH=(TNSNAMES)
NAMES.DIRECTORY_PATH=(TNSNAMES, EZCONNECT, HOSTNAME)
```

This parameter is not necessary but if it exists in the sqlnet.ora file and appears as shown in the preceding example, then the configuration files are technically accurate.

At the Linux prompt, echo the TNS_ADMIN environment variable, as follows:

```
% echo $TNS_ADMIN
```

If nothing is returned, then set the TNS_ADMIN environment variable to explicitly point to the location of the tnsnames.ora file.

- In C shell:

  ```
  % setenv TNS_ADMIN full_path_to_tnsnames.ora_file
  ```

- In Korn shell:

  ```
  % TNS_ADMIN=full_path_to_tnsnames.ora_file; export TNS_ADMIN
  ```

Try the connection again.

If the error persists, then add the AUTOMATIC_IPC=OFF parameter to the sqlnet.ora file. If AUTOMATIC_IPC is already set to ON, then change the value to OFF. Try the connection again.

If the error persists, then check the permissions of the tnsnames.ora and sqlnet.ora files and parent directories. Usually the .ora files are either -rwxrwxrwx or -rwxrwx---. Change the permissions of the configuration files to 777 to set the permissions to fully open and try the connection again.

If the error persists, then remove all line feeds and carriage returns so that the net alias is on one line, and try again.

> **Note:** Setting permissions to 777 enables anyone on the system to access the configuration files. Do this only as a temporary test and reset the permissions after the test.

If the error persists, then redo the configuration as follows:

1. Set the TNS_ADMIN environment variable to /tmp.
2. Go to the /tmp directory and create a new tnsnames.ora file using a text editor.
3. Copy the sample tnsnames.ora file from Example 16–1 into the text editor and save the new tnsnames.ora file.
4. Exit the text editor and at the command line, type:

   ```
   % sqlplus scott@dev1.world
   Enter password: password
   ```

Troubleshooting Network Problems Using Log and Trace Files

Oracle Net Services logs provide detailed information about the source and context of problems. The process of logging and tracing error information helps you to diagnose and resolve network problems.

Logging Error Information for Oracle Net Services

All errors encountered by Oracle Net Services are appended to a log file for evaluation by a network or database administrator. The log file provides additional information for an administrator about on-screen error messages. The error stack in the log file shows the state of the software at various layers.

To ensure that all errors are recorded, logging cannot be disabled on clients or name servers. Furthermore, only an administrator may replace or erase log files. The log file for the listener includes audit trail information about every client connection request, and most listener control commands.

This section contains the following topics:

Oracle Net Error Stacks

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 error stack components are described in Table 16–8.

Table 16–8 Error Stack Components

Error Stack Component	Description
NI	Network Interface. This layer provides a generic interface for Oracle clients, servers, or external processes to access Oracle Net functions. The NI layer handles the "break" and "reset" requests for a connection.
NS	Network Session (main and secondary layers). These layers receive requests from NI, and settle all generic computer-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). ■ How to handle interrupts between client and server based on the capabilities of each (send, receive functions).
NA	Network Authentication. This layer negotiates authentication and encryption requirements.
NT	Network Transport (main, secondary, and operating system layers). This layer maps Oracle Net foundation layer functionality to industry-standard protocols.

Understanding Error Stack Messages

Suppose that a user of a client application tries to establish a connection with a database server using Oracle Net and TCP/IP, by entering the following commands:

```
sqlplus scott@example.com
Enter password: password
```

When the commands are entered, the following error displays:

```
ORA-12543: TNS:Unable to connect to destination
```

This message indicates that the connection to the server failed because the database could not be contacted. Although the application displays only a one-line error message, an error stack that is much more informative is recorded in the log file by the network layer.

On the client side, the `sqlnet.log` file as shown in Example 16–3 contains an error stack corresponding to the `ORA-12543` error.

Example 16–3 sqlnet.log File

```
Fatal OSN connect error 12543, connecting to:
 (DESCRIPTION=(CONNECT_DATA=(SID=trace)(CID=(PROGRAM=)
   (HOST=lala)(USER=stiger)))(ADDRESS_LIST=(ADDRESS=
   (PROTOCOL=ipc)(KEY=trace))(ADDRESS=(PROTOCOL=tcp)
   (HOST=lala)(PORT=1521))))

VERSION INFORMATION:
TNS for Linux:
Oracle Bequeath NT Protocol Adapter for Linux:
Unix Domain Socket IPC NT Protocol Adaptor for Linux:
TCP/IP NT Protocol Adapter for Linux:
  Tracing to file: /home/db_tracefiles/trace_admin.trc
  Tns error struct:
    TNS-12543: TNS:unable to connect to destination
    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
```

Oracle Net Services Log File Names

Each Oracle Net Services component produces its own log file. When using ADR, the default, the log file names are log.xml in the appropriate alert directory. Table 16–9 lists the default log file names and lists the components that generate the log files that appear in the `ADR/diag/instance_name/trace` directory.

Table 16–9 Log Files

Component	Log File
Listener	listener.log
Client or Database Server	sqlnet.log
Oracle Connection Manager listener	instance-name_pid.log

Table 16–9 (Cont.) Log Files

Component	Log File
Oracle Connection Manager **CMGW (Oracle Connection Manager gateway)** process	instance-name_cmgw_pid.log
Oracle Connection Manager **CMADMIN (Oracle Connection Manager Administration)** process	instance-name_cmadmin_pid.log
Oracle Connection Manager alert log	instance-name_alert.log

Setting Logging Parameters

Parameters that control logging, including the type and amount of information logged, and the location where the files are stored, are set in the configuration file of each network component as described in Table 16–10.

Table 16–10 Location of Log Parameters

Network Component	Configuration File
Oracle Connection Manager Processes	cman.ora
Listener	listener.ora
Client	sqlnet.ora
Database Server	sqlnet.ora

Note: If `ADR_ENABLED` is set to `ON`, then all logging parameters are set by ADR. Using Oracle Net Manager to change the parameters will not work.

This section contains the following topics:

- sqlnet.ora Log Parameters
- listener.ora Log Parameters
- cman.ora Log Parameters
- Setting Logging Parameters in Configuration Files

See Also: *Oracle Database Net Services Reference* for additional information about the parameters

sqlnet.ora Log Parameters

Table 16–11 describes the log parameters settings that can be set in the `sqlnet.ora` file.

Table 16–11 sqlnet.ora Log Parameters

sqlnet.ora Parameter	Oracle Net Manager Field	Description
ADR_BASE	You must set this parameter manually.	The ADR_BASE parameter specifies the base directory for storing tracing and logging incidents. Use this parameter when DIAG_ADR_ENABLED is set to ON.
DIAG_ADR_ENABLED	You must set this parameter manually.	The DIAG_ADR_ENABLED parameter indicates whether ADR tracing is enabled. When the DIAG_ADR_ENABLED parameter is set to OFF, non-ADR file tracing is used.
LOG_DIRECTORY_CLIENT	Client Information: Log Directory	The destination directory for the client log file. By default, the client directory is the current working directory. This parameter is disabled when ADR_ENABLED is on.
LOG_DIRECTORY_SERVER	Server Information: Log Directory	The destination directory for the database server log files. By default the server directory is ORACLE_HOME/network/log. This parameter is disabled when ADR_ENABLED is on.
LOG_FILE_CLIENT	Client Information: Log File	The name of the log file for the client. By default the log name is sqlnet.log.
LOG_FILE_SERVER	You must set this parameter manually.	The name of the log file for the database server. By default the log name is sqlnet.log.

listener.ora Log Parameters

Table 16–12 describes the log parameters settings that can be set in the listener.ora file.

Table 16–12 listener.ora Log Parameters

listener.ora Parameter	Oracle Net Manager Field	Description
ADR_BASE_listener_name	You must set this parameter manually.	The ADR_BASE_*listener_name* parameter specifies the base directory for storing which tracing and logging incidents. Use when DIAG_ADR_ENABLED_*listener_name* is set to ON.
DIAG_ADR_ENABLED_listener_name	You must set this parameter manually.	The DIAG_ADR_ENABLED_*listener_name* parameter indicates whether ADR tracing is enabled. When DIAG_ADR_ENABLED_*listener_name* is set to OFF, non-ADR file tracing is used.
LOG_DIRECTORY_listener_name LOG_FILE_listener_name	Log File	The destination directory and file for the log file that is automatically generated for listener events. By default the directory is ORACLE_HOME/network/log, and the file name is defaulted to listener.log. These parameters are disabled when ADR_ENABLED is on.

cman.ora Log Parameters

Table 16–13 describes the log parameters settings that can be set in the cman.ora file.

Table 16–13 cman.ora Log Parameters

cman.ora Parameter	Description
ADR_BASE	The ADR_BASE parameter specifies the base directory for storing tracing and logging incidents. Use this parameter when DIAG_ADR_ENABLED is set to ON.
DIAG_ADR_ENABLED	The DIAG_ADR_ENABLED parameter indicates whether ADR tracing is enabled. When the DIAG_ADR_ENABLED parameter is set to OFF, non-ADR file tracing is used.
EVENT_GROUP	The event groups that are logged. Multiple events may be designated using a comma-delimited list. This parameter accepts the following values: ■ INIT_AND_TERM: initialization and termination ■ MEMORY_OPS: memory operations ■ CONN_HDLG: connection handling ■ PROC_MGMT: process management ■ REG_AND_LOAD: registration and load update ■ WAKE_UP: events related to CMADMIN wakeup queue ■ TIMER: gateway time outs ■ CMD_PROC: command processing ■ RELAY: events associated with connection control blocks
LOG_DIRECTORY	The destination directory for log files. By default, the directory is ORACLE_HOME/network/log. This parameter is disabled when ADR_ENABLED is on.
LOG_LEVEL	The level of logging. Four levels are supported: ■ off (default): no logging ■ user: user log information ■ admin: administrative log information ■ support: Oracle Support Services information

Setting Logging Parameters in Configuration Files

You configure logging parameters for the sqlnet.ora file with Oracle Net Manager and for the listener.ora file with either Oracle Enterprise Manager or Oracle Net Manager. You must manually configure cman.ora file logging parameters.

This section contains the following topics:

- Setting Parameters for the sqlnet.ora File
- Setting Parameters for the listener.ora File Using Oracle Enterprise Manager
- Setting Parameters for the listener.ora File Using Oracle Net Manager

Setting Parameters for the sqlnet.ora File The following procedure describes how to set the logging parameters in the sqlnet.ora file.

1. Start Oracle Net Manager.

> **See Also:** "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, expand **Profile** under the Local heading.
3. From the list in the right pane, select **General**.
4. Click the **Logging** tab.
5. Specify the settings.
6. Choose **Save Network Configuration** from the File menu.

The name of the log file is `sqlnet.log`.

Setting Parameters for the listener.ora File Using Oracle Enterprise Manager The following procedure describes how to set the logging parameters in the `listener.ora` file using Oracle Enterprise Manager:

1. Access the Oracle Net Administration page in Oracle Enterprise Manager.

   See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

2. Select **Listeners** from the **Administer** list, and then select the Oracle home that contains the location of the configuration files.
3. Click **Go** to display the Listeners page.
4. Select a listener, and then click **Edit** to display the Edit Listeners page.
5. Click the **Logging & Tracing** tab.
6. Specify the settings.
7. Click **OK**.

The name of the log file is `listener.log`.

Setting Parameters for the listener.ora File Using Oracle Net Manager The following procedure describes how to set the logging parameters in the `listener.ora` file using Oracle Net Manager:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, expand **Listeners** under the Local heading.
3. Select a listener.
4. From the list in the right pane, select **General**.
5. Click the **Logging and Tracing** tab.
6. Specify the settings.
7. Choose **Save Network Configuration** from the File menu.

The name of the log file is `listener.log`.

See Also: *Oracle Database Net Services Reference*

Setting Logging During Control Utilities Run Time

You can set logging during control utility run time. Setting logging with a control utility does not set parameters in the `*.ora` files, and the setting is only valid for the control utility session.

The following settings can be set for a control utility:

- For a listener, use the `SET LOG_FILE` and `SET LOG_DIRECTORY` commands from the Listener Control utility.
- For an Oracle Connection Manager, use the `SET LOG_DIRECTORY`, `SET LOG_LEVEL`, and `SET EVENT` commands from the Oracle Connection Manager control utility.

Note: If `ADR_ENABLED` is set to `ON`, then all logging parameters are set by ADR. Using Oracle Net Manager to change the parameters will not work.

See Also: *Oracle Database Net Services Reference*

Using Log Files

The following steps describe how to use a log file to diagnose a network error:

1. Review the log file for the most recent error number received from the application. This is usually the last entry in the log file.
2. Starting from the bottom of the file, locate the first nonzero entry in the error report. This is usually the actual cause.
3. If that error does not provide the information, then review the next error in the log until you locate the correct error information.
4. If the cause of the error is still not clear, then turn on tracing and repeat the command that produced the error message.

Analyzing Listener Log Files

This section describes what is recorded in the listener log file. This section contains the following topics:

- Listener Log Audit Trail Information
- Listener Service Registration Event Information
- Listener Direct Hand-Off Information
- Listener Subscription for ONS Node Down Event Information
- Listener Oracle Clusterware Notification Information

Listener Log Audit Trail Information

The listener log file contains audit trail information that enables you to collect and analyze network usage statistics, as well as information indicating the following:

- A client connection request
- A `RELOAD`, `START`, `STOP`, `STATUS`, or `SERVICES` command issued by the Listener Control utility

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

Format of the Listener Log Audit Trail

The audit trail formats text into the following fields:

```
Timestamp * Connect Data [* Protocol Info] * Event [* SID | Service] * Return Code
```

Properties of the audit trail are as follows:

- Each field is delimited by an asterisk (`*`).
- Protocol address information and service name or SID information appear only when a connection is attempted.
- A successful connection or command returns a code of zero.
- A failure produces a code that maps to an error message.

Example 16–4 shows a log file excerpt with `RELOAD` command request.

Example 16–4 Listener Log Event for Successful Reload Request

```
14-MAY-2009 00:29:54 *
(connect_data=(cid=(program=)(host=sales-server)(user=jdoe))(command=reload)
(arguments=64)(service=listener)(version=135290880))
* reload * 0
```

Example 16–5 shows a log file excerpt with a successful connection request.

Example 16–5 Listener Log Events for a Successful Connection Request

```
14-MAY-2009 15:28:58 *
(connect_data=(service_
name=sales.us.example.com)(cid=(program=)(host=sales-server)
(user=jdoe)))
* (address=(protocol=tcp)(host=192.168.2.35)(port=41349)) * establish
* sales.us.example.com * 0
```

Example 16–6 shows a log file excerpt with a successful execution of the `STATUS` command by host `sales-server`. It is followed by an unsuccessful connection attempt by a client with an IP address of `192.168.2.35`. This connection attempt resulted in an ORA-12525: TNS:listener has not received client's request in time allowed error message. This error occurs when a client fails to complete its connection request in the time specified by the `INBOUND_CONNECT_TIMEOUT_`*`listener_name`* parameter in the `listener.ora` file. This client could be attempting a denial-of-service attack on the listener.

Example 16–6 Listener Log Events for an Unsuccessful Connection Request

```
03-MAY-2009 16:41:57 *
(CONNECT_DATA=(CID=(PROGRAM=)(HOST=sales-server)(USER=jdoe))(COMMAND=status)
(ARGUMENTS=64)(SERVICE=LISTENER)(VERSION=153092352)) * status * 0
03-MAY-2009 16:42:35 * <unknown connect data> *
(ADDRESS=(PROTOCOL=tcp)(HOST=192.168.2.35)(PORT=53208)) * establish *
<unknown sid> * 12525
TNS-12525: TNS:listener has not received client's request in time allowed
TNS-12604: TNS: Application timeout occurred
```

See Also: *Oracle Database Error Messages* for a complete listing of error messages

Listener Service Registration Event Information

The listener records service registration events. During service registration, the **PMON process** provides the listener with information about the following:

- Service names for each running instance of the database
- Instance names of the database
- Service handlers (dispatchers or dedicated servers) available
- Dispatcher, instance, and node load information
- Dynamic listening endpoints

The recorded service registration-related events listed in Table 16–14 are listed in the `listener.log` file:

Table 16–14 Service Registration Event Log Information

Event	Description
service_register	The listener received registration information for an instance.
service_update	The listener received updated registration information for a particular instance, such as dispatcher or instance load information.
service_died	The listener lost its connection to PMON. All registration information for the instance is discarded. Clients will be unable to connect to the instance until PMON registers it again.

Format of the Listener Service Registration Information

The service registration events are formatted into the following fields:

```
Timestamp * Event *  Instance Name * Return Code
```

Properties of service registration fields are as follows:

- Each field is delimited by an asterisk (*).
- It is normal for the events to appear multiple times in a row for one instance.
- A successful registration returns a code of zero, meaning the client can connect to the instance.
- A failure produces a code that maps to an error message.

Example 16–7 shows a log file with service registration events. The listener is able to receive a client request after a successful `service_register` event, but is unable to receive client requests after a `service_died` event.

Example 16–7 Listener Log with Service Registration Events

```
------------------------------
14-MAY-2009 15:28:43 * service_register * sales * 0
14-MAY-2009 15:28:43 * service_register * sales * 0
14-MAY-2009 15:28:58 *
(connect_data=(service_name=sales.us.example.com)
(cid=(program=)(host=sales-server)(user=jdoe)))
* (address=(protocol=tcp)(host=192.168.2.35)(port=41349)) * establish
```

```
* sales.us.example.com * 0
14-MAY-2009 15:38:44 * service_update * sales * 0
14-MAY-2009 15:38:44 * service_update * sales * 0
14-MAY-2009 15:48:45 * service_update * sales * 0
14-MAY-2009 15:48:45 * service_update * sales * 0
14-MAY-2009 15:50:57 *
(connect_data=(service_name=sales.us.example.com)(cid=(program=)
(host=sales-server)(user=jdoe)))
* (address=(protocol=tcp)(host=192.168.2.35)(port=41365)) * establish
* sales.us.example.com * 0
14-MAY-2009 15:51:26 * service_died * sales * 12537
14-MAY-2009 15:51:26 * service_died * sales * 12537
14-MAY-2009 15:52:06 *
(connect_data=(service_name=sales.us.example.com)
(cid=(program=)(host=sales-server)(user=jdoe)))
* (address=(protocol=tcp)(host=192.168.2.35)(port=41406)) * establish
* sales.us.example.com * 12514
TNS-12514: TNS:listener could not resolve SERVICE_NAME given in connect descriptor
------------------------------
```

> **See Also:** *Oracle Database Error Messages* for a complete listing of error messages

Listener Direct Hand-Off Information

The listener records direct hand-off events to **dispatcher**s. These events are formatted into the following fields:

```
Timestamp * Presentation * Handoff  * Error Code
```

Properties of direct hand-off fields are as follows:

- Each field is delimited by an asterisk (*).
- A successful connection or command returns a code of zero.
- A failure produces a code that maps to an error message.

Example 16–8 shows a direct hand-off event in the log file.

Example 16–8 Listener Log Event for Direct Hand-Off

```
21-MAY-2009 10:54:55 * oracle.aurora.net.SALESHttp2 * handoff * 0
```

Listener Subscription for ONS Node Down Event Information

Listener subscribes to the Oracle Notification Service (ONS) node down event on startup if the ONS configuration file is available. This subscription enables the listener to remove the affected service when it receives node down event notification from ONS. The listener uses asynchronous subscription for the event notification.

The following warning message is recorded to the listener log file on each `STATUS` command if the subscription has not completed; for example if the ONS daemon is not running on the host.

```
WARNING: Subscription for node down event still pending
```

The listener cannot receive the ONS event while subscription is pending. Other than that, no other listener functionality is affected.

Listener Oracle Clusterware Notification Information

If the required Oracle Clusterware (CRS in the following log messages) libraries are installed and Oracle Clusterware is started on the host, then Oracle Listener will notify Oracle Clusterware about its status during start and stop processes. After successful notification, listeners record the event in the log. No message is recorded if the notification fails.

```
Listener completed notification to CRS on start
Listener completed notification to CRS on stop
```

Analyzing Oracle Connection Manager Logs

Oracle Connection Manager generates four types of log files: one each for its listener, gateway, CMADMIN processes and one for alerts. The last is a chronological record of all critical errors. In addition to logging critical errors, the alert log captures information about instance startup and shutdown. It also records the value of all configuration parameters at the beginning and end of a session.

The CMADMIN and gateway log files are reproduced here. Table 16–15, " CMADMIN and Gateway Log Entries" explains log entries. Each entry consists of a timestamp and an event. You can configure `cman.ora` to log events for the following categories:

- Initialization and termination
- Memory operations
- Connection handling
- Process management
- Registration and load update
- Events related to CMADMIN wakeup queue
- Gateway timeouts
- Command processing
- Events associated with connection control blocks

Use the `SET EVENT` command to specify which events to log.

Example 16–9 shows a typical CMADMIN log.

Example 16–9 CMADMIN Log File

```
-------------------------------
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:40)(EVENT=Parameter list)
    (listener_address=(address=(protocol=tcp)(host=sales1)(port=1574)))
    (aso_authentication_filter=OFF)
    (connection_statistics=ON)
    (log_directory=/home/user/network/admin/log)
    (log_level=support)
    (max_connections=256)
    (idle_timeout=5)
    (inbound_connect_timeout=0)
    (session_timeout=20)
    (outbound_connect_timeout=0)
    (max_gateway_processes=1)
    (min_gateway_processes=1)
    (password=OFF)
    (trace_directory=/home/user/network/admin/log)
    (trace_level=off)
    (trace_timestamp=OFF)
```

```
    (trace_filelen=0)
    (trace_fileno=0)
)
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:40)(EVENT=Shared Memory Size)
(BYTES=82524))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:40)(EVENT=GMON Attributes validated)
(Type=Information))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:40)(EVENT=NS Listen Successful)
((ADDRESS=(PROTOCOL=tcp)(HOST=usunnae16)(PORT=55878))))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:44)(EVENT=Received command)(CMD=verify
password))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:44)(EVENT=Received command)
(CMD=version))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:44)(EVENT=Received command)
(CMD=show status))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:44)(EVENT=Failed to get procedure id))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:12)(EVENT=Received command)(CMD=verify
password))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:15)(EVENT=Failed to get procedure id))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:29)(EVENT=Received command)(CMD=verify
password))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:46)(EVENT=Failed to get procedure id))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:50)(EVENT=Received command)(CMD=verify
password))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:50)(EVENT=Received command)
(CMD=probe monitor))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:50)(EVENT=Received command)
(CMD=shutdown normal))
-------------------------------
```

Example 16–10 shows a typical gateway log file.

Example 16–10 Gateway Log File

```
-------------------------------
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=NS Initialised))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=Memory Allocated)
(BYTES=1024))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=NCR Initialised))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=Connected to Monitor))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=State Change from Empty to
Init))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=Memory Allocated)
(BYTES=251904))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=Memory Allocated)
(BYTES=2048))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=CCB Initialised))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=Started Listening))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:41)(EVENT=State Change from Init to
Ready))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:46:47)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:06)(EVENT=Ready)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:06)(EVENT=Ready)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:07)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:12)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:13)(EVENT=Idle Timeout)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:17)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:22)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:25)(EVENT=Ready)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:25)(EVENT=Ready)(CONN NO=0))
```

```
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:27)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:30)(EVENT=Idle Timeout)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:32)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:37)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:42)(EVENT=Ready)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:42)(EVENT=Ready)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:42)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:47)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:52)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:48:57)(EVENT=Housekeeping))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:02)(EVENT=Session Timeout)(CONN NO=0))
(LOG_RECORD=(TIMESTAMP=08-MAY-2009 08:49:02)(EVENT=Housekeeping))
-------------------------------
```

Table 16–15 shows the log file entries and their descriptions.

Table 16–15 CMADMIN and Gateway Log Entries

Log File	Event	Description
CMADMIN	Failed to get procedure ID	The CMCTL session connected to CMADMIN has disconnected.
CMADMIN	GMON Attributes validated	Informational message. The parameters needed for CMADMIN to come up are specified correctly.
CMADMIN	Invalid connect data	An unknown client is trying to connect to CMADMIN. This is most likely a denial of service attack.
CMADMIN	No connect data	An unknown client is trying to connect to CMADMIN. This is most likely a denial of service attack.
Gateway	Connected to Monitor	The gateway has connected to CMADMIN.
Gateway	Housekeeping	Informational message. Internal housekeeping for the gateway process is in order. The gateway process is properly connected to the CMADMIN process.
Gateway	Idle Timeout	The connection was disconnected because it was idle longer than the time specified in `cman.ora`.
Gateway	Out of connection control block (CCB)	CMADMIN cannot process a connection request. There could be two reasons: ■ Faulty load update between CMADMIN and listener. ■ Someone is trying to connect to CMADMIN directly (possibly a denial of service attack).
Gateway	Session Timeout	The connection was disconnected because it exceeded the session timeout specified in `cman.ora`.
Gateway	State change from Empty to Init	State change message from the gateway. After it reaches a ready state, the gateway begins accepting connections from the client.
Gateway	State change from Init to Ready	State change message from the gateway. After it reaches a ready state, the gateway begins accepting connections from the client.

Tracing Error Information for Oracle Net Services

Tracing produces a detailed sequence of statements that describe network events as they are run. Tracing an operation enables you to obtain more information about the internal operations of the components of Oracle Net Services 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.

> **Note:** Tracing uses a large amount of disk space and may have a significant impact upon system performance. Therefore, you should enable tracing only when necessary.

This section contains the following topics:

Oracle Net Services Trace File Names

Each Oracle Net Services component produces its own trace file. Table 16–16 provides the default trace file names and lists the components that generate the trace files.

Table 16–16 Trace Files Names

Trace File	Component
instance-name_pid.trc	Oracle Connection Manager listener
instance-name_cmgw_pid.trc	Oracle Connection Manager **CMGW (Oracle Connection Manager gateway)** process
instance-name_cmadmin_pid.trc	Oracle Connection Manager **CMADMIN (Oracle Connection Manager Administration)** process
listener.trc	Listener
sqlnet.trc	Client
svr_pid.trc	Database server
tnsping.trc	TNSPING utility

Setting Tracing Parameters

Parameters that control tracing, including the type and amount of information trace, and the location where the files are stored, are set in the configuration file of each network component as described in Table 16–17.

Table 16–17 Location of Trace Parameters

Configuration File	Component
cman.ora	Oracle Connection Manager Processes
listener.ora	Listener

Table 16–17 (Cont.) Location of Trace Parameters

Configuration File	Component
sqlnet.ora	Client Database server TNSPING utility

This section contains the following topics:

- cman.ora Trace Parameters
- listener.ora Trace Parameters
- sqlnet.ora Trace Parameters
- Setting Tracing Parameters in Configuration Files

> **See Also:** *Oracle Database Net Services Reference* for additional information about these parameters

cman.ora Trace Parameters

Table 16–18 describes the trace parameters settings for Oracle Connection Manager that can be set in the `cman.ora` file.

Table 16–18 cman.ora Trace Parameters

cman.ora Parameter	Description
TRACE_DIRECTORY	The destination directory for trace files. By default, the directory is `ORACLE_HOME/network/trace`.
TRACE_FILELEN	The size of the trace file in KB. When the size is reached, the trace information is written to the next file. The number of files is specified with the `TRACE_FILENO` parameter.
TRACE_FILENO	The number of trace files for tracing. When this parameter is set along with the `TRACE_FILELEN` parameter, trace files are used in a cyclical fashion. The first file is filled, then the second file, and so on. When the last file has been filled, the first file is reused, and so on. The trace file names are distinguished from one another by their sequence number. For example, if this parameter is set to 3, then the Oracle Connection Manager trace files for the gateway processes would be named *instance-name*`_cmgw1_`*pid*`.trc`, *instance-name*`_cmgw2_`*pid*`.trc` and *instance-name*`_cmgw3_`*pid*`.trc`. In addition, trace events in the trace files are preceded by the sequence number of the file.

Table 16–18 (Cont.) cman.ora Trace Parameters

cman.ora Parameter	Description
TRACE_LEVEL	The level of detail the trace facility records for the listener. The trace level value can either be a value within the range of 0 (zero) to 16 where 0 is no tracing and 16 represents the maximum amount of tracing) or one of the following values: ■ `off` (equivalent to 0) provides no tracing. ■ `user` (equivalent to 4) traces to identify user-induced error conditions. ■ `admin` (equivalent to 6) traces to identify installation-specific problems. ■ `support` (equivalent to 16) provides trace information for troubleshooting information for Oracle Support Services. The Oracle Connection Manager listener, gateway, and CMADMIN processes create trace files on both Linux and Microsoft Windows.
TRACE_TIMESTAMP	If the `TRACING` parameter is enabled, then a time stamp in the form of *`dd-mon-yyyy hh:mi:ss:mil`* is created for every trace event in the listener trace file.

listener.ora Trace Parameters

Table 16–19 describes the trace parameters settings for the listener that can be set in the `listener.ora` file.

Table 16–19 listener.ora Trace Parameters

listener.ora Parameter	Oracle Enterprise Manager/Oracle Net Manager Field	Description
TRACE_LEVEL_ listener_name	Select a trace level/Trace Level	The level of detail the trace facility records for the listener. The trace level value can either be a value within the range of 0 (zero) to 16 where 0 is no tracing and 16 represents the maximum amount of tracing) or one of the following values: ■ `off` (equivalent to 0) provides no tracing. ■ `user` (equivalent to 4) traces to identify user-induced error conditions. ■ `admin` (equivalent to 6) traces to identify installation-specific problems. ■ `support` (equivalent to 16) provides trace information for troubleshooting information for Oracle Support Services.
TRACE_DIRECTORY_ listener_name TRACE_FILE_listener_ name	Trace File	The destination directory and file for the trace file. By default the directory is `ORACLE_ HOME/network/trace`, and the file name is `listener.trc`.
TRACE_FILELEN_ listener_name	You must set this parameter manually.	The size of the listener trace files in KB. When the size is reached, the trace information is written to the next file. The number of files is specified with the `TRACE_FILENO_`*`listener_ name`* parameter

Table 16–19 (Cont.) listener.ora Trace Parameters

listener.ora Parameter	Oracle Enterprise Manager/Oracle Net Manager Field	Description
TRACE_FILENO_ listener_name	You must set this parameter manually.	The number of trace files for listener tracing. When this parameter is set along with the TRACE_FILELEN_*listener_name* parameter, trace files are used in a cyclical fashion. The first file is filled, then the second file, and so on. When the last file has been filled, the first file is re-used, and so on. The trace file names are distinguished from one another by their sequence number. For example, if the default trace file of `listener.trc` is used, and this parameter is set to 3, then the trace files would be named `listener1.trc`, `listener2.trc` and `listener3.trc`. In addition, trace events in the trace files are preceded by the sequence number of the file.
TRACE_TIMESTAMP_ listener_name	You must set this parameter manually.	A time stamp in the form of *dd-mon-yyyy hh:mi:ss:mil* for every trace event in the listener trace file.

sqlnet.ora Trace Parameters

Table 16–20 describes the trace parameters settings that can be set in the `sqlnet.ora` file.

Table 16–20 sqlnet.ora Trace Parameters

sqlnet.ora Parameter	Oracle Net Manager Field	Description
TRACE_DIRECTORY_ CLIENT	Client Information: Trace Directory	The destination directory for the client trace output. By default, the client directory is `ORACLE_HOME/network/trace`.
TRACE_DIRECTORY_ SERVER	Server Information: Trace Directory	The destination directory for the database server trace output. By default, the server directory is `ORACLE_HOME/network/trace`.
TRACE_FILE_CLIENT	Client Information: Trace File	The name of the trace file for the client. By default, the trace file name is `sqlnet.trc`.
TRACE_FILE_SERVER	Server Information: Trace File	The name of the trace file for the database server. By default the trace file name is `svr_pid.trc`.
TRACE_FILELEN_CLIENT	You must set this parameter manually.	The size of the client trace files in KB. When the size is reached, the trace information is written to the next file. The number of files is specified with the `TRACE_FILENO_CLIENT` parameter.
TRACE_FILELEN_SERVER	You must set this parameter manually.	The size of the database server trace files in KB. When the size is reached, the trace information is written to the next file. The number of files is specified with the `TRACE_FILENO_SERVER` parameter.

Table 16–20 (Cont.) sqlnet.ora Trace Parameters

sqlnet.ora Parameter	Oracle Net Manager Field	Description
TRACE_FILENO_CLIENT	You must set this parameter manually.	The number of trace files for client tracing. When this parameter is set along with the `TRACE_FILELEN_CLIENT` parameter, trace files are used in a cyclical fashion. The first file is filled, then the second file, and so on. When the last file has been filled, the first file is re-used, and so on. The trace file names are distinguished from one another by their sequence number. For example, if the default trace file of `sqlnet.trc` is used, and this parameter is set to 3, then the trace files would be named `sqlnet1_`*`pid`*`.trc`, `sqlnet2_`*`pid`*`.trc` and `sqlnet3_`*`pid`*`.trc`. In addition, trace events in the trace files are preceded by the sequence number of the file.
TRACE_FILENO_SERVER	You must set this parameter manually.	The number of trace files for database server tracing. When this parameter is set along with the `TRACE_FILELEN_SERVER` parameter, trace files are used in a cyclical fashion. The first file is filled, then the second file, and so on. When the last file has been filled, the first file is re-used, and so on. The trace file names are distinguished from one another by their sequence number. For example, if the default trace file of `svr_`*`pid`*`.trc` is used, and this parameter is set to 3, then the trace files would be named `svr1_`*`pid`*`.trc`, `svr2_`*`pid`*`.trc` and `svr3_`*`pid`*`.trc`. In addition, trace events in the trace files are preceded by the sequence number of the file.
TRACE_LEVEL_CLIENT	Client Information: Trace Level	The level of detail the trace facility records for the client. The trace level value can either be a value within the range of 0 (zero) to 16 where 0 is no tracing and 16 represents the maximum amount of tracing) or one of the following values: ■ `off` (equivalent to 0) provides no tracing. ■ `user` (equivalent to 4) traces to identify user-induced error conditions. ■ `admin` (equivalent to 6) traces to identify installation-specific problems. ■ `support` (equivalent to 16) provides trace information for troubleshooting information for Oracle Support Services.
TRACE_LEVEL_SERVER	Server Information: Trace Level	The level of detail the trace facility records for the database server. The trace level value can either be a value within the range of 0 (zero) to 16 where 0 is no tracing and 16 represents the maximum amount of tracing) or one of the following values: ■ `off` (equivalent to 0) provides no tracing. ■ `user` (equivalent to 4) traces to identify user-induced error conditions. ■ `admin` (equivalent to 6) traces to identify installation-specific problems. ■ `support` (equivalent to 16) provides trace information for troubleshooting information for Oracle Support Services.

Table 16–20 (Cont.) sqlnet.ora Trace Parameters

sqlnet.ora Parameter	Oracle Net Manager Field	Description
TRACE_TIMESTAMP_CLIENT	You must set this parameter manually.	A time stamp in the form of *dd-mon-yyyy hh:mi:ss:mil* for every trace event in the client trace file, `sqlnet.trc`.
TRACE_TIMESTAMP_SERVER	You must set this parameter manually.	A time stamp in the form of *dd-mon-yyyy hh:mi:ss:mil* for every trace event in the client trace file, `sqlnet.trc`.
TRACE_UNIQUE_CLIENT	Client Information: Unique Trace File Name	The value is set to `on`, Oracle Net creates a unique file name for each trace session by appending a process identifier to the name of each trace file generated, and enabling several files to coexist. For example, trace files named `sqlnet`*pid*`.trc` are created if default trace file name `sqlnet.trc` is used. When the value is set to `off`, data from a new client trace session overwrites the existing file.

You can manually add the TNSPING utility tracing parameters described in Table 16–21 to the `sqlnet.ora` file. The TNSPING utility determines whether a service, such as a database or other TNS services, on a Oracle Net network can be successfully reached.

Table 16–21 TNSPING Trace Parameters

sqlnet.ora Parameter	Description
TNSPING.TRACE_DIRECTORY	The destination directory for TNSPING trace file, `tnsping.trc`. By default, the directory is `ORACLE_HOME/network/trace`.
TNSPING.TRACE_LEVEL	The level of detail the trace facility records for the TNSPING utility. The trace level value can either be a value within the range of 0 (zero) to 16 where 0 is no tracing and 16 represents the maximum amount of tracing) or one of the following values: ■ `off` (equivalent to 0) provides no tracing. ■ `user` (equivalent to 4) traces to identify user-induced error conditions. ■ `admin` (equivalent to 6) traces to identify installation-specific problems. ■ `support` (equivalent to 16) provides trace information for troubleshooting information for Oracle Support Services.

Setting Tracing Parameters in Configuration Files

Configure tracing parameters for the `sqlnet.ora` file with Oracle Net Manager and `listener.ora` file with either Oracle Enterprise Manager or Oracle Net Manager. You must manually configure `cman.ora` file tracing parameters.

This section contains the following topics:

- Setting Tracing Parameters for sqlnet.ora File Using Oracle Net Manager
- Setting Tracing Parameters for the Listener Using Oracle Enterprise Manager
- Setting Tracing Parameters for the Listener Using Oracle Net Manager

Setting Tracing Parameters for sqlnet.ora File Using Oracle Net Manager The following procedure describes how to set the tracing parameters for the `sqlnet.ora` file using Oracle Net Manager:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, expand **Profile** under the Local heading.
3. From the list in the right pane, select **General**.
4. Click the **Tracing** tab.
5. Specify the settings.
6. Choose **Save Network Configuration** from the File menu.

The name of the trace file for the client is `sqlnet.trc`. The name of the trace file for the server is `svr_pid.trc`.

Setting Tracing Parameters for the Listener Using Oracle Enterprise Manager The following procedure describes how to set the tracing parameters for the listener using Oracle Enterprise Manager:

1. Access the Oracle Net Administration page in Oracle Enterprise Manager.

   See Also: "Using Oracle Enterprise Manager to Configure Oracle Net Services" on page 7-1

2. Select **Listeners** from the **Administer** list, and then select the Oracle home that contains the location of the configuration files.
3. Click **Go** to display the Listeners page.
4. Select a listener, and then click **Edit** to display the Edit Listeners page.
5. Click the **Logging & Tracing** tab.
6. Specify the settings.
7. Click **OK**.

The name of the trace file is `listener.trc`.

Setting Tracing Parameters for the Listener Using Oracle Net Manager The following procedure describes how to set the tracing parameters for the listener using Oracle Net Manager:

1. Start Oracle Net Manager.

   See Also: "Using Oracle Net Manager to Configure Oracle Net Services" on page 7-2

2. In the navigator pane, expand **Listeners** from the Local heading.
3. Select a listener.
4. From the list in the right pane, select **General**.
5. Click the **Logging and Tracing** tab.
6. Specify the settings.

7. Choose **Save Network Configuration** from the File menu.

Setting Tracing During Control Utilities Run Time

You can set tracing during control utility run time. Setting tracing with a control utility does not set parameters in the `*.ora` files. The setting is only valid for the session of the control utility:

- For the listener, use the `SET TRC_DIRECTORY`, `SET TRC_FILE`, and `SET TRC_LEVEL` commands from the Listener Control utility.
- For an Oracle Connection Manager, use the `SET TRACE_DIRECTORY` and `SET TRACE_LEVEL`, and `SET TRACE_TIMESTAMP` commands from the Oracle Connection Manager control utility.

Evaluating Oracle Net Services Trace Files

Trace files can help Oracle Support Services diagnose and troubleshoot network problems. This section explains how to perform basic analysis of trace files. It contains the following topics:

- Flow of Data Packets Between Network Nodes
- Oracle Net Data Packet Formats
- Pertinent Oracle Net Trace Error Output

Flow of Data Packets Between Network Nodes

Oracle Net performs its functions by sending and receiving data packets. You can view the actual contents of the Oracle Net packet in your trace file by specifying a trace level of `support`. The order of the packet types sent and received will help you to determine how the connection was established.

Oracle Net Data Packet Formats

Each line in the trace file begins with a procedure followed by a message. Following each procedure is a line of hexadecimal data representing actual data. The actual data that flows inside the packet is sometimes viewable to the right of the hexadecimal data.

Each packet has a keyword that denotes the packet type. All packet types begin with the prefix "`nsp`". This is helpful when reviewing trace files for specific packet information. The following keywords are used in a trace file:

- NSPTCN: Used with Connect packet types.
- NSPTAC: Used with Accept packet types.
- NSPTRF: Used with Refuse packet types.
- NSPTRS: Used with Resend packet types.
- NSPTDA: Used with Data packet types.
- NSPCNL: Used with Control packet types.
- NSPTMK: Used with Marker packet types.

Example 16–11 provides typical packet information. In the example, the `nscon` procedure sends an NSPTCN packet over the network.

Example 16–11 Packet Information

```
nscon: entry
nscon: doing connect handshake...
nscon: sending NSPTCN packet
nspsend: entry
nspsend: plen=187, type=1
nspsend: 187 bytes to transport
nspsend:packet dump
nspsend:00 BB 00 00 01 00 00 00  |........|
nspsend:01 33 01 2C 0C 01 08 00  |.3.,....|
nspsend:7F FF 7F 08 00 00 00 01  |........|
nspsend:00 99 00 22 00 00 08 00  |..."....|
nspsend:01 01 28 44 45 53 43 52  |..(DESCR|
nspsend:49 50 54 49 4F 4E 3D 28  |IPTION=(|
nspsend:43 4F 4E 4E 45 43 54 5F  |CONNECT_|
nspsend:44 41 54 41 3D 28 53 49  |DATA=(SI|
nspsend:44 3D 61 70 33 34 37 64  |D=ap347d|
nspsend:62 31 29 28 43 49 44 3D  |b1)(CID=|
nspsend:28 50 52 4F 47 52 41 4D  |(PROGRAM|
nspsend:3D 29 28 48 4F 53 54 3D  |=)(HOST=|
nspsend:61 70 32 30 37 73 75 6E  |sales-12|
nspsend:29 28 55 53 45 52 3D 6D  |)(USER=m|
nspsend:77 61 72 72 65 6E 29 29  |stiger))|
nspsend:29 28 41 44 44 52 45 53  |)(ADDRES|
nspsend:53 5F 4C 49 53 54 3D 28  |S_LIST=(|
nspsend:41 44 44 52 45 53 53 3D  |ADDRESS=|
nspsend:28 50 52 4F 54 4F 43 4F  |(PROTOCO|
nspsend:4C 3D 74 63 70 29 28 48  |L=tcp)(H|
nspsend:4F 53 54 3D 61 70 33 34  |OST=sale|
nspsend:37 73 75 6E 29 28 50 4F  |s-12)(PO|
nspsend:52 54 3D 31 35 32 31 29  |RT=1521)|
nspsend:29 29 29 00 00 00 00 00  |)))).....|
nspsend: normal exit
nscon: exit (0)
```

Pertinent Oracle Net Trace Error Output

When there is a problem, the error code is logged in the trace file. Example 16–12 illustrates typical trace file output for a failed SQL*Plus connection to a database server. The error message and error stack are shown in bold.

Example 16–12 Trace Example

```
[22-MAY-2009 13:34:07:687] nsprecv: entry
[22-MAY-2009 13:34:07:687] nsbal: entry
[22-MAY-2009 13:34:07:687] nsbgetfl: entry
[22-MAY-2009 13:34:07:687] nsbgetfl: normal exit
[22-MAY-2009 13:34:07:687] nsmal: entry
[22-MAY-2009 13:34:07:687] nsmal: 44 bytes at 0x132d90
[22-MAY-2009 13:34:07:687] nsmal: normal exit
[22-MAY-2009 13:34:07:687] nsbal: normal exit
[22-MAY-2009 13:34:07:687] nsprecv: reading from transport...
[22-MAY-2009 13:34:07:687] nttrd: entry
[22-MAY-2009 13:35:09:625] nttrd: exit
[22-MAY-2009 13:35:09:625] ntt2err: entry
[22-MAY-2009 13:35:09:625] ntt2err: Read unexpected EOF ERROR on 10
[22-MAY-2009 13:35:09:625] ntt2err: exit
[22-MAY-2009 13:35:09:625] nsprecv: transport read error
[22-MAY-2009 13:35:09:625] nsprecv: error exit
[22-MAY-2009 13:35:09:625] nserror: entry
```

```
[22-MAY-2009 13:35:09:625] nserror: nsres: id=0, op=68, ns=12537, ns2=12560;
nt[0]=507, nt[1]=0, nt[2]=0; ora[0]=0, ora[1]=0, ora[2]=0
[22-MAY-2009 13:35:09:625] nscon: error exit
[22-MAY-2009 13:35:09:625] nsdo: nsctxrnk=0
[22-MAY-2009 13:35:09:625] nsdo: error exit
[22-MAY-2009 13:35:09:625] nscall: unexpected response
[22-MAY-2009 13:35:09:625] nsclose: entry
[22-MAY-2009 13:35:09:625] nstimarmed: entry
[22-MAY-2009 13:35:09:625] nstimarmed: no timer allocated
[22-MAY-2009 13:35:09:625] nstimarmed: normal exit
[22-MAY-2009 13:35:09:625] nsdo: entry
[22-MAY-2009 13:35:09:625] nsdo: cid=0, opcode=98, *bl=0, *what=0,
uflgs=0x440, cflgs=0x2
[22-MAY-2009 13:35:09:625] nsdo: rank=64, nsctxrnk=0
[22-MAY-2009 13:35:09:625] nsdo: nsctx: state=1, flg=0x4201, mvd=0
[22-MAY-2009 13:35:09:625] nsbfr: entry
[22-MAY-2009 13:35:09:625] nsbaddfl: entry
[22-MAY-2009 13:35:09:625] nsbaddfl: normal exit
[22-MAY-2009 13:35:09:625] nsbfr: normal exit
[22-MAY-2009 13:35:09:625] nsbfr: entry
[22-MAY-2009 13:35:09:625] nsbaddfl: entry
[22-MAY-2009 13:35:09:625] nsbaddfl: normal exit
[22-MAY-2009 13:35:09:625] nsbfr: normal exit
[22-MAY-2009 13:35:09:625] nsdo: nsctxrnk=0
[22-MAY-2009 13:35:09:625] nsdo: normal exit
[22-MAY-2009 13:35:09:625] nsclose: closing transport
[22-MAY-2009 13:35:09:625] nttdisc: entry
[22-MAY-2009 13:35:09:625] nttdisc: Closed socket 10
[22-MAY-2009 13:35:09:625] nttdisc: exit
[22-MAY-2009 13:35:09:625] nsclose: global context check-out (from slot 0)
complete
[22-MAY-2009 13:35:09:703] nsnadisc: entry
[22-MAY-2009 13:35:09:703] nadisc: entry
[22-MAY-2009 13:35:09:703] nacomtm: entry
[22-MAY-2009 13:35:09:703] nacompd: entry
[22-MAY-2009 13:35:09:703] nacompd: exit
[22-MAY-2009 13:35:09:703] nacompd: entry
[22-MAY-2009 13:35:09:703] nacompd: exit
[22-MAY-2009 13:35:09:703] nacomtm: exit
[22-MAY-2009 13:35:09:703] nas_dis: entry
[22-MAY-2009 13:35:09:703] nas_dis: exit
[22-MAY-2009 13:35:09:703] nau_dis: entry
[22-MAY-2009 13:35:09:703] nau_dis: exit
[22-MAY-2009 13:35:09:703] naeetrm: entry
[22-MAY-2009 13:35:09:703] naeetrm: exit
[22-MAY-2009 13:35:09:703] naectrm: entry
[22-MAY-2009 13:35:09:703] naectrm: exit
[22-MAY-2009 13:35:09:703] nagbltrm: entry
[22-MAY-2009 13:35:09:703] nau_gtm: entry
[22-MAY-2009 13:35:09:703] nau_gtm: exit
[22-MAY-2009 13:35:09:703] nagbltrm: exit
[22-MAY-2009 13:35:09:703] nadisc: exit
[22-MAY-2009 13:35:09:703] nsnadisc: normal exit
[22-MAY-2009 13:35:09:703] nsbfr: entry
[22-MAY-2009 13:35:09:703] nsbaddfl: entry
[22-MAY-2009 13:35:09:703] nsbaddfl: normal exit
[22-MAY-2009 13:35:09:703] nsbfr: normal exit
[22-MAY-2009 13:35:09:703] nsmfr: entry
[22-MAY-2009 13:35:09:703] nsmfr: 2256 bytes at 0x130508
[22-MAY-2009 13:35:09:703] nsmfr: normal exit
```

```
[22-MAY-2009 13:35:09:703] nsmfr: entry
[22-MAY-2009 13:35:09:703] nsmfr: 484 bytes at 0x1398a8
[22-MAY-2009 13:35:09:703] nsmfr: normal exit
[22-MAY-2009 13:35:09:703] nsclose: normal exit
[22-MAY-2009 13:35:09:703] nscall: connecting...
[22-MAY-2009 13:35:09:703] nsclose: entry
[22-MAY-2009 13:35:09:703] nsclose: normal exit
[22-MAY-2009 13:35:09:703] nladget: entry
[22-MAY-2009 13:35:09:734] nladget: exit
[22-MAY-2009 13:35:09:734] nsmfr: entry
[22-MAY-2009 13:35:09:734] nsmfr: 144 bytes at 0x132cf8
[22-MAY-2009 13:35:09:734] nsmfr: normal exit
[22-MAY-2009 13:35:09:734] nsmfr: entry
[22-MAY-2009 13:35:09:734] nsmfr: 156 bytes at 0x138e70
[22-MAY-2009 13:35:09:734] nsmfr: normal exit
[22-MAY-2009 13:35:09:734] nladtrm: entry
[22-MAY-2009 13:35:09:734] nladtrm: exit
[22-MAY-2009 13:35:09:734] nscall: error exit
[22-MAY-2009 13:35:09:734] nioqper:  error from nscall
[22-MAY-2009 13:35:09:734] nioqper:    ns main err code: 12537
[22-MAY-2009 13:35:09:734] nioqper:    ns (2)  err code: 12560
[22-MAY-2009 13:35:09:734] nioqper:    nt main err code: 507
[22-MAY-2009 13:35:09:734] nioqper:    nt (2)  err code: 0
[22-MAY-2009 13:35:09:734] nioqper:    nt OS   err code: 0
[22-MAY-2009 13:35:09:734] niomapnserror: entry
[22-MAY-2009 13:35:09:734] niqme: entry
[22-MAY-2009 13:35:09:734] niqme: reporting NS-12537 error as ORA-12537
[22-MAY-2009 13:35:09:734] niqme: exit
[22-MAY-2009 13:35:09:734] niomapnserror: returning error 12537
[22-MAY-2009 13:35:09:734] niomapnserror: exit
[22-MAY-2009 13:35:09:734] niotns: Couldn't connect, returning 12537
[22-MAY-2009 13:35:10:734] niotns: exit
[22-MAY-2009 13:35:10:734] nsbfrfl: entry
[22-MAY-2009 13:35:10:734] nsbrfr: entry
[22-MAY-2009 13:35:10:734] nsbrfr: nsbfs at 0x132d90, data at 0x132dc8.
[22-MAY-2009 13:35:10:734] nsbrfr: normal exit
[22-MAY-2009 13:35:10:734] nsbrfr: entry
[22-MAY-2009 13:35:10:734] nsbrfr: nsbfs at 0x1248d8, data at 0x132210.
[22-MAY-2009 13:35:10:734] nsbrfr: normal exit
[22-MAY-2009 13:35:10:734] nsbrfr: entry
[22-MAY-2009 13:35:10:734] nsbrfr: nsbfs at 0x12d820, data at 0x1319f0.
[22-MAY-2009 13:35:10:734] nsbrfr: normal exit
[22-MAY-2009 13:35:10:734] nsbfrfl: normal exit
[22-MAY-2009 13:35:10:734] nigtrm: Count in the NI global area is now 1
[22-MAY-2009 13:35:10:734] nigtrm: Count in the NL global area is now 1
```

Note: In the error stack in the preceding example, an operating system error code is shown. Each operating system has its own error codes, refer to your system documentation for information about the operating system error code.

The most efficient way to evaluate error codes is to find the most recent `nserror` entry logged, as the session layer controls the connection. The most important error messages are the ones at the bottom of the file. They are the most recent errors and the source of the problem with the connection.

For information about the specific return codes, use the Oracle error tool `oerr`, by entering the following at any command line:

```
oerr tns error_number
```

As an example, consider the following `nserror` entry logged in the trace file shown in Example 16–12 on page 16-42:

```
[22-MAY-2009 13:35:09:625] nserror: nsres: id=0, op=68, ns=12537, ns2=12560;
nt[0]=507, nt[1]=0, nt[2]=0; ora[0]=0, ora[1]=0, ora[2]=0
```

In the preceding entry, the main TNS error is 12537, and its secondary error is 12560. The protocol adapter error is 507. Using `oerr`, you can find out more information about return codes 12537, 12560, and 507. User input is shown in bold in the following examples.

```
oerr tns 12537
12537, 00000, "TNS:connection closed"
// *Cause: "End of file" condition has been reached; partner has disconnected.
// *Action: None needed; this is an information message.

oerr tns 12560
12560, 00000, "TNS:protocol adapter error"
// *Cause: A generic protocol adapter error occurred.
// *Action: Check addresses used for proper protocol specification. Before
// reporting this error, look at the error stack and check for lower level
// transport errors. For further details, turn on tracing and reexecute the
// operation. Turn off tracing when the operation is complete.

oerr tns 507
00507, 00000, "Connection closed"
// *Cause: Normal "end of file" condition has been reached; partner has
//  disconnected.
// *Action: None needed; this is an information message.
```

Using the Trace Assistant to Examine Trace Files

Oracle Net Services provides a tool called the **Trace Assistant** to help you understand the information provided in trace files by converting existing lines of trace file text into a more readable paragraph. The Trace Assistant works only with level 16 (support) Oracle Net Services trace files.

> **Note:** The Trace Assistant can only be used when `DIAG_ADR_ENABLED` is set to `off` (see "Understanding Automatic Diagnostic Repository" on page 16-1).

This section contains the following topics:

- Trace Assistant Syntax
- Packet Examples
- Two-Task Common Packet Examples
- Connection Example
- Statistics Example

Trace Assistant Syntax

To run the Trace Assistant, enter the following command at any command line prompt:

```
trcasst [options] filename
```

The options are described in Table 16–22.

Table 16–22 Trace Assistant Syntax

Option	Description
-e*level*	Displays error information. After the `-e`, use `0`, `1`, or `2` error decoding level may follow: ■ `0` or nothing translates the NS error numbers dumped from the `nserror` function plus lists all other errors ■ `1` displays only the NS error translation from the `nserror` function ■ `2` displays error numbers without translation
-la	If a connection ID exists in the NS connect packet, then the output displays the connection IDs. Connection IDs are displayed as hexadecimal, eight-byte IDs. A generated ID is created by Trace Assistant if the packet is not associated with any connection, that is, the connect packet is overwritten in the trace file. This can occur with cyclic trace files. For each ID, the output lists the following: ■ Socket ID, if the connection has one. ■ Connect packet send or receive operation. ■ Current setting of the `MULTIPLEX` attribute of the `DISPATCHERS` parameter in the initialization parameter file. When `MULTIPLEX` is set to `ON`, **session multiplexing** is enabled. ■ Session ID, if `MULTIPLEX` is set to `ON`. ■ Connect data information. **Notes:** ■ Do not use this option with other options. ■ The IDs generated by the Trace Assistant do not correlate with client/server trace files.
-li ID	Displays the trace for a particular ID from the `-la` output **Note:** Only use this option with output from the `-la` option.
-o*type*	Displays the amount and type of information to be output. After the `-o` the following options can be used: ■ c to display summary connectivity information. ■ d to display detailed connectivity information. ■ u to display summary **Two-Task Common (TTC)** information. ■ t to display detailed TTC information. ■ q to display SQL commands enhancing summary TTC information. Use this option with `u`, such as `-ouq`. **Note:** As output for `d` contains the same information as displayed for `c`, do not submit both `c` and `d`. If you submit both, then only output `d` is processed.

Table 16–22 (Cont.) Trace Assistant Syntax

Option	Description
-s	Displays the following statistical information: ■ Total number of bytes sent and received. ■ Maximum open cursors. ■ Currently open cursors. ■ Count and ratio of operations. ■ Parsing and execution count for PL/SQL. ■ Total calls sent and received. ■ Total, average, and maximum number of bytes sent and received. ■ Total number of transports and sessions present. ■ Timestamp information, if any. ■ Sequence numbers, if any.

If no options are provided, then the default is `-odt -e0 -s`, which provides detailed connectivity and TTC events, error level zero (`0`), and statistics in the trace file.

Example 16–13 shows how the Trace Assistant converts the trace file information into a more readable format using the `-e1` option.

Example 16–13 trcasst -e1 Output

```
    ************************************************************************
    *                          Trace Assistant                             *
    ************************************************************************

ntus2err: exit
ntuscni: exit
ntusconn: exit
nserror: entry
-<ERROR>- nserror: nsres: id=0, op=65, ns=12541, ns2=12560; nt[0]=511, nt[1]=2,
nt[2]=0
//////////////////////////////////////////////////////////////
Error found. Error Stack follows:
              id:0
  Operation code:65
      NS Error 1:12541
      NS Error 2:12560
NT Generic Error:511
  Protocol Error:2
        OS Error:0
 NS & NT Errors Translation
12541, 00000 "TNS:no listener"
 // *Cause: The connection request could not be completed because the listener
 // is not running.
 // *Action: Ensure that the supplied destination address matches one of
 // the addresses used by the listener - compare the TNSNAMES.ORA entry with
 // the appropriate LISTENER.ORA file (or TNSNAV.ORA if the connection is to
 // go by way of an Interchange). Start the listener on the remote machine.
/
12560, 00000 "TNS:protocol adapter error"
 // *Cause: A generic protocol adapter error occurred.
 // *Action: Check addresses used for proper protocol specification. Before
 // reporting this error, look at the error stack and check for lower level
```

```
 // transport errors.For further details, turn on tracing and reexecute the
 // operation. Turn off tracing when the operation is complete.
/
00511, 00000 "No listener"
 // *Cause: The connect request could not be completed because no application
 // is listening on the address specified, or the application is unable to
 // service the connect request in a sufficiently timely manner.
 // *Action: Ensure that the supplied destination address matches one of
 // the addresses used by the listener - compare the TNSNAMES.ORA entry with
 // appropriate LISTENER.ORA file (or TNSNAV.ORA if the connection is to go
 // by way of an Interchange. Start the listener on the remote machine.
/
////////////////////////////////////////////////////////////
    ************************************************************************
    *                    Trace Assistant has completed                     *
    ************************************************************************
```

However, other errors may also exist within the trace file that were not logged from the `nserror` function.

Packet Examples

Trace Assistant also enables you to view data packets from both the Oracle Net and TTC communication layers. Trace Assistant offers two options to view these packets:

- Summary connectivity (using option `-oc`)
- Detailed connectivity (using option `-od`)

Example 16–14 shows summary information from the `-oc` option.

Example 16–14 Summary Information from trcasst -oc Output

```
    ************************************************************************
    *                         Trace Assistant                              *
    ************************************************************************

---> Send 198 bytes - Connect packet
Connect data length: 140
Connect Data:
    (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=)
    (HOST=sales-server)(USER=joe))))

<--- Received 76 bytes - Redirect packet
Redirect data length: 66
Redirect Data:
    (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))

---> Send 198 bytes - Connect packet
Connect data length: 140
Connect Data:
    (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
    (CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=)
    (HOST=sales-server)(USER=joe))))

<--- Received 32 bytes - Accept packet
Connect data length: 0
---> Send 153 bytes - Data packet
        Native Services negotiation packet
```

```
<--- Received 127 bytes - Data packet
        Native Services negotiation packet

---> Send 32 bytes - Data packet

<--- Received 140 bytes - Data packet

    ************************************************************************
    *                    Trace Assistant has completed                     *
    ************************************************************************
```

The packets being sent or received have a prefix of `---> Send` *nnn* `bytes` or `<--- Received` *nnn* `bytes` showing that this node is sending or receiving a packet of a certain type and with *nnn* number of bytes. This prefix enables you to determine if the node is the client or the database server. The connection request is always sent by the client, and received by the database server or listener.

Example 16–15 shows detailed information from the `-od` option. The output shows all of the details sent along with the connect data in negotiating a connection.

Example 16–15 Detailed Information from trcasst -od Output

```
    ************************************************************************
    *                          Trace Assistant                             *
    ************************************************************************
--->  Send   241 bytes - Connect packet
Current NS version number is: 311.
Lowest NS version number can accommodate is: 300.
Global options for the connection:
      can receive attention
      no attention processing
      Don't care
      Maximum SDU size:8192
      Maximum TDU size:32767
      NT protocol characteristics:
            Test for more data
            Test operation
            Full duplex I/O
            Urgent data support
            Generate SIGURG signal
            Generate SIGPIPE signal
            Generate SIGIO signal
            Handoff connection to another
      Line turnaround value :0
      Connect data length :183
      Connect data offset :58
      Connect data maximum size :512
            Native Services wanted
            NAU doing O3LOGON - DH key foldedin
            Native Services wanted
            NAU doing O3LOGON - DH key foldedin
      Cross facility item 1: 0
      Cross facility item 2: 0
      Connection id : Ox000059F70000004C
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(SRVR=SHARED)(CID=(PROGRAM=)
(HOST=sales-server)(USER=joe))))

<--- Received 76 bytes - Redirect packet
     Redirect data length: 66
```

```
(ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521))

--->  Send   241 bytes - Connect packet
Current NS version number is: 311.
Lowest NS version number can accommodate is: 300.
Global options for the connection:
      can receive attention
      no attention processing
      Don't care
      Maximum SDU size:8192
      Maximum TDU size:32767
      NT protocol characteristics:
      Test for more data
      Test operation
      Full duplex I/O
      Urgent data support
      Generate SIGURG signal
      Generate SIGPIPE signal
      Generate SIGIO signal
      Handoff connection to another
Line turnaround value :0
Connect data length :183
Connect data offset :58
Connect data maximum size :512
      Native Services wanted
      NAU doing O3LOGON - DH key foldedin
      Native Services wanted
      NAU doing O3LOGON - DH key foldedin
Cross facility item 1: 0
Cross facility item 2: 0
Connection id : Ox000059F70000007A
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
(CONNECT_DATA=(SERVICE_NAME=sales.us.example.com)(SRVR=SHARED)(CID=(PROGRAM=)
(HOST=sales-server)(USER=joe))))
<--- Received 32 bytes - Accept packet
      Accepted NS version number is: 310.
Global options for the connection:
      no attention processing
      Don't care
      Accepted maximum SDU size: 8192
      Accepted maximum TDU size: 32767
      Connect data length: 0
            Native Services wanted
            NAU doing O3LOGON - DH key foldedin
            Native Services wanted
            NAU doing O3LOGON - DH key foldedin

--->  Send   153 bytes - Data packet
      Native Services negotiation packet version#: 150999040
            Service data packet #0 for Supervisor has 3 subpackets
               Subpacket #0:  Version #150999040
               Subpacket #1: 0000000000000000
               Subpacket #2: DEADBEEF0003000000040004000100010002
            Service data packet #1 for Authentication has 3 subpackets
               Subpacket #0:  Version #150999040
               Subpacket #1: UB2: 57569
               Subpacket #2: FCFF
            Service data packet #2 for Encryption has 2 subpackets
               Subpacket #0:  Version #150999040
               Subpacket #1: 000000000000000000
```

```
            Service data packet #3 for Data Integrity has 2 subpackets
               Subpacket #0:  Version #150999040
               Subpacket #1: 000000

<--- Received 127 bytes - Data packet
     Native Services negotiation packet version#: 135290880
            Service data packet #0 for Supervisor has 3 subpackets
               Subpacket #0:  Version #135290880
               Subpacket #1: 0000
               Subpacket #2: DEADBEEF00030000000200040001
            Service data packet #1 for Authentication has 2 subpackets
               Subpacket #0:  Version #135290880
               Subpacket #1: FBFF
            Service data packet #2 for Encryption has 2 subpackets
               Subpacket #0:  Version #135290880
               Subpacket #1: UB1: 0
            Service data packet #3 for Data Integrity has 2 subpackets
               Subpacket #0:  Version #135290880
               Subpacket #1: UB1: 0
....

--->  Send   11 bytes - Marker packet
      One data byte.
      Hex character sent over to the server: 2

<--- Received 11 bytes - Marker packet
     One data byte.
     Hex character sent over to the server: 2

<--- Received 155 bytes - Data packet

--->  Send   25 bytes - Data packet

<--- Received 11 bytes - Data packet

--->  Send   13 bytes - Data packet

<--- Received 11 bytes - Data packet

--->  Send   10 bytes - Data packet
      Data Packet flags:
      End of file
    ********************************************************************
    *                   Trace Assistant has completed                  *
    ********************************************************************
```

Two-Task Common Packet Examples

TTC handles requests such as open cursor, select rows, and update rows that are directed to the database server. All requests are answered by the database server. If you request to log on, then a response is returned from the database server that the request was completed.

Summary information for TTC from the `-ou` option is different from other displays in that it shows two packets on each line, rather than one. This is done to mirror the request/response pairings process by which TTC operates.

Output is displayed in the following format:

description TTC_message cursor_number SQL_statement bytes_sent bytes_received

Example 16–16 shows all of the details sent along with the connect data in negotiating a connection.

Example 16–16 trcasst -ou Output

```
*************************************************************************
*                        Trace Assistant                                *
*************************************************************************

                                                                  Bytes  Bytes
                                                                   Sent   Rcvd

Send operation(TTIPRO)                                               32    140
Send operation(TTIDTY)                                               33     22
Get the session key (OSESSKEY)                                      229    145
Generic authentication call (OAUTH)                                 368   1001
Send operation(TTIPFN)                                               44    144
Send operation(TTIPFN)                                               36     16
Parse a statement (OSQL)                  # 1  SELECT USER FROM ...   47    100
Fast upi calls to opial7 (OALL7)          # 1                        130    111
Fetch row (OFETCH)                        # 1                         21    137
Close cursor (OCLOSE)                     # 1                         17     11
New v8 bundled call (OALL8)               # 0  !Keep Parse  BEGI...   156    145
Send operation(TTIPFN)                                               51     16
Parse a statement (OSQL)                  # 1  SELECT ATTRIBUTE,...   186    100
Fast upi calls to opial7 (OALL7)          # 1                        246    111
Fetch row (OFETCH)                        # 1                         21    126
Close cursor (OCLOSE)                     # 1                         17     11
Send operation(TTIPFN)                                               36     16
Parse a statement (OSQL)                  # 1  SELECT CHAR_VALUE...  208    100
Fast upi calls to opial7 (OALL7)          # 1                        130    111
Fetch row (OFETCH)                        # 1                         21    126
Close cursor (OCLOSE)                     # 1                         17     11
Send operation(TTIPFN)                                               36     16
Fast upi calls to opial7 (OALL7)          # 1  !Keep Parse  BEGI...   183     41
Send operation(TTIRXD)                                               20    111
Close cursor (OCLOSE)                     # 1                         17     11
New v8 bundled call (OALL8)               # 0  Parse Fetch  SELE...   165    278
Send operation(TTIPFN)                                               51     16
Parse a statement (OSQL)                  # 1  commit                 31    100
Execute statement (OEXEC)                 # 1  number of rows: 1      25    100
Close cursor (OCLOSE)                     # 1                         17     11
Send operation(TTIPFN)                                               36     16
Fast upi calls to opial7 (OALL7)          # 1  !Keep Parse  BEGI...   183     41
Send operation(TTIRXD)                                               60    111
Close cursor (OCLOSE)                     # 1                         17     11
Send operation(TTIPFN)                                               36     16
Fast upi calls to opial7 (OALL7)          # 1  !Keep Parse  BEGI...   183     41
Send operation(TTIRXD)                                               20    111
Close cursor (OCLOSE)                     # 1                         17     11
New v8 bundled call (OALL8)               # 0  Parse Fetch  sele...   144    383
New v8 bundled call (OALL8)               # 1  !Keep Fetch            121    315
Logoff off of Oracle (OLOGOFF)                                       13     11

*************************************************************************
*                  Trace Assistant has completed                        *
*************************************************************************
```

Example 16–17 shows detailed TTC information from the `-ot` option.

Example 16–17 Detailed TTC Information from trcasst -ot Output

```
    ********************************************************************************
    *                      Trace  Assistant                                        *
    ********************************************************************************

Set protocol (TTIPRO)
      Operation 01 (con) Send protocol version=6
      Originating platform: SVR4-be-8.1.0

Set protocol (TTIPRO)
      Operation 01 (con) Receive protocol version=6
      Destination platform: SVR4-be-8.1.0

Set datatypes (TTIDTY)

Set datatypes (TTIDTY)

Start of user function (TTIFUN)
      (OSESSKEY)

Return opi parameter (TTIRPA)

Start of user function (TTIFUN)
      (OAUTH)

Return opi parameter (TTIRPA)

Start of user function (TTIFUN)
      session operations 71 (O71SESOPN) (switch session)

Return opi parameter (TTIRPA)

Start of user function (TTIFUN)
      Get Oracle version/date string in new format (OVERSION)

Return opi parameter (TTIRPA)
Oracle Enterprise Edition Release 11.2.0.0.0
With the Partitioning option
JServer Release 11.2.0.0.0

Start of user function (TTIFUN)
      session operations 71 (O71SESOPN) (switch session)

Return opi parameter (TTIRPA)

Start of user function (TTIFUN)
      Open a cursor (OOPEN)

Return opi parameter (TTIRPA)
      Cursor #: 1

Start of user function (TTIFUN)
      Parse a statement (OSQL) Cursor # 1
SELECT USER FROM DUAL
    ********************************************************************************
    *                    Trace Assistant has completed                             *
    ********************************************************************************
```

Example 16–18 shows detailed SQL information from the `-ouq` option. On each line of the output, the first item displayed is the actual request made. The second item shows

on what cursor that operation has been performed. The third item is either a listing of the SQL command or flag that is being answered. The number of bytes sent and received are displayed at the far right. A flag can be one of the following:

- !PL/SQL = Not a PL/SQL request
- COM = Commit
- IOV = Get I/O Vector
- DEFN = Define
- EXEC = Execute
- FETCH = Fetch
- CAN = Cancel
- DESCSEL = Describe select
- DESCBND = Describe Bind
- BND = Bind
- PARSE = Parse
- EXACT = Exact

Example 16–18 Detailed SQL Information from trcasst -ouq Output

```
    ************************************************************************
    *                        Trace Assistant                               *
    ************************************************************************
                                                                      Bytes  Bytes
                                                                       Sent   Rcvd

Send operation(TTIPRO)                                                   32    140
Send operation(TTIDTY)                                                   33     22
Get the session key (OSESSKEY)                                          229    145
Generic authentication call (OAUTH)                                     368   1001
Send operation(TTIPFN)                                                   44    144
Send operation(TTIPFN)                                                   36     16
Parse a statement (OSQL)                 # 1                             47    100
          SELECT USER FROM DUAL

Fast upi calls to opial7 (OALL7)         # 1                            130    111
Fetch row (OFETCH)                       # 1                             21    137
Close cursor (OCLOSE)                    # 1                             17     11
New v8 bundled call (OALL8)              # 0  !Keep Parse                156    145
          BEGIN DBMS_OUTPUT.DISABLE; END;

Send operation(TTIPFN)                                                   51     16
Parse a statement (OSQL)                 # 1                            186    100
          SELECT ATTRIBUTE,SCOPE,NUMERIC_VALUE,CHAR_VALUE,DA
          TE_VALUE FROM SYSTEM.PRODUCT_PRIVS WHERE (UPPER('S
          QL*Plus') LIKE UPPER(PRODUCT)) AND (UPPER(USER) LI
          KE USERID)

Fast upi calls to opial7 (OALL7)         # 1                            246    111
Fetch row (OFETCH)                       # 1                             21    126
Close cursor (OCLOSE)                    # 1                             17     11
Send operation(TTIPFN)                                                   36     16
Parse a statement (OSQL)                 # 1                            208    100
          SELECT CHAR_VALUE FROM SYSTEM.PRODUCT_PRIVS WHERE
          (UPPER('SQL*Plus') LIKE UPPER(PRODUCT)) AND ((UPPE
```

```
          R(USER) LIKE USERID) OR (USERID = 'PUBLIC')) AND (
          UPPER(ATTRIBUTE) = 'ROLES')

Fast upi calls to opial7 (OALL7)         # 1                          130    111
Fetch row (OFETCH)                       # 1                           21    126
Close cursor (OCLOSE)                    # 1                           17     11
Send operation(TTIPFN)                                                 36     16
Fast upi calls to opial7 (OALL7)         # 1  !Keep Parse             183     41
          BEGIN DBMS_APPLICATION_INFO.SET_MODULE(:1,NULL); E
          ND;

Send operation(TTIRXD)                                                 20    111
Close cursor (OCLOSE)                    # 1                           17     11
New v8 bundled call (OALL8)              # 0  Parse Fetch             165    278
          SELECT DECODE('A','A','1','2') FROM DUAL

Send operation(TTIPFN)                                                 51     16
Parse a statement (OSQL)                 # 1                           31    100
          commit

Execute statement (OEXEC)                # 1  number of rows: 1        25    100
Close cursor (OCLOSE)                    # 1                           17     11
Send operation(TTIPFN)                                                 36     16
Fast upi calls to opial7 (OALL7)         # 1  !Keep Parse             183     41
          BEGIN DBMS_APPLICATION_INFO.SET_MODULE(:1,NULL); E
          ND;

Send operation(TTIRXD)                                                 60    111
Close cursor (OCLOSE)                    # 1                           17     11
Send operation(TTIPFN)                                                 36     16
Fast upi calls to opial7 (OALL7)         # 1  !Keep Parse             183     41
          BEGIN DBMS_APPLICATION_INFO.SET_MODULE(:1,NULL); E
          ND;

Send operation(TTIRXD)                                                 20    111
Close cursor (OCLOSE)                    # 1                           17     11
New v8 bundled call (OALL8)              # 0  Parse Fetch             144    383
          select * from dept

New v8 bundled call (OALL8)              # 1  !Keep Fetch             121    315
Logoff off of Oracle (OLOGOFF)                                         13     11

    ********************************************************************
    *                   Trace Assistant has completed                  *
    ********************************************************************
```

Connection Example

Example 16–19 shows output from the `-la` option. The output shows the following information:

- Connect IDs received
- Socket ID on which this connection has come
- Operation
  - `Receive` identifies the trace as a database server trace. In this example, `Receive` is the operation.
  - `Send` identifies the trace as a client trace.

`Receive` identifies the trace as a database server trace; `Send` identifies the trace as a client trace. In this output, `Receive` is the operation.

- `MULTIPLEX` attribute of the `DISPATCHERS` parameter is set to `ON`
- 32-bit session ID
- Connect data information received

Example 16–19 trcasst -la Output

```
    ************************************************************************
    *                        Trace Assistant                               *
    ************************************************************************

Connection ID: 00000B270000000B
        Socket Id: 15
        Operation: Receive
        Multiplex: ON
        Session Id: 8362785DE4FC0B19E034080020F793E1
        Connect Data:
        (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
        (CONNECT_DATA=(SERVER=shared)
        (SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=)(HOST=sales-server)
        (USER=oracle))))
Connection ID: 00000B240000000B
        Socket Id: 15
        Operation: Receive
        Multiplex: ON
        Session Id: 8362785DE4FB0B19E034080020F793E1
        Connect Data:
        (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
        (CONNECT_DATA=(SERVER=shared)
        (SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=)(HOST=sales-server)
        (USER=oracle))))
Connection ID: 00000B1F00000008
        Socket Id: 15
        Operation: Receive
        Multiplex: ON
        Session Id: 8362785DE4F90B19E034080020F793E1
        Connect Data:
        (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
        (CONNECT_DATA=(SERVER=shared)
        (SERVICE_NAME=sales.us.example.com)(CID=(PROGRAM=)(HOST=sales-server)
        (USER=oracle))))
    ************************************************************************
    *                   Trace Assistant has completed                      *
    ************************************************************************
```

Example 16–20 shows output for connection ID `00000B1F00000008` from the `-li 00000B1F00000008` option.

Example 16–20 trcasst -li Output

```
    ************************************************************************
    *                             Trace Assistant                          *
    ************************************************************************
<--- Received 246 bytes - Connect packet
Current NS version number is: 310.
Lowest NS version number can accommodate is: 300.
Global options for the connection:
```

```
        Can receive attention
        No attention processing
        Don't care
        Maximum SDU size: 8192
        Maximum TDU size: 32767
        NT protocol characteristics:
                Test for more data
                Test operation
                Full duplex I/O
                Urgent data support
                Generate SIGURG signal
                Generate SIGPIPE signal
                Generate SIGIO signal
                Handoff connection to another
        Line turnaround value: 0
        Connect data length: 188
        Connect data offset: 58
        Connect data maximum size: 512
                Native Services wanted
                NAU doing O3LOGON - DH key foldedin
                Native Services wanted
                NAU doing O3LOGON - DH key foldedin
        Cross facility item 1: 0
        Cross facility item 2: 0
        Connection id: Ox00000B1F00000008
    (DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(HOST=sales-server)(PORT=1521))
    (CONNECT_DATA=(SERVER=shared)(SERVICE_NAME=sales.us.example.com)
    (CID=(PROGRAM=)(HOST=sales-server)(USER=oracle))))

---> Send 114 bytes - Accept packet
Accepted NS version number is: 310.
Global options for the connection:
        No attention processing
        Don't care
        Accepted maximum SDU size: 8192
        Accepted maximum TDU size: 32767
        Connect data length: 0
                Native Services wanted
                NAU doing O3LOGON - DH key foldedin
                Native Services wanted
                NAU doing O3LOGON - DH key foldedin
        Connection Time out: 1000
        Tick Size: 100
        Reconnect Data: (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=34454))
        Session Id: 8362785DE4F90B19E034080020F793E1
<--- Received 164 bytes - Data packet
        Native Services negotiation packet version#: 135290880
                 Service data packet #0 for Supervisor has 3 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: 0000000000000000
                        Subpacket #2: DEADBEEF00030000000400040001000100002
                 Service data packet #1 for Authentication has 3 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: UB2: 57569
                        Subpacket #2: FCFF
                 Service data packet #2 for Encryption has 2 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: 0000000000
                 Service data packet #3 for Data Integrity has 2 subpackets
                        Subpacket #0:  Version #135290880
```

```
                        Subpacket #1: 0000
---> Send 143 bytes - Data packet
        Native Services negotiation packet version#: 135290880
                Service data packet #0 for Supervisor has 3 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: 0000
                        Subpacket #2: DEADBEEF00030000000200040001
                Service data packet #1 for Authentication has 2 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: FBFF
                Service data packet #2 for Encryption has 2 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: UB1: 0
                Service data packet #3 for Data Integrity has 2 subpackets
                        Subpacket #0:  Version #135290880
                        Subpacket #1: UB1: 0
<--- Received 48 bytes - Data packet
Set protocol (TTIPRO)
        Operation 01 (con) Receive protocol version=6
        Destination platform: SVR4-be-8.1.0
---> Send 156 bytes - Data packet
Set protocol (TTIPRO)
        Operation 01 (con) Send protocol version=6
        Originating platform: SVR4-be-8.1.0
<--- Received 49 bytes - Data packet
Set datatypes (TTIDTY)
---> Send 38 bytes - Data packet
Set datatypes (TTIDTY)
<--- Received 245 bytes - Data packet
Start of user function (TTIFUN)
        Get the session key (OSESSKEY)
---> Send 161 bytes - Data packet
Return opi parameter (TTIRPA)
...
    **************************************************************************
    *                    Trace Assistant has completed                       *
    **************************************************************************
```

Statistics Example

The type of statistics gathered is approximately the number of TTC calls, packets, and bytes were sent and received between the network partners. Example 16–21 shows typical trace file statistics from the `-s` option.

Example 16–21 trcasst -s Output

```
    **************************************************************************
    *                               Trace Assistant                          *
    **************************************************************************
----------------------
Trace File Statistics:
----------------------
Total number of Sessions: 3

DATABASE:
  Operation Count:    0 OPENS,    21 PARSES,    21 EXECUTES,     9 FETCHES
    Parse Counts:
       9 PL/SQL,     9 SELECT,      0 INSERT,     0 UPDATE,     0 DELETE,
       0 LOCK,       3 TRANSACT,    0 DEFINE,     0 SECURE,     0 OTHER
    Execute counts with SQL data:
```

```
      9 PL/SQL,     0 SELECT,      0 INSERT,     0 UPDATE,     0 DELETE,
      0 LOCK,       0 TRANSACT,    0 DEFINE,     0 SECURE,     0 OTHER

  Packet Ratio: 6.142857142857143 packets sent per operation
  Currently opened Cursors: 0
  Maximum opened Cursors  : 0

ORACLE NET SERVICES:
  Total Calls  :       129 sent,        132 received,          83 oci
  Total Bytes  :     15796 sent,      13551 received
    Average Bytes:       122 sent per packet,        102 received per packet
    Maximum Bytes:      1018 sent,        384 received

  Grand Total Packets:    129  sent,     132 received

    ************************************************************************
    *                    Trace Assistant has completed                     *
    ************************************************************************
```

Contacting Oracle Support Services

Some messages recommend contacting Oracle Support Services to report a problem. When you contact Oracle Support Services, have the following information available:

- The hardware, operating system, and release number of the operating system running Oracle Database.
- The complete release number of Oracle Database, such as release 11.2.0.1.0.
- All Oracle programs (with release numbers) in use when the error occurred, such as SQL*Plus release 11.2.0.1.0.
- If you encountered one or more error codes or messages, then the exact code numbers and message text, in the order in which they appeared.
- The problem severity, according to the following codes:
  - 1: Program not usable. Critical effect on operations.
  - 2: Program usable. Operations severely restricted.
  - 3: Program usable with limited functions. Not critical to overall operations.
  - 4: Problem circumvented by customer. Minimal effect, if any, on operations.

You will also be expected to provide the following:

- Your name
- The name of your organization
- Your Oracle Support ID number
- Your telephone number

Glossary

access control list (ACL)

The group of access directives that you define. The directives grant levels of access to specific data for specific clients or groups of clients.

ACL

See **access control list (ACL)**.

access control

A feature of Oracle Connection Manager that sets rules for denying or allowing certain clients to access designated servers.

address

See **protocol address**.

ADR

See **automatic diagnostic repository**.

alias

An alternative name for a network object in an Oracle Names server. An alias stores the name of the object it is referencing. When a client requests a lookup of an alias, Oracle completes the lookup as if it is the referenced object.

application gateway

A host computer that runs the **Oracle Net Firewall Proxy**. An application gateway looks and acts like a real server from the client's point of view, and a real client from the server's point of view. An application gateway sits between the Internet and company's internal network and provides middleman services (or proxy services) to users on either side.

ASCII character set

American Standard Code for Information Interchange character set, a convention for representing alphanumeric information using digital data. The collation sequence used by most computers with the exception of IBM and IBM-compatible computers.

attribute

A piece of information that describes some aspect of a directory entry. An entry comprises a set of attributes, each of which belongs to an **object class**. Moreover, each attribute has both a type—which describes the kind of information in the attribute—and a value—which contains the actual data.

authentication method

A security method that enables you to have high confidence in the identity of users, clients, and servers in distributed environments. Network authentication methods can also provide the benefit of single sign-on for users. The following authentication methods are supported, depending on whether **Oracle Advanced Security** is installed:

- RADIUS
- Kerberos
- **SSL**
- **Microsoft Windows NT native authentication**

automatic diagnostic repository

The automatic diagnostic repository (ADR) is a systemwide tracing and logging central repository. The repository is a file-based hierarchical data store for depositing diagnostic information, including network tracing and logging information.

cache

Memory that stores recently-accessed data so that subsequent requests to access the same data can be processed quickly.

CIDR

Classless Inter-Domain Routing. In CIDR notation, an IPv6 subnet is denoted by the subnet prefix and the size in bits of the prefix (in decimal), separated by the slash (`/`) character. For example, `2001:0DB8:0000:0000::/64` denotes a subnet with addresses `2001:0DB8:000:0000:0000:0000:0000:0000` through `2001:0DB8:000:0000:FFFF:FFFF:FFFF:FFFF`. The CIDR notation includes support for IPv4 addresses. For example, `192.168.2.1/24` denotes the subnet with addresses `192.168.2.1` through `192.168.2.255`.

Classless Inter-Domain Routing (CIDR)

See **CIDR**.

client

A user, software application, or computer that requests the services, data, or processing of another application or computer. The client is the user process. In a network environment, the client is the local user process and the server may be local or remote.

client load balancing

Load balancing, whereby if more than one listener services a single database, a client can randomly choose between the listeners for its connect requests. This randomization enables all listeners to share the burden of servicing incoming connect requests.

client profile

The properties of a client, which may include the preferred order of **naming method**s, client and server **logging** and **tracing**, the domain from which to request names, and other client options for **Oracle Advanced Security**.

client/server architecture

Software architecture based on a separation of processing between two CPUs. One CPU acts as the client in the transaction, requesting and receiving services. The other acts as the server that provides the requests.

cman.ora file

A configuration file that specifies protocol addresses for incoming requests and administrative commands, as well as Oracle Connection Manager parameters and **access control** rules.

CMADMIN (Oracle Connection Manager Administration)

An **Oracle Connection Manager** process that monitors the health of the listener and Oracle Connection Manager gateway processes, shutting down and starting processes as needed. `CMADMIN` registers information about gateway processes with the listener and processes commands run with the Oracle Connection Manager Control utility.

CMGW (Oracle Connection Manager gateway)

An **Oracle Connection Manager** process that receives client connections screened and forwarded by the listener located at the Oracle Connection Manager instance. The gateway process forwards the requests to the database server. In addition, it can multiplex or funnel multiple client connections through a single protocol connection.

connect data

A portion of the **connect descriptor** that defines the destination database **service name** or **Oracle system identifier (SID)**. In the following example, `SERVICE_NAME` defines a database service called `sales.us.example.com`:

```
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-server)(PORT=1521)
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.example.com))
```

connect descriptor

A specially formatted description of the destination for a network connection. A connect descriptor contains destination service and network route information.

The destination service is indicated by using its **service name**. The network route provides, at a minimum, the location of the listener through use of a network address.

connect identifier

A **connect descriptor** or a name that maps to a connect descriptor. A connect identifier can be a **net service name**, database **service name**, or **net service alias**. Users initiate a connect request by passing a user name and password along with a connect identifier in a connect string for the service to which they want to connect:

`CONNECT` *`username@connect_identifier`*

connect string

Information the user passes to a service to connect, such as user name, password, and **connect identifier**:

`CONNECT` *`username@net_service_name`*

connect-time failover

A client connect request is forwarded to a another listener if a listener is not responding. Connect-time failover is enabled by **service registration**, because the listener knows if an instance is running to attempt a connection.

connection

An interaction between two processes on a network. Connections are originated by an initiator (client) that requests a connection with a destination (server).

connection load balancing

The method for balancing the number of active connections for the same service across the instances and dispatchers. Connection load balancing enables listeners to make routing decisions based on how many connections for each dispatcher and the load on the nodes.

connection pooling

A resource utilization and user scalability feature that enables you to maximize the number of sessions over a limited number of protocol connections to a **shared server**.

connection request

A notification sent by an initiator and received by a listener that indicates that the initiator wants to start a connection.

data packet

See **packet**.

database administrator (DBA)

(1) A person responsible for operating and maintaining an Oracle Server or a database application. (2) An Oracle user name that has been given DBA privileges and can perform database administration functions. Usually the two meanings coincide. Many sites have multiple DBAs.

Database Configuration Assistant

A tool that enables you to create, delete, and modify a database.

database link

A pointer that defines a one-way communication path from an Oracle database server to another database server. The link is a defined entry in a data dictionary table. To access the link, the user must be connected to the local database that contains the data dictionary entry.

A client connected to local database A can use a link stored in database A to access information in remote database B. However, users connected to database B cannot use the same link to access data in database A. If local users on database B want to access data on database A, then a link must be defined and stored in the data dictionary of database B.

The following database links are supported:

- A **private database link** in a specific schema of a database. Only the owner of a private database link can use it.
- A **public database link** for a database. All users in the database can use it.

dedicated connection

A dedicated server with a database session.

dedicated server

A server process that is dedicated to one client connection. Contrast with **shared server process**.

default domain

The **domain** within which most client requests take place. It could be the domain where the client resides, or it could be a domain from which the client requests

network services often. Default domain is also the client configuration parameter that determines what domain should be appended to unqualified network name requests. A name request is unqualified if it does not have a "." character within it.

directory information tree (DIT)

A hierarchical tree-like structure in a **directory server** of the **distinguished names (DNs)** of the entries.

directory naming

A **naming method** that resolves a database service, **net service name**, or **net service alias** to a **connect descriptor** stored in a central directory server. A **directory server** provides central administration of directory naming objects, reducing the work effort associated with adding or relocating services.

directory server

A directory server that is accessed with the **Lightweight Directory Access Protocol (LDAP)**. Support of LDAP-compliant directory servers provides a centralized vehicle for managing and configuring a distributed Oracle network. The directory server can replace client-side and server-side localized `tnsnames.ora` files.

dispatcher

A process that enables many clients to connect to the same server without the need for a dedicated server process for each client. A dispatcher handles and directs multiple incoming network session requests to shared server processes. See also **shared server**.

distinguished name (DN)

Name of entry in a **directory server**. The DN specifies where the entry resides in the LDAP directory hierarchy, much the way a directory path specifies the exact location of a file.

distributed processing

Division of front-end and back-end processing to different computers. Oracle Network Services support distributed processing by transparently connecting applications to remote databases.

domain

Any tree or subtree within the **Domain Name System (DNS)** namespace. Domain most commonly refers to a group of computers whose host names share a common suffix, the domain name.

Domain Name System (DNS)

A system for naming computers and network services that is organized into a hierarchy of **domain**s. DNS is used in TCP/IP networks to locate computers through user-friendly names. DNS resolves a friendly name into an **IP address**, which is understood by computers.

For Oracle Network Services, DNS translates the host name in a TCP/IP address into an IP address.

DNS

Domain Name System. See **Domain Name System (DNS)**.

enterprise role

An enterprise role is analogous to a regular database role, except that it spans authorization on multiple databases. An enterprise role is a category of roles that define privileges on a particular database. An enterprise role is created by the database administrator of a particular database. An enterprise role can be granted to or revoked from one or more enterprise users. The information for granting and revoking these roles is stored in the directory server.

enterprise user

A user that has a unique identity across an enterprise. Enterprise users connect to individual databases through a schema. Enterprise users are assigned enterprise roles that determine their access privileges on databases.

entry

The building block of a directory server, it contains information about an object of interest to directory users.

external naming

A **naming method** that uses a third-party naming service, such as **NIS**.

external procedure

Function or procedure written in a third-generation language (3GL) that can be called from PL/SQL code. Only C is supported for external procedures.

failover

See **connect-time failover**.

firewall support

See **access control**.

foreign domains

The set of domains not managed within a given administrative region. Domains are foreign only in relation to a region; they are not foreign in any absolute sense. A network administrator typically defines foreign domains relative to a particular region to optimize caching performance.

FTP protocol

File Transfer Protocol. A client/server protocol which allows a user on one computer to transfer files to and from another computer over a TCP/IP network.

global database name

The full name of the database which uniquely identifies it from any other database. The global database name is of the form "*database_name*.*database_domain*," for example, `sales.us.example.com`.

The database name portion, `sales`, is a simple name you want to call your database. The database domain portion, `us.example.com`, specifies the database domain in which the database is located, making the global database name unique. When possible, Oracle recommends that your database domain mirror the network domain.

The global database name is the default service name of the database, as specified by the `SERVICE_NAMES` parameter in the initialization parameter file.

Heterogeneous Services

An integrated component that provides the generic technology for accessing third-party systems from the Oracle database server. Heterogeneous Services enables you to:

- Use Oracle SQL to transparently access data stored in third-party systems as if the data resides within an Oracle server.
- Use Oracle procedure calls to transparently access third-party systems, services, or application programming interfaces (APIs), from your Oracle distributed environment.

hierarchical naming model

An infrastructure in which names are divided into multiple hierarchically-related domains. For Oracle Names, hierarchical naming model can be used with either central or delegated administration.

host naming

A **naming method** resolution that enables users in a TCP/IP environment to resolve names through their existing name resolution service. This name resolution service might be **Domain Name System (DNS)**, **Network Information Service (NIS)**, or simply a centrally-maintained set of `/etc/hosts` files. Host Naming enables users to connect to an Oracle database server by simply providing the server computer's host name or host name alias. No client configuration is required to take advantage of this feature. This method is recommended for simple TCP/IP environments.

HTTP protocol

Hypertext Transfer Protocol. A protocol that provides the language that enables Web browsers and application Web servers to communicate.

identity management realm

A collection of identities, all of which are governed by the same administrative policies. In an enterprise, all employees having access to the intranet may belong to one realm, while all external users who access the public applications of the enterprise may belong to another realm. An identity management realm is represented in the directory by a specific entry with a special object class associated with it.

instance

The combination of the **System Global Area (SGA)** and the Oracle background processes. When a database is started on a database server (regardless of the type of computer), Oracle allocates a memory area called the SGA and starts one or more Oracle processes. The memory and processes of an instance efficiently manage the associated database's data and serve the database users. You can connect to any instance to access information within a cluster database.

instance name

A name of an Oracle database instance. The instance name is identified by the `INSTANCE_NAME` parameter in the database initialization parameter file. `INSTANCE_NAME` corresponds to the **Oracle system identifier (SID)** of the instance. Clients can connect to a specific instance by specifying the `INSTANCE_NAME` parameter in the connect descriptor.

The instance name is included in the **connect data** part of the **connect descriptor**.

Interprocess Communication (IPC)

A protocol used by client applications that resides on the same node as the listener to communicate with the database. IPC can provide a faster local connection than TCP/IP.

IP address

Used to identify a node on a network. Each computer on the network is assigned a unique Internet Protocol (IP) address, which is made up of the network ID and a unique host ID.

IPv4

Internet Protocol Version 4. IPv4 is the current standard for the IP protocol. IPv4 uses 32-bit (four-byte) addresses, which are typically represented in dotted-decimal notation. The decimal value of each octet is separated by a period, as in `192.168.2.22`.

IPv6

Internet Protocol Version 6. The protocol designed to replace **IPv4**. In IPv6, an IP address is typically represented in eight fields of hexadecimal values separated by colons, as in `2001:0DB8:0000:0000:0000:0000:1428:57AB`. In some cases, fields with `0` values can be compressed, as in `2001:DB8::1428:57AB`.

IP Version 4 (IPv4)

See **IPv4**

IP Version 6 (IPv6)

See **IPv6**

IPC

See **Interprocess Communication (IPC)**.

Java Database Connectivity (JDBC) Driver

A driver that provides Java applications and applets access to an Oracle database.

JDBC OCI Driver

A Type II driver for use with client/server Java applications. This driver requires an Oracle client installation.

JDBC Thin Driver

A Type IV driver for Oracle JDBC applets and applications. Because it is written entirely in Java, this driver is platform-independent. It does not require any additional Oracle software on the client side. The Thin driver communicates with the server using **Two-Task Common (TTC)**, a protocol developed by Oracle to access the database server.

keyword-value pair

The combination of a keyword and a value, used as the standard unit of information in connect descriptors and many configuration files. Keyword-value pairs may be nested; that is, a keyword may have another keyword-value pair as its value.

latency

Networking round-trip time.

Lightweight Directory Access Protocol (LDAP)

A standard, extensible directory access protocol. It is a common language that LDAP clients and servers use to communicate. The framework of design conventions supporting industry-standard **directory server**s.

LDAP Data Interchange Format (LDIF)

The set of standards for formatting an input file for any of the LDAP command line utilities.

ldap.ora file

A file created by Oracle Internet Directory Configuration Assistant or Oracle Net Configuration Assistant that contains the following directory server access information:

- Type of directory server
- Location of the directory server
- Default Oracle Context that the client or server use to look up or configure connect identifiers for connections to database services

When created with Oracle Internet Directory Configuration Assistant, `ldap.ora` is located in the `ORACLE_HOME/ldap/admin` directory. When created with Oracle Net Configuration Assistant, `ldap.ora` is located in the `ORACLE_HOME/network/admin` directory.

link qualifier

A qualifier appended to a global database link to provide alternate settings for the database user name and password credentials. For example, a link qualifier of `fieldrep` can be appended to a global database link of `sales.us.example.com`.

```
SQL> SELECT * FROM emp@sales.us.example.com@fieldrep
```

listener

See **Oracle Net Listener**.

listener.ora file

A configuration file for Oracle Net Listener that identifies the following:

- Unique name
- Protocol addresses that it is accepting connection requests on
- Services it is listening for

The `listener.ora` file typically resides in the `ORACLE_HOME/network/admin` directory.

Oracle Database does not require identification of the database service because of **service registration**. However, static service configuration is required if you plan to use Oracle Enterprise Manager.

Listener Control utility

A utility included with Oracle Network Services to control various listener functions, such as starting, stopping, and getting the status of the listener.

load balancing

A feature by which client connections are distributed evenly among multiple listeners, dispatchers, instances, and nodes so that no single component is overloaded.

Oracle Network Services support **client load balancing** and **connection load balancing**.

local naming

A **naming method** that locates network addresses by using information configured and stored on each individual client's **tnsnames.ora file**. Local naming is most appropriate for simple distributed networks with a small number of services that change infrequently.

location transparency

A distributed database characteristic that enables applications to access data tables without knowing where they reside. All data tables appear to be in a single database, and the system determines the actual data location based on the table name. The user can reference data on multiple nodes in a single statement, and the system automatically and transparently routes (parts of) SQL statements to remote nodes, if needed. The data can move among nodes with no impact on the user or application.

logging

A feature in which errors, service activity, and statistics are written to a log file. The log file provides additional information for an administrator when the error message on the screen is inadequate to understand the failure. The log file, by way of the error stack, shows the state of the software at various layers.

See also **tracing**.

loopback test

A connection from the server back to itself. Performing a successful loopback verifies that Oracle Net is functioning on the database server.

map

Files used by the **Network Information Service (NIS)** `ypserv` program to handle name requests.

Microsoft Active Directory

An LDAP-compliant directory server included with the Microsoft Windows 2000 Server. It stores information about objects on the network, and makes this information available to users and network administrators. Active Directory also provides access to resources on the network using a single logon process.

Active Directory can be configured as a directory naming method to store service information that clients can access.

Microsoft Windows NT native authentication

An **authentication method** that enables a client to have single login access to a Microsoft Windows NT server and a database running on the server.

Named Pipes protocol

A high-level interface protocol providing interprocess communications between clients and servers using distributed applications.

naming context

A subtree that resides entirely on one directory server. It is a contiguous subtree, that is, it must begin at an entry that serves as the top of the subtree, and extend downward to either leaf entries or references to subordinate naming contexts. It can range in size from a single entry to the entire **directory information tree (DIT)**.

Oracle Context can be created under a naming context.

naming method

The resolution method used by a client application to resolve a **connect identifier** to a **connect descriptor** when attempting to connect to a database service. Oracle Net provides four naming methods:

- **local naming**
- **directory naming**
- Easy Connect naming
- **external naming**

net service alias

An alternative name for a **directory naming** object in a directory server. A directory server stores net service aliases for any defined **net service name** or database service. A net service alias entry does not have connect descriptor information. Instead, it only references the location of the object for which it is an alias. When a client requests a directory lookup of a net service alias, the directory determines that the entry is a net service alias and completes the lookup as if the alias was actually the entry it is referencing.

net service name

A simple name for a service that resolves to a **connect descriptor**. Users initiate a connect request by passing a user name and password, along with a net service name in a connect string, for the service to which they want to connect:

```
CONNECT username@net_service_name
```

Depending on your needs, net service names can be stored in a variety of places, including:

- Local configuration file, `tnsnames.ora`, on each client
- Directory server
- External naming service, such as **NIS**

network

A group of two or more computers linked together through hardware and software to allow the sharing of data and peripherals.

network administrator

The person who performs network management tasks such as installing, configuring, and testing network components. The administrator typically maintains the configuration files, connect descriptors and service names, aliases, and public and global database links.

network character set

As defined by Oracle, the set of characters acceptable for use as values in keyword-value pairs (that is, in connect descriptors and configuration files). The set includes alphanumeric upper- and lowercase, and some special characters.

Network Information Service (NIS)

Sun Microsystems Yellow Pages (`yp`) client/server protocol for distributing system configuration data such as user and host names between computers on a network.

Network Interface (NI)

A network layer that provides a generic interface for Oracle clients, servers, or external processes to access Oracle Net functions. The NI layer handles the "break" and "reset" requests for a connection.

network listener

See **listener**.

network object

Any service that can be directly addressed on a network; for example, a listener.

network protocol

See **Oracle protocol support**.

Network Program Interface (NPI)

An interface for server-to-server interactions that performs all of the functions that the **OCI** does for clients, allowing a coordinating server to construct SQL requests for additional servers.

Network Session (NS)

A **session layer** that is used in typical Oracle Net connections to establish and maintain the connection between a client application and a database server.

NI

Network Interface

NIS

See **Network Information Service (NIS)**.

node

A computer or terminal that is part of a network

NPI

See **Network Program Interface (NPI)**.

NR

Network Routing

NS

Network Session. See **Network Session (NS)**.

NT

Network Transport. See **transport**.

object class

In a directory server, a named group of attributes. When you want to assign attributes to an entry, you do so by assigning to that entry the object classes that hold those attributes.

All objects associated with the same object class share the attributes of that object class.

OCI

Oracle Call Interface. See **Oracle Call Interface (OCI)**.

OPI

See **Oracle Program Interface (OPI)**.

Open Systems Interconnection (OSI)

A model of network architecture developed by International Organization for Standardization (ISO) as a framework for international standards in heterogeneous computer network architecture.

The OSI architecture is split between seven layers, from lowest to highest:

1. Physical layer
2. Data link layer
3. Network layer
4. Transport layer
5. Session layer
6. Presentation layer
7. Application layer

Each layer uses the layer immediately preceding it and provides a service to the layer following it.

Oracle Advanced Security

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

Oracle Call Interface (OCI)

An application programming interface (API) that enables you to create applications that use the native procedures or function calls of a third-generation language to access an Oracle database server and control all phases of SQL statement execution. OCI supports the data types, calling conventions, syntax, and semantics of a number of third-generation languages including C, C++, COBOL and FORTRAN.

Oracle Connection Manager

A router through which a client connection request may be sent either to its next hop or directly to the database server. Clients who route their connection requests through an Oracle Connection Manager can then take advantage of the **session multiplexing**, **access control**, or **protocol conversion** features configured on that Oracle Connection Manager.

Oracle Connection Manager Control utility

A utility included with Oracle Network Services to control various functions, such as starting, stopping, and getting the status of the Oracle Connection Manager.

Oracle Context

An entry in an LDAP-compliant Internet directory called `cn=OracleContext`, under which all Oracle software relevant information is kept, including entries for Oracle Net Services directory naming and checksumming security. There may be one or more than one Oracle Context in a directory. An Oracle Context can be associated with a directory naming context.

Oracle Internet Directory automatically creates an Oracle Context at the root of the DIT structure. This root Oracle Context has a DN of `dn:cn=OracleContext`.

Oracle Enterprise Manager

A separate Oracle product that combines a graphical console, agents, common services, and tools to provide an integrated and comprehensive systems management platform for managing Oracle products.

Oracle Identity Management

An infrastructure enabling deployments to manage centrally and securely all enterprise identities and their access to various applications in the enterprise.

Oracle Internet Directory

A directory server implemented as an application on the Oracle database. It enables retrieval of information about dispersed users and network resources. It combines **Lightweight Directory Access Protocol (LDAP)** Version 3, the open Internet standard directory server access protocol, with the high performance, scalability, robustness, and availability of the Oracle database.

Oracle Net

Communication software that enables a network session from a client application to an Oracle database server. After a network session is established, Oracle Net acts as a data courier for the client application and the database server. It is responsible for establishing and maintaining the connection between the client application and database server, as well as exchanging messages between them. Oracle Net is able to perform these jobs because it is located on each computer in the network.

Oracle Net Configuration Assistant

A postinstallation tool that configures basic network components after installation, including:

- Listener names and protocol addresses
- Naming methods the client uses to resolve **connect identifier**s
- Net service names in a `tnsnames.ora` file
- Directory server usage

Oracle Net Firewall Proxy

Product offered by some firewall vendors that supplies **Oracle Connection Manager** functionality.

Oracle Net foundation layer

A networking communication layer that is responsible for establishing and maintaining the connection between the client application and server, as well as exchanging messages between them.

Oracle Net Listener

A process that resides on the server whose responsibility is to listen for incoming client connection requests and manage the traffic to the server.

When a client requests a network session with a database server, a listener receives the actual request. If the client information matches the listener information, then the listener grants a connection to the database server.

Oracle Net Manager

A tool that combines configuration abilities with component control to provide an integrated environment for configuring and managing Oracle Net Services.

You can use Oracle Net Manager to configure the following network components:

- Naming

  Define **connect identifier**s and map them to **connect descriptor**s to identify the network location and identification of a service. Oracle Net Manager supports configuration of connect descriptors in a local `tnsnames.ora` file or directory server.

- Naming Methods

  Configure the different ways in which connect identifiers are resolved into connect descriptors.

- Listeners

  Create and configure listeners to receive client connections.

Oracle Net Services

A suite of networking components that provide enterprise-wide connectivity solutions in distributed, heterogeneous computing environments. Oracle Net Services is comprised of **Oracle Net**, **listener**, **Oracle Connection Manager**, **Oracle Net Configuration Assistant**, and **Oracle Net Manager**.

Oracle Program Interface (OPI)

A networking layer responsible for responding to each of the possible messages sent by **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.

Oracle protocol support

A software layer responsible for mapping **Transparent Network Substrate (TNS)** functionality to industry-standard protocols used in the client/server connection.

Oracle Rdb

A database for Digital 64-bit platforms. Because Oracle Rdb has its own listener, the client interacts with Rdb in the same manner as it does with an Oracle database.

Oracle schema

A set of rules that determine what can be stored in a **directory server**. Oracle has its own schema that is applied to many types of Oracle entries, including Oracle Net

Services entries. The Oracle schema for Oracle Net Services' entries includes the attributes the entries may contain.

Oracle system identifier (SID)

A name that identifies a specific instance of a database. For any database, there is at least one instance referencing the database.

Oracle XML DB

A high-performance XML storage and retrieval technology provided with Oracle database server. It is based on the W3C XML data model.

Oracle Real Application Clusters (Oracle RAC)

An architecture that allows multiple instances to access a shared database of datafiles. Oracle Real Application Clusters is also a software component that provides the necessary cluster database scripts, initialization files, and datafiles needed for Oracle Enterprise Edition and Oracle Real Application Clusters.

ORACLE_HOME

An alternate name for the top directory in the Oracle directory hierarchy on some directory-based operating systems.

OSI

See **Open Systems Interconnection (OSI)**.

packet

A block of information sent over the network each time a connection or data transfer is requested. The information contained in packets depends on the type of packet: connect, accept, redirect, data, and so on. Packet information can be useful in troubleshooting.

PMON process

A process monitor database process that performs process recovery when a user process fails. PMON is responsible for cleaning up the cache and freeing resources that the process was using. PMON also checks on dispatcher and server processes and restarts them if they have failed. As a part of **service registration**, PMON registers instance information with the listener.

presentation layer

A networking communication layer that manages the representation of information that application layer entities either communicate or reference in their communication. **Two-Task Common (TTC)** is an example of presentation layer.

private database link

A database link created by one user for his or her exclusive use.

See also **database link** and **public database link**.

profile

A collection of parameters that specifies preferences for enabling and configuring Oracle Net Services' features on the client or server. A profile is stored and implemented through the `sqlnet.ora` file.

protocol

A set of rules that defines how data is transported across the network.

protocol address

An address that identifies the network address of a network object.

When a connection is made, the client and the receiver of the request, such as the **listener** or **Oracle Connection Manager**, are configured with identical protocol addresses. The client uses this address to send the connection request to a particular network object location, and the recipient "listens" for requests on this address. It is important to install the same protocols for the client and the connection recipient, and to configure the same addresses.

protocol conversion

A feature of Oracle Connection Manager that enables a client and server with different networking protocols to communicate with each other.

protocol stack

Designates a particular **presentation layer** and **session layer** combination.

proxy server

A server that substitutes for the real server, forwarding client connection requests to the real server or to other proxy servers. Proxy servers provide access control, data and system security, monitoring, and caching.

public database link

A database link created by a DBA on a local database that is accessible to all users on that database.

See also **database link** and **private database link**.

realm Oracle Context

An Oracle Context contained in each **identity management realm**. It stores the following information:

- User naming policy of the identity management realm—that is, how users are named and located
- Mandatory authentication attributes
- Location of groups in the identity management realm
- Privilege assignments for the identity management realm—for example: who has privileges to add more users to the realm.
- Application specific data for that realm including authorizations

RDBMS

Relational Database Management System

RDN

See **relative distinguished name (RDN)**.

relative distinguished name (RDN)

The local, most granular level entry name. It has no other qualifying entry names that would serve to address the entry uniquely. In the example, `cn=sales,dc=us,dc=example,dc=com`, `cn=sales` is the RDN.

root Oracle Context

In the **Oracle Identity Management** infrastructure, the root Oracle Context is an entry in Oracle Net Services containing a pointer to the default **identity management realm** in the infrastructure. It also contains information about how to locate an identity management realm given a simple name of the realm.

RPC

Remote Procedure Call

SDP

Sockets Direct Protocol (SDP).

Secure Sockets Layer (SSL)

An industry standard protocol designed by Netscape Communications Corporation for securing network connections. SSL provides authentication, encryption, and data integrity using public key infrastructure (PKI).

server parameter file

A binary file containing initialization parameter settings that is maintained on the Oracle Database host. You cannot manually edit this file with a text editor. A server parameter file is initially built from a text initialization parameter file by means of the `CREATE SPFILE` statement or created directly with the Database Configuration Assistant.

server process

Database processes that handle a client request on behalf of a database.

service

Work done for others. The database is a service that stores and retrieves data for clients.

service handler

A process that acts as a connection point from the listener to the database server. A service handler can be a **dispatcher** or **dedicated server**.

service name

A logical representation of a database, which is the way a database is presented to clients. A database can be presented as multiple services and a service can be implemented as multiple database instances. The service name is a string that is the **global database name**, that is, a name comprised of the database name and domain name, entered during installation or database creation. If you are not sure what the global database name is, then you can obtain it from the value of the `SERVICE_NAMES` parameter in the initialization parameter file.

The service name is included in the **connect data** part of the **connect descriptor**.

service registration

A feature by which the **PMON process** automatically registers information with a **listener**. Because this information is registered with the listener, the `listener.ora` file does not need to be configured with this static information.

Service registration provides the listener with information about:

- Service names for each running instance of the database

- Instance names of the database
- Service handlers (**dispatcher** or **dedicated server**) available for each instance

  These enable the listener to direct a client request appropriately.
- Dispatcher, instance, and node load information

This load information enables the listener to determine which dispatcher can best handle a client connection request. If all dispatchers are blocked, then the listener can spawn a dedicated server for the connection.

session data unit (SDU)

A buffer that Oracle Net uses to place data before transmitting it across the network. Oracle Net sends the data in the buffer either when requested or when it is full.

session layer

A network layer that provides the services needed by the **protocol address** entities that enable them to organize and synchronize their dialogue and manage their data exchange. This layer establishes, manages, and terminates network sessions between the client and server. An example of a session layer is **Network Session (NS)**.

session multiplexing

Combining multiple sessions for transmission over a single network connection to conserve the operating system's resources.

shared server

A database server that is configured to allow many user processes to share very few server processes, so the number of users that can be supported is increased. With shared server configuration, many user processes connect to a **dispatcher**. The dispatcher directs multiple incoming network session requests to a common queue. An idle shared server process from a shared pool of server processes picks up a request from the queue. Thus, a small pool of server processes can serve a large number of clients. Contrast with **dedicated server**.

shared server process

A process type used with **shared server** configuration.

SID

See **Oracle system identifier (SID)**.

SID_LIST_*listener_name*

A section of the `listener.ora` file that defines the **Oracle system identifier (SID)** of the database served by the listener. This configuration is not required for an Oracle database because information for the instance is automatically registered with the listener. However, static configuration is required for other services, such as **external procedure** calls and **Heterogeneous Services**.

single sign-on

The ability of a user to log in to different servers using a single password. This permits the user to authenticate to all servers the user is authorized to access.

sqlnet.ora file

A configuration file for the client or server that specifies:

- Client domain to append to unqualified service names or net service names

- Order of naming methods the client should use when resolving a name
- Logging and tracing features to use
- Route of connections
- External naming parameters
- Oracle Advanced Security parameters

The `sqlnet.ora` file typically resides in the `ORACLE_HOME/network/admin` directory.

SSL

See **Secure Sockets Layer (SSL)**.

System Global Area (SGA)

A group of shared memory structures that contain data and control information for an Oracle **instance**.

TCP/IP

Transmission Control Protocol/Internet Protocol. The standard communication protocol used for client/server conversation over a network.

TCP/IP with SSL protocol

A protocol that enables an Oracle application on a client to communicate with remote Oracle databases through the **TCP/IP** and **Secure Sockets Layer (SSL)**.

tick

The amount of time it takes for a message to be sent and processed from the client to the server or from the server to the client

TNS

See **Transparent Network Substrate (TNS)**.

tnsnames.ora file

A configuration file that contains maps **net service name**s to **connect descriptor**s. This file is used for the **local naming** method. The `tnsnames.ora` file typically resides in the `ORACLE_HOME/network/admin` directory.

tracing

A utility that writes detailed information about an operation to an output file. The trace utility produces a detailed sequence of statements that describe the events of an operation as they are run. Administrators use the trace utility for diagnosing an abnormal condition; it is not normally turned on.

See also **logging**.

Transparent Application Failover (TAF)

A run-time failover for high-availability environments, such as Oracle Real Application Clusters and Oracle Fail Safe, that refers to the failover and re-establishment of application-to-service connections. It enables client applications to automatically reconnect to the database if the connection fails, and, optionally, resume a `SELECT` statement that was in progress. This reconnect happens automatically from within the Oracle Call Interface (OCI) library.

Transparent Network Substrate (TNS)

A foundation technology, built into the **Oracle Net foundation layer** that works with any standard network transport protocol.

transport

A networking layer that maintains end-to-end reliability through data flow control and error recovery methods. The **Oracle Net foundation layer** uses **Oracle protocol support** for the transport layer.

TTC

See **Two-Task Common (TTC)**.

Two-Task Common (TTC)

A **presentation layer** type that is used in a typical Oracle Net connection to provide character set and data type conversion between different character sets or formats on the client and server.

UPI

User Program Interface

virtual circuit

A piece of shared memory used by the **dispatcher** for client database connection requests and replies. The dispatcher places a virtual circuit on a common queue when a request arrives. An idle shared server picks up the virtual circuit from the common queue, services the request, and relinquishes the virtual circuit before attempting to retrieve another virtual circuit from the common queue.

WebDAV protocol

World Wide Web Distributed Authoring and Versioning. A protocol with a set of extensions to the **HTTP protocol** which allows users to manage files on remote Web servers.

Index

Numerics

A

B

C

D

E

F

G

H

I

J

L

M

N

O

P

Q

R

S

T

U

V

W

Y