Oracle Net8™

Oracle Net8 Administrator's Guide

Release 8.0

December, 1997

Part No. A58230-01

ORACLE®

Enabling the Information Age™

Oracle Net8 Administrator's Guide

Part No. A58230-01

Release 8.0

Primary Author: — Rick Wong

Contributors: — Omar Bellal, Toby Close, Gil Cohen, Harvey Enamen, Shuvayu Kanjilal, Nancy Kramer, Michael Ledesma, Tong Ming Lee, Ethan Malasky, Sheryl Maring, Scot McKinley, Sergio Mendiola, Michael Mesaros, Ed Miner, Andrew Scott, Cyril Scott, P.V. Shivkumar, Sandra Venning, Wynn White, Norman Woo.

Contents

2 Understanding Net8

3 Planning Your Network

4 Configuring Network Services

5 Configuring Network Clients

6 Oracle Names

7 Oracle Connection Manager

8 Using Net8

9 Migrating to Net8

10 Troubleshooting Net8

B Configuration Parameters

C Sample Configuration Files

D Native Naming Adapters

Figures

Tables

Preface

The *Oracle Net8 Administrator's Guide* provides the information you need to understand and use the Net8 Release 8.0 product and its related applications.

Intended Audience

The information in this manual is intended primarily for network or database administrators (DBAs) responsible for Net8. This guide is also provided for anyone who wants to understand how Net8 works.

Feature Coverage and Availability

The *Oracle Net8 Administrator's Guide* contains information that describes the features and functionality of the Oracle8 and the Oracle8 Enterprise Edition products. Oracle8 and the Oracle8 Enterprise Edition have the same basic features. However, several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to use Oracle Connection Manager, you must have the Enterprise Edition.

For information about the differences between Oracle8 and the Oracle8 Enterprise Edition, and the features and options that are available to you, please refer to *Getting to Know Oracle8 and the Oracle8 Enterprise Edition*.

Structure

This manual contains ten chapters and four appendices:

Chapter 1 Introduces the Net8 release 8.0.4 product, and provides an overview of its main applications, features, and functionality.

Chapter 2	Describes the Transparent Network Substrate (TNS) and its role in distributed systems. It also explains how Net8 works with the TNS architecture to perform basic connectivity and transport operations.
Chapter 3	Describes considerations for planning a network using Net8. It explains the relationships of the network products, and options for better managing your future network.
Chapter 4	Describes and outlines procedures to configure Net8 services, including the network listener.
Chapter 5	Describes and outlines procedures to configure Net8 client components. This includes an explanation of the files and parameters required by Net8, and a review of the Oracle Net8 Assistant, a tool that is provided to automate client configuration using a graphical user interface.
Chapter 6	Describes features and functionality of Oracle Names. This includes procedures to configure and use Names Servers.
Chapter 7	Describes features and functionality of Oracle Connection Manager. Outlines procedures to configure and use Connection Managers.
Chapter 8	Describes issues associated with migrating from SQL*Net version 2. Provides scenarios detailing considerations for migrating to Net8 using Oracle Names.
Chapter 9	Outlines procedures to use Net8 once it has been configured. This includes information on starting and testing the network.
Chapter 10	Describes procedures to troubleshoot Net8. This includes information on tracing and logging.
Chapter 11	Describes Net8 enhancements for programmers. This includes a review of Net8 OPEN, UNIX signal handling, and bequeath adapter.
Chapter 12	Describes related Oracle products that extend Net8 functionality. This includes Oracle Enterprise Manager, Oracle Advanced Networking Option, and Oracle Security Server.
Appendix A	Describes all commands for Net8 Control Utilities including Listener Control (LSNRCTL), Oracle Names Control (NAMESCTL), and Connection Manager Control (CMCTL).

Appendix B	Lists and describes configuration parameters for profiles, local names, listener, Names Servers, Connection Manager, protocols, and database initialization.
Appendix C	Provides sample configuration files for profiles (SQLNET.ORA), local names (TNSNAMES.ORA), listener (LISTENER.ORA), Names Servers (NAMES.ORA, and Connection Manager (CMAN.ORA).
Appendix D	Describes Native Naming Adapters.

Related Documents

The *Oracle Net8 Administrator's Guide* replaces information previously documented in the following manuals:

- *Understanding SQL*Net 2.3.3*
- *Oracle Names Administrator's Guide*
- *Oracle MultiProtocol Interchange Administrator's Guide*
- *Oracle Network Manager Administrator's Guide*
- *Oracle Network Products Troubleshooting Guide*

Networking error messages formally documented in the *Oracle Network Products Troubleshooting Guide* are now available on a CD-ROM containing all Oracle8 error messages.

For more information, refer to the following manuals:

- *Oracle8 Distributed Database Systems*
- *Oracle8 Tuning*
- *Oracle Security Server Guide*
- *Oracle Cryptographic Toolkit Programmer's Guide*
- *Oracle Enterprise Manager Administrator's Guide*
- *Oracle Advanced Networking Option Administrator's Guide*

Conventions

The following conventions are also used in this manual:

Convention	Meaning
. . .	Horizontal ellipsis points in statements or commands mean that parts of the statement or command not directly related to the example have been omitted.
boldface text	Boldface type in text indicates a term defined in the text, the glossary, or in both locations.
UPPERCASE	Uppercase type identifies file names, command names, directory names, and function arguments.
italics	Identifies a variable in command or function call syntax; replace this variable with a specific value or string. It also identifies book titles and the first use of a technical term.
< >	Angle brackets enclose user-supplied names.
[ ]	Brackets enclose optional clauses from which you can choose one or none.

Send Us Your Comments

Oracle Net8 Administrator's Guide

Release 8.0

Part No. A58230-01

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

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

If you find any errors or have any other suggestions for improvement, please indicate the chapter, section, and page number (if available).

You can send comments to us in the following ways:

- electronic mail - infodev@us.oracle.com
- FAX - (650)506-7200. Attn: Server Technologies Documentation Manager
- postal service

  Oracle Corporation
  Attn: Server Technologies Documentation Manager
  400 Oracle Parkway
  Redwood Shores, CA 94065 USA

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

1

Introducing Net8

Net8 is the follow-on networking product to SQL*Net version 2. Its primary purpose is to enable the underlying network connectivity between remote client applications and Oracle8 and Oracle7 servers.

This chapter introduces Net8 Release 8.0, and provides an overview of its main applications, features, and functionality. It contains the following sections:

1.1 What Net8 Does

Net8 enables the machines in your network to “talk” with one another. It facilitates and manages communication sessions between a client application and a remote database. Specifically, Net8 performs three basic operations:

- Connection - opening and closing connections between a client (or a server acting as a client) and a database server over a network protocol.
- Data Transport - packaging and sending data such as SQL statements and data responses so that it can be transmitted and understood between a client and a server.
- Exception Handling - initiating interrupt requests from the client or server.

For more information on these operations, refer to Chapter 2, “Understanding Net8”.

1.1.1 What Happened to SQL*Net?

Net8 replaces SQL*Net as the networking services and connectivity component for Oracle8. The new name reflects enhanced capabilities of the product that now go beyond simply sending SQL statements across a network. Net8 supports application programming interfaces, Java-enabled Internet browsers, and network services such as naming and security.

Following Net8 Release 8.0, all future networking products will synchronize version numbers with those of the Oracle server to eliminate any confusion regarding product compatibility.

1.2 Net8 Applications

Net8 is part of the standard Oracle networking package. This package also includes:

- Oracle Names
- Oracle Connection Manager (available with Oracle8 Enterprise Edition)
- Net8 OPEN

For more information on Oracle Names, refer to Chapter 6, "Oracle Names".

For more information on the Oracle Connection Manager, refer to Chapter 7, "Oracle Connection Manager".

For more information about Net8 OPEN, refer to Chapter 11, "Net8 Enhancements for Programmers".

1.3 Why Use Net8?

Net8 provides the following benefits to users of networked applications:

- Network Transparency
- Protocol Independence
- Media / Topology Independence
- Heterogeneous Networking
- Large Scale Scalability

1.3.1 Network Transparency

Net8 provides support for a broad range of network transport protocols including TCP / IP, SPX / IPX, IBM LU6.2, Novell, and DECnet. It does so in a manner that is invisible to the application user. This enables Net8 to interoperate across different types of computers, operating systems, and networks to transparently connect any combination of PC, UNIX, legacy, and other system without changes to the existing infrastructure.

1.3.2 Protocol Independence

Net8 enables Oracle applications to run over any supported network protocol by using the appropriate Oracle Protocol Adapter. Applications can be moved to another protocol stack by installing the necessary Oracle Protocol Adapter and the industry protocol stack. Oracle Protocol Adapters provide Net8 access to connections over specific protocols or networks. On some platforms, a single Oracle Protocol Adapter will operate on several different network interface boards, allowing you to deploy applications in any networking environment.

1.3.3 Media/Topology Independence

When Net8 passes control of a connection to the underlying protocol, it inherits all media and / or topologies supported by that network protocol stack. This allows the network protocol to use any means of data transmission, such as Ethernet, Token Ring, or other, to accomplish low level data link transmissions between two machines.

1.3.4 Heterogeneous Networking

Oracle's client-server and server-server models provide connectivity between multiple network protocols using Oracle Connection Manager.

1.3.5 Large Scale Scalability

By enabling you to use advanced connection concentration and connection pooling features, Net8 makes it possible for thousands of concurrent users to connect to a server.

1.4 Net8 Features

Net8 Release 8.0 features several enhancements that extend scalability, manageability and security for the Oracle network.

1.4.1 Scalability Features

Scalability refers to the ability to support simultaneous network access by a large number of clients to a single server. With Net8, this is accomplished by optimizing the usage of network resources by reducing the number of physical network connections a server must maintain.

Net8 offers improved scalability through two new features:

- Connection pooling
- Connection concentration

Both of these features optimize usage of server network resources to eliminate data access bottlenecks and enable large numbers of concurrent clients to access a single server. Additionally, other enhancements such as a new buffering methods and asynchronous operations further improve Net8 performance. For more information on Net8's scalability features, refer to Section 3.4, "Improving Large Network Performance".

Connection pooling is implemented as a configuration option with the Multi-Threaded Server. For more information on connection pooling, refer to Section 3.4.2, "Connection Pooling".

1.4.2 Manageability Features

Net8 introduces a number of new features that will simplify configuration and administration of the Oracle network for both workgroup and enterprise environments.

For workgroup environments, Net8 offers simple configuration-free connectivity through installation defaults and a new name resolution feature called host naming. For enterprise environments, Net8 centralizes client administration and simplifies network management with Oracle Names. In addition to these new features, Net8 introduces the Oracle Net8 Assistant.

1.4.2.1 Host Naming

Host Naming refers to a new naming method which resolves service names to network addresses by enlisting the services of existing TCP/IP hostname resolution systems. Host Naming can eliminate the need for a local naming configuration file (TNSNAMES.ORA) in environments where simple database connectivity is desired. For more information on host naming, refer to Section 3.3.2, "Host Naming".

1.4.2.2 Oracle Net8 Assistant

The Oracle Net8 Assistant is a new end user, stand-alone Java application that can be launched either as a stand-alone application or from the Oracle Enterprise Manager console. It automates client configuration and provides an easy-to-use interface as well as wizards to configure and manage Net8 networks. For more information on the Oracle Net8 Assistant, refer to Section 5.1.3, "Oracle Net8 Assistant".

Because the Oracle Net8 Assistant is implemented in Java, it is available on any platform that supports the Java Virtual Machine.

1.4.2.2.1 What happened to Oracle Network Manager? Oracle Network Manager is not needed or supported with Net8. The tasks previously accomplished by Oracle Network Manager are either performed automatically by Net8 and its components or can be executed more simply by using the Oracle Net8 Assistant. Oracle Network Manager is, however, still required and supported for SQL*Net V2 environments.

1.4.3 Security Features

Three complementary products extend and enhance security using Net8:

- Oracle Security Server, a new X.509 certificate-based server that provides scalable, standards-based security services and support for electronic commerce
- Oracle Advanced Networking Option, which supports not only network data encryption and checksumming, but also biometric user authentication devices such as the IdentixTouchSAFE II finger print scanner

- Oracle Enterprise Manager, which provides the interface to the Oracle Security Server certificate authority enabling administrators to maintain uniform user identities throughout the enterprise via unique identities and roles. This allows administrators to grant and revoke user privileges uniformly and at once, allowing for easier and more efficient administration, as well as improved security gained by faster removal of terminated users privileges.

For more information about Net8's related Oracle products, refer to Chapter 12, "Extending Net8 Functionality".

1.4.4 Other Features

Other features and changes notable in Net8 Release 8.0 are as follows:

1.4.4.1 Multiprotocol Support Using Oracle Connection Manager

Oracle Connection Manager provides the capability to seamlessly connect two or more network protocol communities, enabling transparent Net8 access across multiple protocols. In this sense, it replaces the functionality provided by the Oracle MultiProtocol Interchange with SQL*Net. Oracle Connection Manager can also be used to provide network access control. For example, links processed through Oracle Connection Manager can be filtered on the basis of origin, destination, or user ID. It incorporates a Net8 application proxy for implementing firewall-like functionality.

For more information on Oracle Connection Manager, refer to Chapter 7, "Oracle Connection Manager".

1.4.4.2 Oracle Trace Assistant

Net8 includes the Oracle Trace Assistant, a tool introduced with SQL*Net V2.3 to help decode and analyze the data stored in Net8 trace files. The Oracle Trace Assistant (formerly called Trace Evaluator) provides an easy way to understand and take advantage of the information stored in trace files, it is useful for diagnosing network problems and analyzing network performance. It can be used to better pinpoint the source of a network problem or identify a potential performance bottleneck.

For more information on the Oracle Trace Assistant, refer to Section 10.4.3, "Using the Trace Assistant to Examine Your Trace Files".

1.4.4.3 Native Naming Adapters

Native Naming Adapters, previously bundled with the Advanced Networking Option, are now included with Net8. These adapters provide native support for industry-standard name services, including Sun NIS/Yellow Pages and Novell NetWare Directory Services (NDS).

2

Understanding Net8

Net8 uses the Transparent Network Substrate (TNS) and industry-standard networking protocols to connect a client to a server and establish an Oracle session.

This chapter describes TNS and the role it plays in distributed systems. It also explains how Net8 interacts with TNS to perform basic connectivity and transport operations. This chapter contains the following sections:

2.1 Net8 Operations

Net8 is responsible for enabling communications between the cooperating partners in an Oracle distributed transaction, whether they be client-server or server-server. Specifically, Net8 provides three basic networking operations:

- Connect Operations
- Data Operations
- Exception Operations

2.2 Connect Operations

Net8 supports two types of connect operations:

- Connecting to Servers (open)
- Disconnecting from Servers (close)

2.2.1 Connecting to Servers

Users initiate a connect request by passing information such as a username and password along with a short name for the database service that they wish to connect. That short name, called a *service name*, is mapped to a network address contained in a *connect descriptor*. Depending upon your specific network configuration, this connect descriptor may be stored in one of the following:

- a local names configuration file called TNSNAMES.ORA
- a Names Server for use by Oracle Names
- a native naming service such as NIS or DCE CDS

Note: If the network includes Oracle Names, you do not need a local names (TNSNAMES.ORA) configuration file. The service names and associated connect descriptors are stored in a Names Server. Similarly, if an Oracle Native Naming Adapter such as NIS or DCE CDS is being used, this information is stored and retrieved from that native name service.

Net8 coordinates its sessions with the help of a network listener.

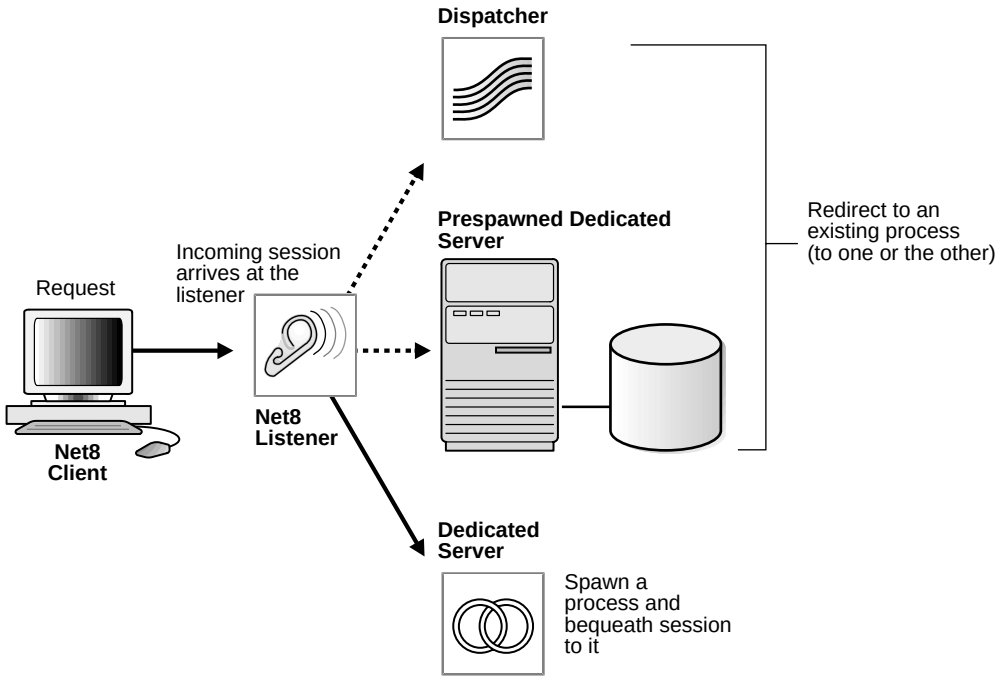
2.2.2 Establishing Connections with the Network Listener

The network listener is a single process or task setup specifically to receive connection requests on behalf of an application. Listeners are configured to "listen on" an address specified in a listener configuration file (called LISTENER.ORA) for a database or non-database service. Once started, the listener will receive client connect requests on behalf of a service, and respond in one of three ways:

- Bequeath the session to a new dedicated server process
- Redirect to an existing server process (such as a dispatcher or prestarted dedicated server process)
- Refuse the session

Figure 2–1 depicts the role of the network listener in a typical Net8 connection to a server.

Figure 2–1 Network Listener In a Typical Net8 Connection



2.2.2.1 Bequeathed Sessions to Dedicated Server Processes

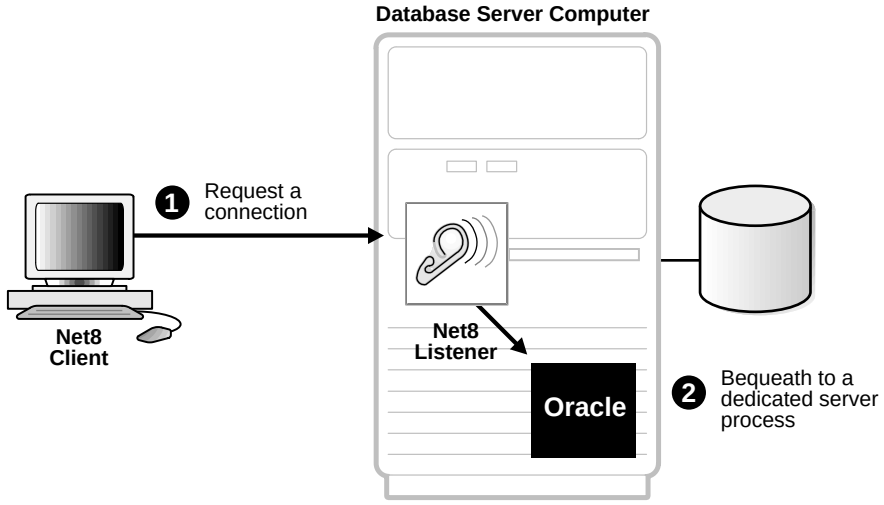
If the listener and server exist on the same node, the listener may create or 'spawn' dedicated server processes as connect requests are received. Dedicated server processes are committed to one session only and exist for the duration of that session. The sequence of events that occur when the listener creates a dedicated server process and passes or 'bequeaths' control of a session to it is as follows:

1. The listener is started and listens on an address specified in a listener configuration file (LISTENER.ORA).
2. A client connects to the listener with the network address.
3. The listener receives the session request, and determines if the client's request may be serviced. If not, the listener refuses the session and then resumes at Step 5.
4. The listener spawns a new dedicated server process to serve the incoming session, and bequeaths the session to that server process. Once the session is established, data flows directly between the client and dedicated server process.
5. The listener continues listening for incoming sessions.

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

Figure 2–2 depicts the role of the network listener in a bequeathed connection to a dedicated server process.

Figure 2–2 Bequeathed Connection To a Dedicated Server Process



2.2.2.2 Redirected Sessions to Existing Server Processes

Alternatively, Net8 may redirect the request to an existing server process. It does this by sending the address of an existing server process back to the client. The client will then resend its connect request to the server address provided.

Existing server processes include:

- Prestarted or Prespawned Dedicated Server Processes by the listener
- Dispatcher Processes created outside the listener process

2.2.2.2.1 Prespawned Dedicated Server Processes Net8 provides the option of automatically creating dedicated server processes **before** the request is received. These processes last for the life of the listener, and can be reused by subsequent connection requests. The use of prespawned dedicated server processes requires specification in a listener configuration file.

Note: Prespawned dedicated servers require SQL*Net release 2.1 or later, and Oracle Server release 7.1 or later.

The sequence of events that occurs when using prespawned dedicated server processes to service client connection requests is as follows:

1. The listener is started and listens on an address specified in a listener configuration file.
2. The listener then spawns a series of dedicated server processes until it reaches the specified pool size for each Oracle System Identifier (SID) defined in its configuration file.
3. Each spawned server process performs a partial address listen and provides the listener with the partial address that it is listening on. The listener initially marks all prespawned servers as idle.

   Note: A partial address listen is where the server process listens, but informs the underlying protocol stack that it has no preference as to the specific address it will listen on. As a result, many protocol stacks will choose a free listening address and automatically assign this to the requesting server process.

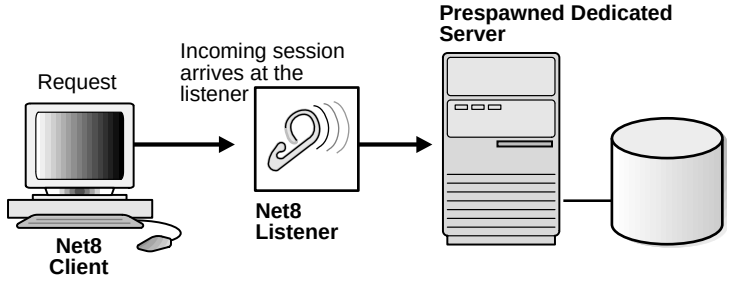
4. The client sends a connect request to the listener.
5. The listener receives the session request, and determines if the client's request may be serviced. If not, the listener refuses the session and then resumes at Step 9.
6. The listener issues a redirect message to the client containing one of the network addresses of the prespawned servers. The listener logs that server as active.
7. The client dissolves the session to the listener and establishes a session to the prespawned server using the address provided in the redirect message.
8. The listener spawns another server process to replace the active prespawned server (provided a value called PRESPAWN_MAX in the listener configuration file is greater than the number of prespawned server processes active and idle).
9. The listener continues listening for incoming sessions.

The above sequence of events continues until the maximum prespawn limit is reached, at which point the listener stops spawning new dedicated server processes.

When clients disconnect, the prespawned dedicated server process associated with the client returns to the idle pool. It then waits a specified length of time to be assigned to another client. If no client is handed to the prespawned server before the timeout expires, the prespawned server shuts down.

Figure 2–3 depicts the role of the network listener in a redirected connection to a prespawned dedicated server process.

Figure 2–3 Redirected Connection To a Prespawned Dedicated Server Process



2.2.2.2.2 Dispatcher Server Processes A dispatcher server process enables many clients to connect to the same server without the need for a dedicated server process for each client. It does this with the help of a *dispatcher* which handles and directs multiple incoming session requests to the shared server.

When an Oracle server has been configured as a multi-threaded server, incoming sessions are always routed to the dispatcher unless either the session specifically requests a dedicated server or no dispatchers are available. The sequence of events that occurs with the dispatcher server is as follows:

1. The listener is started and listens on either a default address or the addresses specified in its configuration file.
2. A database instance starts. Dispatchers start according to the configuration parameters in the initialization parameter file. Each dispatcher then performs a listen on the address assigned to it.
3. Each dispatcher's address is registered. When the listener is not listening on its default address, the listener's network name may be specified in the database initialization file (INIT.ORA). The name may resolve to more than one such address if multiple listeners are used.

Once the dispatcher addresses are registered, the listener can redirect incoming connect requests to them.

– If step 2 is performed before step 1, the dispatchers will not be able to contact the listener in step 3. If this occurs, there may be a delay as the dispatcher attempts to connect to the listener. If a connect request comes in a timeframe where no dispatchers are registered, these requests may be handled through prespawned dedicated or newly spawned dedicated server processes or be rejected.

The listener and the Oracle dispatcher server are now ready to receive incoming sessions.

Note: You can check which dispatchers have registered with the listener by issuing a SERVICES command in the Listener Control Utility. For more information, refer to SERVICES in the LSNRCTL Commands in Appendix A, "Control Utility Reference".

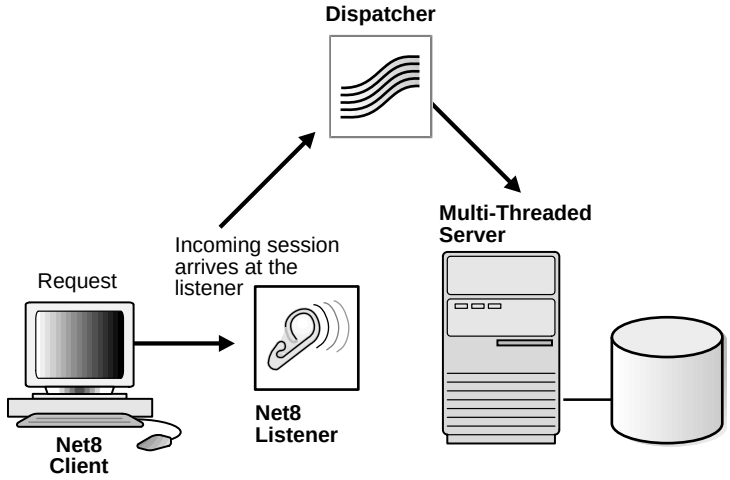
Once the listener and the dispatcher server have been started, the session activity continues as follows:

1. The client connects to the listener with the network address.
2. The listener receives the connect request, and determines if the client's request may be serviced. If not, the listener refuses the session and then resumes at Step 6.
3. The listener issues a redirect message to the client containing the network address of the least-used dispatcher for the shared server.
4. The client dissolves the session to the listener and establishes a session to the shared server using the network address provided in the redirect message.
5. The dispatcher updates the listener with the new load value because of the presence of the new session. This allows the listener to balance the incoming session requests between dispatchers running on the same protocol.
6. The listener resumes listening for incoming sessions.

When clients disconnect, the shared server associated with the client stays active and processes other incoming requests.

Figure 2–4 depicts the role of the network listener in a redirected connection to a dispatcher server process.

Figure 2–4 Redirected Connection To a Dispatcher Server Process



2.2.2.3 Refused Sessions

The network listener will refuse a session in the event that it does not know about the server being requested, or if the server is unavailable. It refuses the session by generating and sending a refuse response packet back to the client.

2.2.3 Disconnecting from Servers

Requests to disconnect from the server can be initiated in the following ways:

- user-initiated disconnect
- additional connection request
- abnormal connection termination
- timer-initiated disconnect

2.2.3.1 User-Initiated Disconnect

A user can request a disconnection from the server when a client-server transaction completes. A server can also disconnect from a second server when all server-server data transfers have been completed, and no need for the link remains.

2.2.3.2 Additional Connection Request

If a client application is connected to a server and requires access to another user account on the same or other server, most Oracle tools will first disconnect the application from the server to which it is currently connected. Once the disconnection is completed, a connection request to the new user account on the appropriate server is initiated.

2.2.3.3 Abnormal Connection Termination

Other components will occasionally disconnect or abort communications without giving notice to Net8. In this event, Net8 will recognize the failure during its next data operation, and clean up client and server operations, effectively disconnecting the current operation.

2.2.3.4 Timer Initiated Disconnect or Dead Connection Detection

Dead connection detection is a feature that allows Net8 to identify connections that have been left hanging by the abnormal termination of a client. On a connection with dead connection detection enabled, a small probe packet is sent from server to client at a user-defined interval (usually several minutes). If the connection is invalid (usually due to the client process or machine being unreachable), the connection will be closed when an error is generated by the send operation, and the server process will terminate the connection.

This feature minimizes the waste of resources by connections that are no longer valid. It also automatically forces a database rollback of uncommitted transactions and locks held by the user of the broken connection.

2.3 Data Operations

Net8 supports four sets of client-server data operations:

- send data synchronously
- receive data synchronously
- send data asynchronously
- receive data asynchronously

On the client side, a SQL dialogue request is forwarded using a send request in Net8. On the server side, Net8 processes a receive request and passes the data to the database. The opposite occurs in the return trip from the server.

Basic send and receive requests are synchronous. When a client initiates a request, it waits for the server to respond with the answer. It can then issue an additional request.

Net8 adds the capability to send and receive data requests asynchronously. This capability was added to support the Oracle shared server, also called a multi-threaded server, which requires asynchronous calls to service incoming requests from multiple clients.

2.4 Exception Operations

Net8 supports three types of exception operations:

- initiate a break over the connection
- reset a connection for synchronization after a break
- test the condition of the connection for incoming break

The user controls only one of these three operations, that is, the initiation of a break. When the user presses the Interrupt key ([Ctrl-C] on some machines), the application calls this function. Additionally, the database can initiate a break to the client if an abnormal operation occurs, such as during an attempt to load a row of invalid data using SQL*Loader.

The other two exception operations are internal to products that use Net8 to resolve network timing issues. Net8 can initiate a test of the communication channel, for example, to see if new data has arrived. The reset function is used to resolve abnormal states, such as getting the connection back in synchronization after a break operation has occurred.

2.5 Net8 and the Transparent Network Substrate (TNS)

Net8 uses the Transparent Network Substrate (TNS) and industry-standard networking protocols to accomplish its basic functionality. TNS is a foundation technology that is built into Net8 providing a single, common interface to all industry-standard protocols.

With TNS, peer-to-peer application connectivity is possible where no direct machine-level connectivity exists. In a peer-to-peer architecture, two or more computers (called *nodes* when they are employed in a networking environment) can communicate with each other directly, without the need for any intermediary devices. In a peer-to-peer system, a node can be both a client and a server.

A review of how Oracle clients and servers operate and communicate with each other will help you to understand what TNS is and how it works with Net8 to establish Oracle sessions.

2.6 Net8 Architecture

Oracle networking environments are based on two concepts:

- Distributed Processing
- Stack Communications

Distributed Processing

Oracle databases and client applications operate in what is known as a distributed processing environment. Distributed or cooperative processing involves interaction between two or more computers to complete a single data transaction. Applications such as an Oracle tool act as clients requesting data to accomplish a specific operation. Database servers store and provide the data.

In a typical network configuration, clients and servers may exist as separate logical entities on separate physical machines. This configuration allows for a division of labor where resources are allocated efficiently between a client workstation and the server machine. Clients normally reside on desktop computers with just enough memory to execute user friendly applications, while a server has more memory, disk storage, and processing power to execute and administer the database.

Distributed Databases

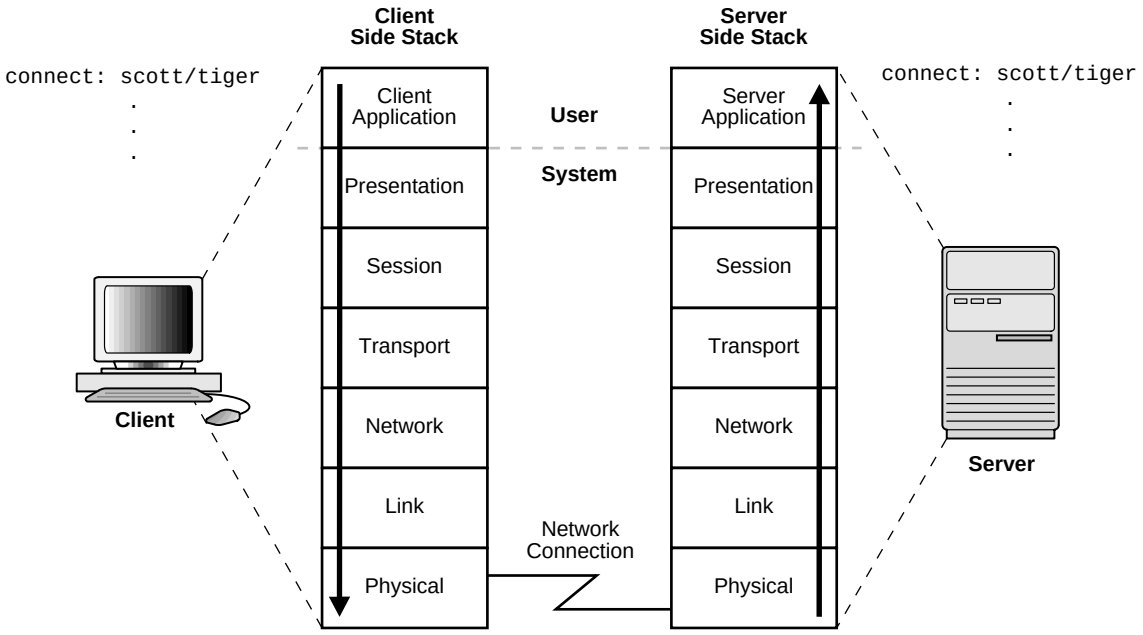
This type of client-server architecture also enables you to distribute databases across a network. A distributed database is a network of databases stored on multiple computers that appears to the user as a single logical database. Distributed database servers are connected by a database link, or path from one database to another. One server uses a database link to query and modify information on a second server as needed, thereby acting as a client to the second server.

2.6.1 Stack Communications

The concept of distributed processing relies on the ability of computers separated by both design and physical location to communicate and interact with each other. This is accomplished through a process known as stack communications.

Stack communications can be explained by referencing the Open System 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. Figure 2–5 depicts a typical OSI Protocol Communications Stack.

Figure 2–5 OSI Communications Stack



Information descends through layers on the client side where it is packaged for transport across a network medium in a manner that it can be translated and understood by corresponding layers on the server side.

A typical OSI protocol communications stack will contain seven such layers:

- Application - this is the OSI layer closest to the user, and as such is dependent on the functionality requested by the user. For example, in a database environment, a Forms application may attempt to initiate communication in order to access data from a server.

- Presentation - ensures that information sent by the application layer of one system is readable by the application layer of another system. This includes keeping track of syntax and semantics of the data transferred between the client and server. If necessary, the presentation layer translates between multiple data representation formats by using a common data format.
- Session - as its name suggests, establishes, manages, and terminates sessions between the client and server. This is a virtual pipe that carries data requests and responses. The session layer manages whether the data traffic can go in both directions at the same time (referred to as asynchronous), or in only one direction at a time (referred to as synchronous).
- Transport - implements the data transport ensuring that the data is transported reliably.
- Network - ensures that the data transport is routed through optimal paths through a series of interconnected subnetworks.
- Link - provides reliable transit of data across a physical link.
- Physical - defines the electrical, mechanical, and procedural specifications for activating, maintaining and deactivating the physical link between client and server.

2.6.2 Stack Communications in an Oracle networking environment

Stack communications allow Oracle clients and servers to share, modify, and manipulate data between themselves. The layers in a typical Oracle communications stack are similar to those of a standard OSI communications stack.

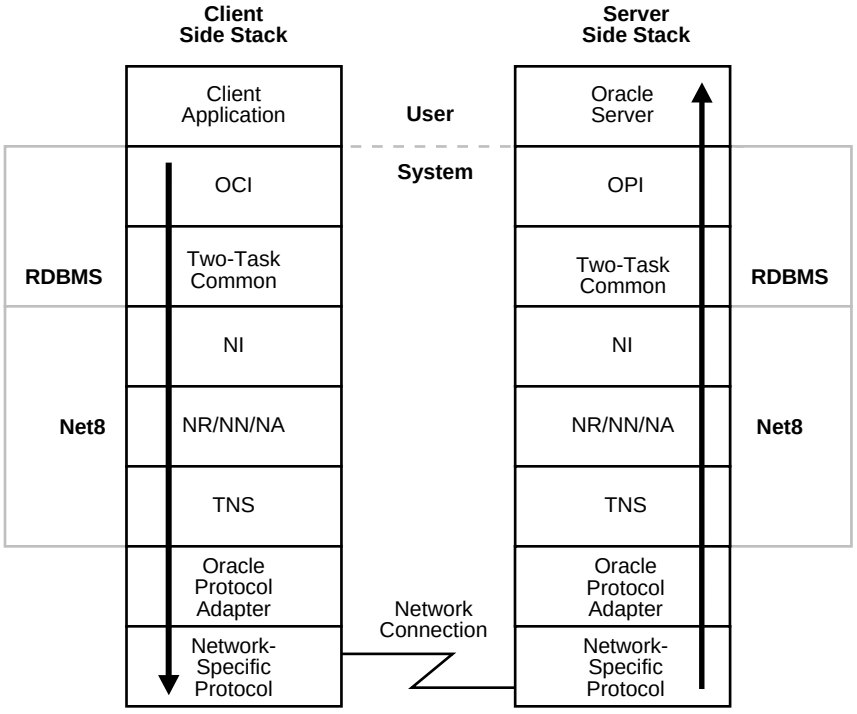
2.6.2.1 Client-Server Interaction

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

- Client Application
- Oracle Call Interface (OCI)
- Two Task Common
- Net8
- Oracle Protocol Adapters
- Network Specific Protocols

Figure 2–6 depicts a typical communications stack in an Oracle networking environment.

Figure 2–6 Typical Communications Stack in an Oracle environment



Client Application

Oracle client applications provide all user-oriented activities, such as character or graphical user display, screen control, data presentation, application flow, and other application specifics. The application identifies database operations to send to the server and passes them through to the Oracle Call Interface (OCI).

Oracle Call Interface (OCI)

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

- parse SQL statements for syntax validation
- open a cursor for the SQL statement
- bind client application variables into the server shared memory
- describe the contents of the fields being returned based on the values in the server's data dictionary
- execute SQL statements within the cursor memory space
- fetch one or more rows of data into the client application
- close the cursor

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

For more information about OCI, refer to the *Oracle Call Interface Programmer's Guide.*

Two-Task Common

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

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

Net8

Net8 provides all session layer functionality in an Oracle communications stack. It is responsible for establishing and maintaining the connection between a client application and server, as well as exchanging messages between them. Net8 itself has three component layers that facilitate session layer functionality:

- Network Interface (NI) - This layer provides a generic interface for Oracle clients, servers, or external processes to access Net8 functions. The NI handles the "break" and "reset" requests for a connection.

- Network Routing (NR)/ Network Naming (NN)/ Network Authentication (NA) - NR provides routing of the session to the destination. This may include any intermediary destinations or 'hops,' on the route to the server destination. NN resolves aliases to a Net8 destination address. NA negotiates any authentication requirement with the destination.
- Transparent Network Substrate (TNS) - TNS is an underlying layer of Net8 providing a common interface to industry standard protocols. TNS receives requests from Net8, and settles all generic machine-level connectivity issues, such as: the location of the server or destination (open**,** close functions); whether one or more protocols will be involved in the connection (open**,** close functions); and how to handle interrupts between client and server based on the capabilities of each (send, receive functions). The generic set of TNS functions (open, close, send, receive) passes control to an Oracle Protocol Adapter to make a protocol-specific call. Additionally, TNS supports encryption and sequenced cryptographic message digests to protect data in transit.

Oracle Protocol Adapters

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

Network-Specific Protocols

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

Server-Side Interaction

Information passed from a client application across a network protocol is received by a similar communications stack on the server side. The process stack on the server side is the reverse of what occurred on the client side with information ascending through communication layers. The one operation unique to the server side is the act of receiving the initial connection through the network listener.

The following components above the Net8 session layer are different from those on the client side:

- Oracle Program Interface (OPI)

- Oracle Server

Oracle Program Interface (OPI)

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

Oracle Server

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

2.6.3 Server-to-Server Interaction

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

3

Planning Your Network

Net8 provides a variety of options to help you design and manage networks that are both flexible and easy to use. With Net8's enhanced scalability and manageability features, you can develop a network to support a wide range of environments whether they be simple workgroups or large mission critical enterprises.

This chapter describes considerations for planning a network using Net8. It explains the relationships of the network products, and options for expanding and better managing your future network. It includes the following sections:

3.1 Planning Overview

Take the time to review and plan your network before you configure it. As you are planning your Oracle network, remember to keep future needs in mind as well as present requirements. Some of the more important decisions which you will need to make regarding your network include:

- Defining your network layout
- Resolving service names
- Managing connection requests
- Improving network performance

3.2 Defining Your Network Layout

The following checklist is provided to help you outline the main components of your network.

1. Define from the outset what it is you hope to accomplish with your network.
2. Research the functionality required by your client applications, then assess the resources that are available to meet those requirements.
3. Determine which machines or nodes are best suited for client or server applications.
4. Select a networking protocol which best suits your existing or future networking requirements. You may be able to choose a single transport level protocol that works well on all the components in your network. Protocol adapters are available for most of the major protocols on many platforms. Your network may also involve clients or servers operating over more than one protocol.
5. If you decide to use multiple protocols on your network, determine which nodes are best suited to install Oracle Connection Manager. Your choice of nodes will be determined by the networking protocols you have chosen as well as the machine's capacity to handle anticipated traffic.

It helps sometimes to draw a picture of your network layout displaying the logical as well as physical relationships between networking components.

3.3 Resolving Service Names

Once you have defined your network layout, you will need to decide how best to configure and manage your network implementation. One of the first and most important decisions that you will need to make is choosing a naming method.

3.3.1 Naming Methods

Naming refers to the method used by a client application to resolve a service name to a network address when attempting to connect to a database service. Net8 provides four naming methods:

- Host Naming
- Local Naming
- Centralized Naming using Oracle Names
- External Naming

Depending on the size and characteristics of your network, each method will have positive and negative implications for both how the network is configured and administered.

3.3.2 Host Naming

Host naming is a new feature enabling users in a TCP/IP environment to resolve Oracle service names using their existing name resolution service. This name resolution service might be DNS, NIS, or simply a centrally-maintained set of "/etc/hosts" files.

Host naming allows users to connect to an Oracle server simply by using the server computer's host name or host name alias. No client configuration is required to take advantage of this feature. The connection is established by using the default TCP/IP port for the listener, that is, port 1521. Multiple databases per node and database location transparency are supported through matching listener global database names with host name aliases.

Host Naming can eliminate the need for a local naming configuration file (TNSNAMES.ORA) in environments where simple database connectivity is desired. It is not however, suitable for large, complex environments where advanced features such as connection pooling, heterogeneous services or application failover (which require additional connect information) are desired.

3.3.2.1 Establishing a Connection Using the Host Naming Option

The process for establishing a client session using the host naming option is as follows:

1. The client initiates a connect request providing a service name which is also a TCP/IP hostname, or hostname alias.
2. A host naming adapter resolves this service name by generating a network address using the service name as both the TCP/IP hostname and the global database name. The TCP/IP port defaults to 1521.
3. Net8 makes the connect request to the address created.
4. A network listener, listening at registered TCP/IP port 1521, receives the request and establishes a connection to the database with a matching global database name as specified in the listener's configuration.
5. The connection is accepted by the server.

3.3.2.2 Host Naming Zero Configuration Scenario

The host naming option enables clients to connect to a remote server without configuration.

As the Oracle8 server is installed, a listener configuration file is generated automatically. The listener will be configured to support an initial database sharing the same name as the host, using the default TCP/IP listening address. Since host naming is used as a naming method by default, it will be used in the absence of a local naming configuration file and Names Server. Clients can make a connection to the initial database by using the host name as a service name.

Furthermore, clients can make connections to additional instances on the same host if they are aliased properly in an IP address translation mechanism such as DNS, NIS, or a centrally maintained TCP/IP hosts file.

3.3.2.2.1 Host Naming Limitations Automatic configuration of the listener however, depends upon whether it receives accurate information about the host and domain name for the machine the server is installed on. If you cannot make a connection to the initial database using Host Naming over a TCP/IP network, verify that the global database name in the listener configuration file specifies a complete name, including domain name. For more information on configuring a listener, refer to Section 4.2, “Configuring the Network Listener”.

3.3.3 Local Naming

Local naming refers to the method of resolving a service name to a network address by using information configured on each individual client. Much like an address book, this information is entered in a local naming configuration file called TNSNAMES.ORA.

3.3.3.1 Establishing a Connection Using the Local Naming Option

The process for establishing a client session using the local naming option is as follows:

1. The client initiates a connect request providing a service name.
2. The service name is resolved to a network address configured in a local naming file.
3. Net8 makes the connect request to the address provided.
4. A network listener receives the request and directs it to the database it is servicing.
5. The connection is accepted by the server.

3.3.3.2 Configuring Local Naming

To configure local naming, proceed as follows:

1. Verify that "**TNSNAMES**" is listed in the field of selected naming methods in the client profile. If it is not, use the Oracle Net8 Assistant to edit the profile.
2. Verify that service names are correctly mapped to their appropriate network addresses in a local naming configuration file (TNSNAMES.ORA). If they are not, you may use the Oracle Net8 Assistant to add or modify service names. For more information on configuring a local naming configuration file, refer to Section 5.4, "Configuring Service Names Using the Oracle Net8 Assistant".

For information on configuring clients to use the local naming option, refer to Section 5.2.1, "Configuring Naming Methods".

3.3.4 Centralized Naming using Oracle Names

Centralized Naming refers to the method of resolving a service name to a network address by using Oracle Names. Oracle Names uses Names Servers to store the names and addresses of all database services on a network. Much like people calling for directory assistance, clients wishing to connect to a server direct their connect requests to a Names Server. Names Servers resolve the service name to a network address and return that information to the client.

3.3.4.1 Establishing a Connection Using the Centralized Naming Option

The process for establishing a client session using the centralized naming option is as follows:

1. The client initiates a connect request providing a service name.
2. The connect request is forwarded to a Names Server where the service name is resolved to a network address. This address is returned to the client.
3. Net8 makes the connect request to the address provided.
4. A network listener receives the request and redirects it to the database it is servicing.
5. The connection is accepted by the server.

3.3.4.2 Configuring Centralized Naming

To configure centralized naming, proceed as follows:

1. Verify that "**ONAMES**" is listed in the field of selected naming methods in your profile. If it is not, use the Oracle Net8 Assistant to edit the profile.
2. Verify that a Names Server exists and is running on the network. If there are no Names Servers in your network, you may use the Oracle Net8 Assistant to start and configure a Names Server.

For information on configuring clients to use centralized naming using Oracle Names, refer to Section 5.2.1, "Configuring Naming Methods".

For more information on configuring clients to use Oracle Names, refer to Section 5.5, "Configuring Clients to Use Oracle Names".

For additional information on configuring Oracle Names, refer to Chapter 6, "Oracle Names".

3.3.5 External Naming

External Naming refers to the method of resolving a service name to a network address by using a supported non-Oracle naming service. Oracle Native Naming Adapters resolve service names stored in customers' native (non-Oracle) naming services. They include:

- Network Information Service (NIS)
- NetWare Directory Service (NDS)

Note: In previous releases of SQL*Net, these Native Naming Adapters were part of the Oracle Advanced Networking Option. They are now included as a standard part of Net8.

Distributed Computing Environment Cell Directory Service (CDS) continues to be available as part of the DCE Integration part of Oracle Advanced Networking Option.

3.3.5.1 Establishing a Connection Using the External Naming Option

The process for establishing a client session using the external naming option is as follows:

1. The client initiates a connect request providing a service name.
2. A native naming adapter forwards the request to a native naming system that resolves the service name to a network address. The address is returned to the client.
3. Net8 makes the connect request to the address provided.
4. A network listener receives the request and redirects it to the database it is servicing.
5. The connection is accepted by the server.

3.3.5.2 Configuring External Naming

To configure external naming using the Oracle Net8 Assistant, proceed as follows:

1. Verify that the applicable Native Naming Adapter has been installed on the client node.
2. Specify the use of an external naming adapter (for example, **CDS**, **NDS**, or **NIS**) in your profile. If it is not, use the Oracle Net8 Assistant to edit the client profile.

For more information on configuring clients to use the external naming option, refer to Section 5.2.1, “Configuring Naming Methods”.

3.3.5.3 Oracle Names and Native Naming Adapters

Oracle Names can be used in conjunction with other proprietary or open naming services to provide cross-environment name resolution. For example, Oracle Native Naming Adapters for CDS/DCE, NIS or NDS could be installed on all clients and servers in an enterprise network already running Oracle Names to provide name resolution across multiple name services.

Since Oracle Names is a proprietary name service storing and resolving names and addresses for Oracle databases only, one names solution could be to store all your Oracle services in Oracle Names, and use a directory service such as DNS or X.500 as your global naming service.

3.3.6 Choosing a Naming Method

Table 3–1 summarizes the relative advantages and disadvantages of each naming method and provides recommendations for using them in your network.

Table 3–1 Naming Method Comparison

Naming Method	Advantages/Disadvantages	Recommended for:
Host Naming	**1.** Requires minimal user configuration. The user may provide only the name of the host to establish a connection. **2.** Eliminates the need to create and maintain a local names configuration file (TNSNAMES.ORA). **3.** Eliminates the need to understand Oracle Names administration procedures. **Disadvantages**: Available only if all of the following are true: ■ Your client and server are connecting using TCP / IP. ■ The hostname is resolved through an IP address translation mechanism such as Domain Name Services (DNS), Network Information Services (NIS), or a centrally maintained TCP / IP hosts file. ■ No Oracle Connection Manager features are requested. ■ The global database name matches the name the name of the host machine; or the host name is aliased to the database services on the host.	Simple TCP / IP networks (with 10-20 databases) that meet the criteria listed.

Table 3–1 Naming Method Comparison

Naming Method	Advantages/Disadvantages	Recommended for:
Local Naming	**1.** Provides a relatively straightforward method for resolving service name addresses. **2.** Resolves service names across networks running different protocols. **Disadvantages**: Requires local configuration of all service name and address changes.	Simple distributed networks with a small number of services that change infrequently.
Centralized Naming	**1.** Centralizes network names and addresses in a single place, facilitating administration of name changes and updates. For example, whenever a change is made to an existing server or a new server is added to the network, the change is made only once on one Names Server. This eliminates the need for an administrator to make changes to what potentially could be hundreds or even thousands of clients. **2.** Resolves service names across networks running different protocols **Disadvantages**: ■ Oracle Names stores network names and addresses for Oracle services only. ■ Requires additional setup and administration of Names Servers.	Large, complex networks (over 20 databases) that change on a frequent basis.
External Naming	Allows administrators to load Oracle service names into their native name service using tools and utilities with which they are already familiar.	Networks with existing name services.

3.4 Improving Large Network Performance

You may improve the performance of large networks by implementing one of the following:

- "Managing Connection Requests"
- "Connection Pooling"
- "Connection Concentration"
- "Optimizing Data Transfer by Adjusting the Session Data Unit (SDU) Size"
- "Persistent Buffer Flushing for TCP/IP"
- "Configuring Listener Queuesize"

3.4.1 Managing Connection Requests

If you expect your network to receive excessive connection traffic, you can use the network listener to manage these requests by redirecting them to either prestarted or prespawned dedicated server processes or dispatcher server processes.

Table 3–2 summarizes the relative advantages of each process and provides recommendations for using them in your network.

Table 3–2 Existing Server Processes

Process	Advantages	Recommended for:
Prestarted or prespawned dedicated server processes	1. Reduces connect time by eliminating the need to create a dedicated server process for each new connection request. 2. Provides better use of allocated memory and system resources by recycling server processes for use by other connections without having to shut down and recreate a server.	Networks where the Oracle shared or multi-threaded server is not supported, or where the creation of a new server process is slow and resource-intensive.

Table 3–2 Existing Server Processes

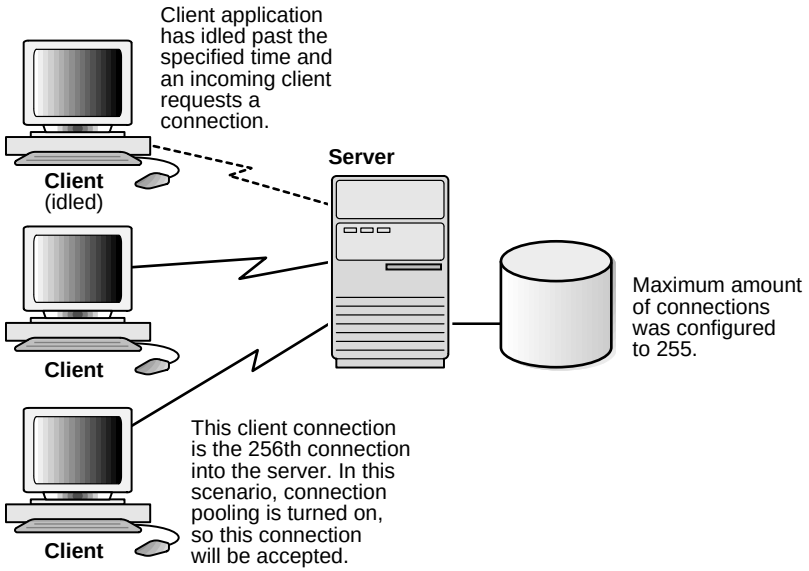
Process	Advantages	Recommended for:
Shared or dispatcher server processes	1. Utilizes network resources more efficiently than a dedicated server process, thus increasing the throughput and performance of your sessions. 2. Enables you to minimize the memory and processing resources needed on the server side as the number of sessions to the database increases.	Networks where the Oracle shared or multi-threaded server is supported, or where the creation of a new server process is slow and resource-intensive.

For more information on configuring your network listener to redirect connect requests to either prestarted or prespawned dedicated server processes, refer to Section 4.2.3.4, "Configuring Prestarted or Prespawned Dedicated Server Processes".

3.4.2 Connection Pooling

Connection pooling is a resource utilization feature that allows you to maximize the number of physical network connections to a multi-threaded server. This is achieved by sharing or pooling a dispatcher's set of connections among multiple client processes. Figure 3–1 shows how connection pooling works.

Figure 3–1 Connection Pooling



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

Connection pooling is enabled by parameters in your server's INIT.ORA configuration file. For more information, refer to the *Oracle8 Reference*.

3.4.3 Connection Concentration

Connection Concentration is a feature that is available through Oracle Connection Manager. It allows you take advantage of Net8's ability to multiplex or funnel multiple client sessions over a single transport to a multi-threaded server. Like connection pooling, concentration optimizes network resources and increases the number of client-server sessions that are possible across a fixed number of physical server ports. Unlike connection pooling, concentration maintains the transport connection.

3.4.3.1 Using Connection Pooling and Concentration

Table 3–3 summarizes the relative advantages with using connection pooling and concentration and provides recommendations for using them in your network.

Table 3–3 Connection Pooling and Concentration

Feature	Advantages	Recommended for:
Connection Pooling	1. Limits the number of network resources used per process. 2. Maximizes the number of client/server sessions over a limited number of physical connections. 3. Optimizes resource utilization.	Networks where many clients run interactive "high think/search time" applications such as messaging and OLAP.
Concentration	1. Supports large client populations. 2. Allows identification and monitoring of real users. 3. Allows mid-tier applications to support additional services. 4. Requires only a single transport for clients with multiple applications. 5. Requires only a single network connection for database links.	Networks where "continuous" connectivity is required.

3.4.4 Load Balancing

Load balancing is a feature that takes advantage of the fact that you can have multiple listeners for a single database or for two or more equivalent databases. By balancing the number of sessions coming into the listeners, you can improve connection performance. This may be achieved in one of two ways:

- Listener load balancing
- Client randomization

3.4.4.1 Listener Load Balancing

The listener load balancing feature allows you to distribute multiple incoming client sessions among several listeners. This feature helps to ensure that no single listener is overburdened. Periodically, each of the service handlers sends its load information to each listener that it is registered with. Thus, each listener knows how busy each of the handlers is and redirects incoming sessions to the least busy of those handlers.

Listener load balancing cannot be used in the following situations:

- Prespawned dedicated server processes cannot use listener load balancing because a prespawned dedicated server process registers itself only with the listener that started it.
- Oracle Parallel Servers have their own method of listener load balancing. For more information, refer to *Oracle8 Parallel Server Concepts and Administration*.

Listener load balancing is configured by defining multiple listeners for each database. Multiple listeners may exist either on the same platform as the database, or on different nodes as is the case with multi-threaded servers. For more information, refer to the *Oracle8 Administrator's Guide*.

3.4.4.2 Randomizing Client Requests Among Several Listeners

If more than one listener services a single database, a client will randomly choose between the listeners for its connect requests. This randomization allows all listeners to share the burden of servicing incoming connect requests.

To enable your clients to choose from listeners at random, you will need to configure the different listening addresses for each service name. For more information on configuring service name addresses, refer to Chapter 6, "Oracle Names".

3.4.5 Optimizing Data Transfer by Adjusting the Session Data Unit (SDU) Size

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, it is also possible to optimize data transfer by adjusting the size of the session data unit (SDU).

The SDU is a buffer that Net8 uses to place data before transmitting across the network. Net8 sends the data in the buffer either when requested or when it is full.

Table 3–4 outlines considerations when modifying the size of the SDU may or may not be appropriate.

Table 3–4 Considerations for modifying the size of the session data unit (SDU)

Modify session data unit size when:	Do not modify session data unit size when:
1. The data coming back from the server is fragmented into separate packets	1. Your application can be tuned to account for the delays
2. You are on a wide area network (WAN) that has long delays	2. You have a higher speed network where the effect of the data transmission is negligible
3. Your packet size is consistently the same	3. Your requests return small amounts of data from the server
4. Large amounts of data are returned	

Note: The SDU size should be set as a multiple of the normal transport frame size. Since the normal Ethernet frame size is 1024, the most efficient SDU size over an Ethernet protocol should be a multiple of 1024, but not more than four times the amount of 1024.

If you are using either connection pooling or connection concentration, keep in mind that these features require an additional 16 bytes per transport. For more information on the protocol frame size, refer to your protocol specific documentation.

You may adjust the session data unit size by adding a parameter in your local naming configuration file (TNSNAMES.ORA). For more information, refer to Section 5.4.3, “Configuring Advanced Service Name Options”.

3.4.6 Persistent Buffer Flushing for TCP/IP

Under certain conditions in some applications using TCP/IP, Net8 packets may not get flushed immediately to the network. Most often, this behavior occurs when large amounts of data are streamed from one end to another. The implementation of TCP/IP itself is the reason for the lack of flushing, and can cause unacceptable delays. To remedy this problem, you can specify no delays in the buffer flushing process. For more information, refer to the section titled, Persistent Buffer Flushing for TCP/IP in Section 4.3.2, “Configuring Persistent Buffer Flushing”.

3.4.7 Configuring Listener Queuesize

If you anticipate receiving a large number of connection requests for a listening process (such as a network listener, Oracle Connection Manager or Oracle Names) over TCP/IP, Net8 allows you to configure the listening queue to be higher than the system default. For more information, refer to Section 4.2.2.3, “Configuring the Listener to Handle Larger Volumes of Connection Requests”.

3.5 Planning Summary

Table 3–5 summarizes many of the options you may have chosen as you planned your network.

Table 3–5 Network Summary

Subject	Options
Network Layout	■ Single or Multiple Protocols
Service Name Resolution	■ Host Naming ■ External Naming ■ Centralized Naming ■ Local Naming

Table 3–5 Network Summary

Subject	Options
Connection Request Management	■ Dedicated Server Processes ■ Prestarted / Prespawned Dedicated Server Processes ■ Dispatcher Shared Server Processes
Network Performance	■ Connection Pooling ■ Connection Concentration ■ Listener Load Balancing ■ Optimizing the Session Data Unit Size ■ Persistent Buffer Flushing ■ Increasing the Listener Queue Size

4

Configuring Network Services

Net8 provides services that establish basic connectivity and enhance the manageability and scalability of your network.This chapter outlines procedures for configuring these services.

This chapter includes the following sections:

- Section 4.1, “Zero Listener Configuration”
- Section 4.3, “Configuring Protocol Specific Parameters”

4.1 Zero Listener Configuration

Before Oracle8 and Oracle7 servers can receive connections from Net8 clients, you must first start a network listener on the server node.

Start a listener and it will listen on a specific default address (Port 1521, TCP/IP, and interprocess communication addresses with KEY=PNPKEY). In a simple TCP/IP network not using Oracle Names, no further configuration is required.

4.2 Configuring the Network Listener

If however, you are using Oracle Names, or wish the listener to handle requests on other listening addresses, you will need to change and specify these preferences in a listener configuration file (called LISTENER.ORA).

The listener configuration file is composed of the following parts:

- Listener Name
- Listening Addresses
- Services
- Configuring Other Listener Features

4.2.1 Naming the Listener

The listener can be given any name. Enter any alphanumeric string to identify the name of the listener in the first line of the listener configuration file, for example:

```
MYLISTENER =
```

If no name is specified in the listener configuration file, the listener will be named LISTENER by default. If you use the default listener name, you will not need to specify the listener name within the startup command.

4.2.2 Configuring Listening Addresses

To instruct the listener to listen for connection requests on a specific address, you will need to configure the address protocol and any protocol specific parameters in an ADDRESS parameter similar to the one that follows in your listener configuration file:

```
(ADDRESS=(PROTOCOL=protocol name)(protocol specific information))
```

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

4.2.2.1 Defining Multiple Listening Addresses

Listeners may also be configured to listen on more than one address. To do this, you will need to specify an ADDRESS LIST in your listener configuration file. For example if a host machine is running both TCP/IP and SPX/IPX, the listener may be configured as follows:

```
listener=
    (address list=
        (address=(protocol=tcp)(host=sunshine)(port=1521))
        (address=(protocol=spx)(service=orasrvc1))
    )
```

4.2.2.2 Interprocess Communication (IPC) Listening Addresses

Interprocess communication (IPC) addresses identify both incoming connection requests from applications on the same node as the listener, and information sent or registered by a database dispatcher. To define IPC addresses, specify IPC as the protocol as well as any KEY values you are using in your listener configuration file:

```
(ADDRESS=(PROTOCOL=IPC)(KEY=string))
```

If identifying connection requests from the same node, the key value is equal to the service name of the database. If identifying a database dispatcher, the key value is equal to the database system identifier (SID). If the service name is the same as the SID, only one IPC address is needed.

4.2.2.3 Configuring the Listener to Handle Larger Volumes of Connection Requests

If you expect the listener to handle large volumes of connection requests, you may specify a queue for the process. This will allow the listener to dynamically handle larger numbers of concurrent connection requests.

To specify a queue size for a listener, enter a value to the QUEUESIZE keyword at the end of any listening address in your listener configuration file.

Example 4–1 depicts a typical listener configuration file with the queue size specified.

Example 4–1 Listener Configuration File with Queue Size Specified

```
listener=
    (address=
        (protocol=tcp)(host=acme.com)(port=1521)(queuesize=20)
    )
```

Note: Currently, you can only configure the queue size for listeners operating on TCP/IP and DECnet. The queue size value is operating system specific. On TCP/IP, the default queue size is set to 17.

4.2.3 Configuring the Listener for Database Services

The network listener may handle requests for more than one service. To configure information about database instances that the listener is servicing, you will need to provide the following information in the listener configuration file:

- Global Database Name
- Oracle Home Directory
- System Identifier (SID)

4.2.3.1 Global Database Name

The global database name is the name and domain name of the database as given in the database initialization parameter file. If you want to refer to the database by its global database name on the network, then you must specify that global database name to the listener.

4.2.3.2 Oracle Home Directory

The Oracle Home Directory identifies the Oracle Home location of the database that you are specifying. Its value is operating system specific. Table 4–1 lists examples of some operating system-specific strings:

Table 4–1 Operating System Specific Strings

Operating System	String
UNIX	(ORACLE_HOME=/usr/oracle)
VMS	(PROGRAM='disk$:[oracle.rdbms]tnslsnr.com')

Table 4–1 Operating System Specific Strings

Operating System	String
OS/2	(PROGRAM=ORACLE8)

4.2.3.3 System Identifier (SID)

The system identifier or SID is the Oracle system ID for the database server.

4.2.3.4 Configuring Prestarted or Prespawned Dedicated Server Processes

To create prespawned dedicated server processes, add the following four keywords in each SID_DESC in your listener configuration file:

- PRESPAWN_MAX
- PROTOCOL
- POOL_SIZE
- TIME_OUT

Example 4–2 depicts a typical SID_DESC in a listener configuration file that includes information about prespawned dedicated server processes:

Example 4–2 Typical SID_DESC in Listener Configuration File

```
sid_list_listener=(sid_list =
    (sid_desc =
        (global_dbname = sales.acme.com)
        (sid_name = db1)
        (oracle_home = /usr/bin/oracle)
        (prespawn_max = 99)
        (prespawn_list=
            (prespawn_desc=
            (protocol=tcp)(pool_size=10)(timeout = 2)
            )
        )
    )
)
```

4.2.3.4.1 PRESPAWN_MAX The maximum number of prespawned dedicated server processes the listener will create. This number must be at least as many as the sum of the pool size for each protocol. Set this value to a large number so that prespawned dedicated server processes are always available for new connections.

4.2.3.4.2 PROTOCOL The protocol on which the listener creates prespawned dedicated server processes.

4.2.3.4.3 POOL_SIZE The number of unused prespawned dedicated server processes for the listener to maintain on the selected protocol. Choose a number that is greater than 0 but no greater than the PRESPAWN_MAXIMUM value. The value should be about what you expect the average number of connections to be at any given time.

4.2.3.4.4 TIME_OUT Time in minutes that an inactive prespawned dedicated server process waits for the next connection. The value should be greater than 0. (A value of 0 will allow an inactive shadow process to continue indefinitely, thus wasting machine resources.) Set a short time out value. The time out is activated only after a prespawned dedicated server process has carried a connection and been disconnected. In other words, prespawned dedicated server processes that are waiting for their first connection do not time out.

4.2.4 Registering Information with a Names Server

If you are using Oracle Names, you can configure each listener to forward information about the database it is servicing to a Names Server by setting the USE_PLUG_AND_PLAY parameter to ON in your listener configuration file. For more information about the USE_PLUG_AND_PLAY_listener_name parameter, refer to “Listener Parameters (LISTENER.ORA)” section in Appendix B, “Configuration Parameters”.

4.2.5 Configuring Other Listener Features

For a complete list of parameters enabling you to configure additional network listener features including logging and tracing, refer to “Listener Parameters (LISTENER.ORA)” in Appendix B, “Configuration Parameters”. For a sample LISTENER.ORA file, refer to “Listener Configuration File (LISTENER.ORA)” in Appendix C, “Sample Configuration Files”.

4.3 Configuring Protocol Specific Parameters

The following protocols require you to configure additional parameters in a protocol specific configuration file. This file is called PROTOCOL.ORA:

- APPC/LU6.2
- ASYNC

- X.25
- OSI4

Protocols that require address information in your protocol specific configuration file have LOCAL_LOOKUP=*alias* as one of their address parameters in your local naming configuration file, or your listener configuration file. The LOCAL_LOOKUP parameter points to a non-global address in a PROTOCOL.ORA file.

The global address information for the server HORNET.WORLD is contained in the local naming and listener configuration files. This information can be used by any client in the network. The PROTOCOL.ORA entry contains additional address parameters needed for a specific node to reach HORNET.WORLD.

4.3.1 Configuring Validnode Checking

Validnode checking restricts a client's connection access to destinations with specific host privileges. The access list is defined in the your protocol-specific configuration file. To enable validnode checking, add the following parameter in your protocol-specific configuration file:

```
protocol.validnode_checking = yes
```

Not all protocols and operating systems support validnode checking. For more information, refer to Oracle operating system-specific documentation for your platform.

4.3.2 Configuring Persistent Buffer Flushing

To configure persistent buffer flushing, add the following parameter in your PROTOCOL.ORA file:

```
tcp.nodelay = yes
```

4.3.3 Configuring Dead Connection Detection

Net8 sends a probe periodically to verify that a client-server connection is still active. This is done to ensure that connections are not left open indefinitely, due to an abnormal client termination. If the probe finds a dead connection, or a connection that is no longer in use, it returns an error, causing the server process to exit.

To configure dead connection detection, you will need to configure a TNS Time-Out Value in a profile. You can do this by editing your profile using the following parameter:

```
SQLNET.EXPIRE_TIME
```

For information on how to configure this parameter using the Oracle Net8 Assistant, refer to Section 5.2.5.1, "TNS Time-Out Value".

For more information on this parameter, refer to the table titled SQLNET.EXPIRE_TIME in Appendix B, "Configuration Parameters".

4.3.3.1 Limitations

Limitations on using the dead connection detection feature are as follows:

- Dead connection detection is not allowed on bequeathed connections.
- Though very small, a probe packet generates additional traffic that may downgrade network performance.
- The server may need to perform additional processing to distinguish the connection probing event from other events that occur, depending on which operating system is in use. This may also result in downgrading network performance.

Some protocols may already include a native mechanism to perform the same functionality as dead connection detection. For more information, refer to Oracle operating system specific documentation for your platform.

5

Configuring Network Clients

Net8 provides you with a new tool called the Oracle Net8 Assistant to help you configure and manage your network clients easily and efficiently.

This chapter includes the following sections:

5.1 Configuring Network Clients Using Oracle Net8 Assistant

Configuration of network clients involves adding or editing parameters in one of the following:

- Profile
- Local Naming Configuration File

5.1.1 Profile

Net8 allows you to configure features on a client by adding or editing parameters in a profile. A profile is stored in a configuration file located on the client called SQLNET.ORA.

5.1.2 Local Naming Configuration File

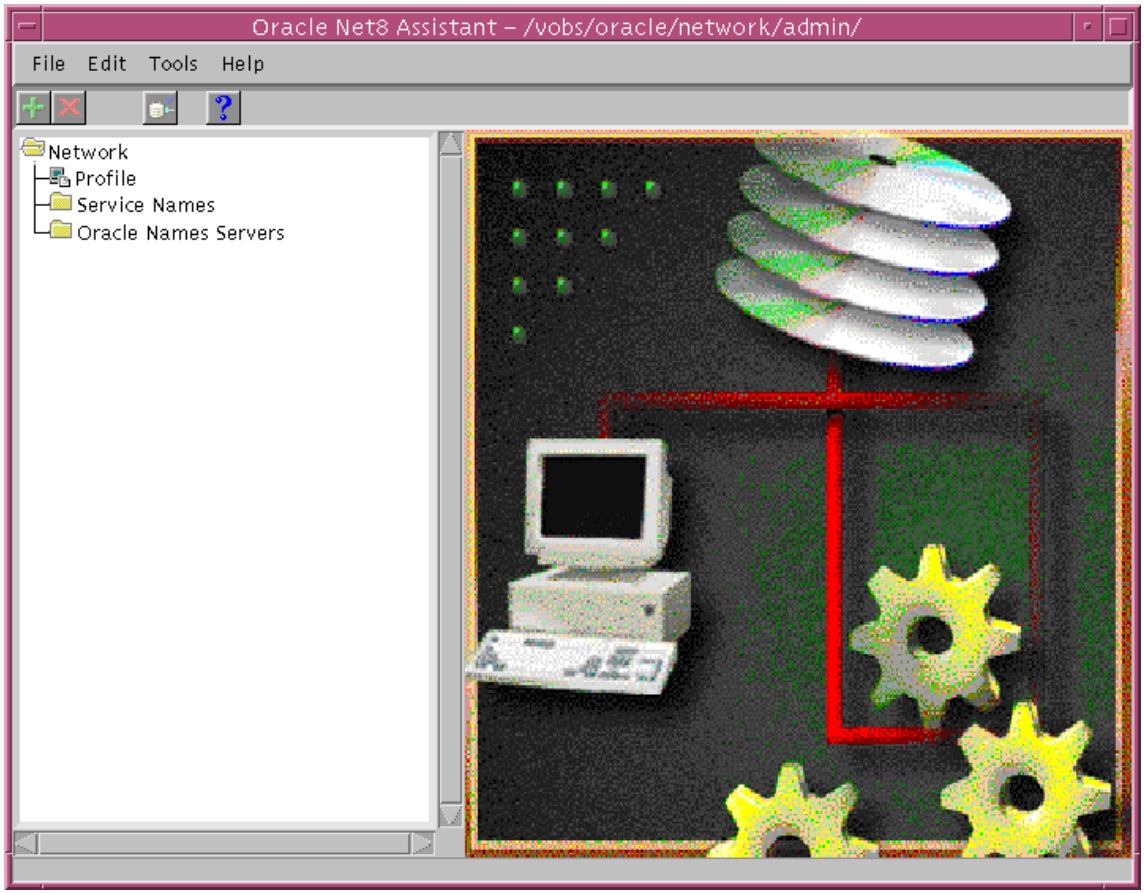
If you decide to use local naming as one of your methods to resolve service names, you will need to configure a local naming configuration file. Local naming functions much like an address book, allowing you to enter service names and their related network addresses, as well as optional configuration information, for use whenever a client wishes to connect to a database service. The local naming file is called TNSNAMES.ORA.

5.1.3 Oracle Net8 Assistant

The Oracle Net8 Assistant is a new configuration and management tool allowing you to setup and administer network features and components through a graphical user interface. You can use the Oracle Net8 Assistant to modify your profile and to configure service names on the client.

Figure 5–1 depicts what you see when you first start the Oracle Net8 Assistant. Note that "**Network**" is the first folder displayed in the tree structure. It includes the components of your Net8 network as viewed from your node.

Figure 5–1 Oracle Net8 Assistant Tree Directory



5.1.4 The Oracle Net8 Assistant and Java

The Oracle Net8 Assistant is implemented in Java and as such is available on any platform where Net8 and Java version 1.1.1 are available. The Oracle Net8 Assistant replaces client configuration functionality previously provided by Oracle Network Manager which was supported on Windows platforms only.

On-line Help is available for the Oracle Net8 Assistant. If you have any questions regarding how to use the Oracle Net8 Assistant, or if you are unclear about what you can do within a certain component, click on any **HELP** button.

5.1.5 Starting the Oracle Net8 Assistant

To start the Oracle Net8 Assistant, select it from the Programs menu on your desktop; or for UNIX platforms, run the shell script "netasst.sh" located in your "oracle/admin/bin" subdirectory.

5.2 Configuring a Profile Using the Oracle Net8 Assistant

A profile on the client configures functionality and defines how Net8 works to establish and maintain connections with services on the network. Use the Oracle Net8 Assistant to modify a profile and:

- configure naming methods
- enable tracing and logging features
- route connections through specific processes
- configure security features

The Oracle Net8 Assistant saves your preferences to a profile. A profile is stored and implemented through a configuration file called SQLNET.ORA. For a complete list of all available parameters that may be configured in a profile, refer to "Profile Parameters (SQLNET.ORA)" in Appendix B, "Configuration Parameters".

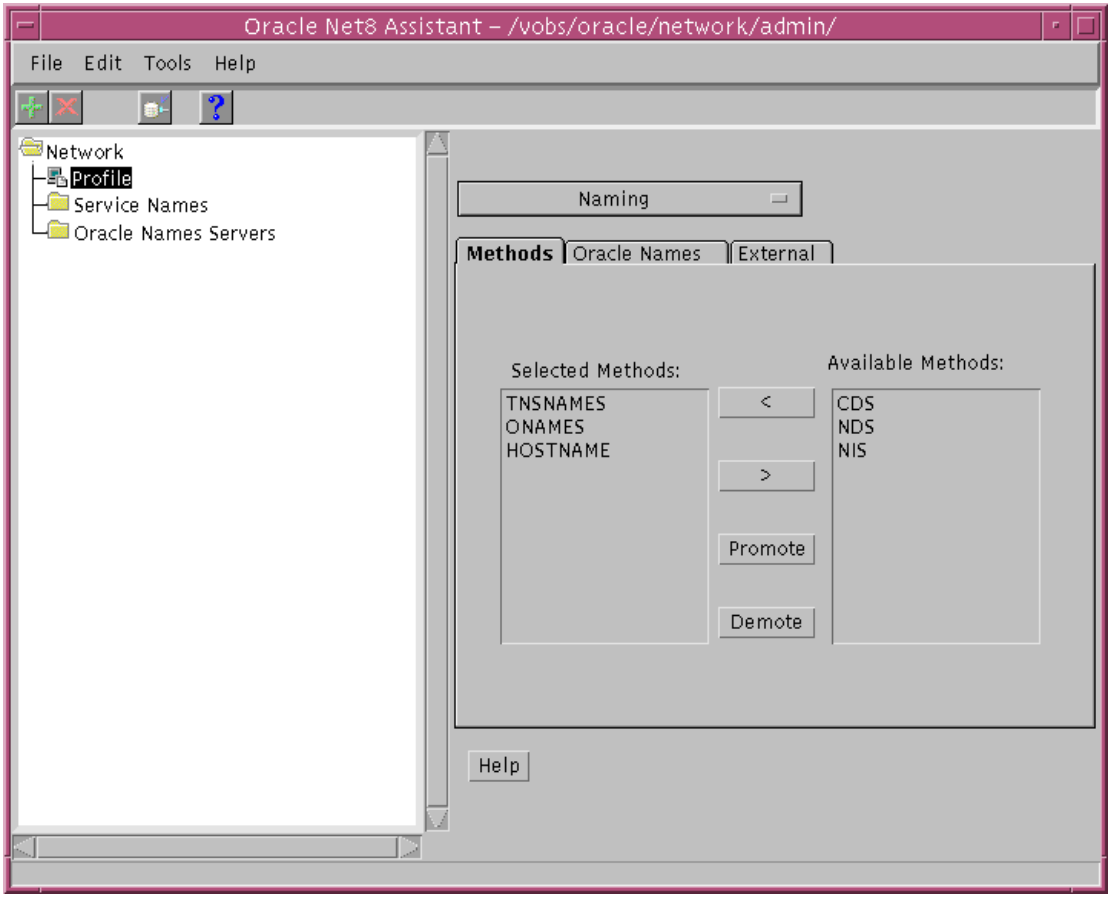
5.2.1 Configuring Naming Methods

Net8 will attempt to resolve a service name by using one or more naming methods. in the order in which they are specified in a profile until it establishes a connection. For more information on naming methods, refer to Section 3.3.1, “Naming Methods”.

Note that once you have selected naming methods on the client, additional configuration may be necessary for Oracle Names and external naming. For more information on configuring Oracle Names on the client, refer to Section 5.5, “Configuring Clients to Use Oracle Names”.

Figure 5–2 depicts the graphical user interface used in the Oracle Net8 Assistant to choose which naming methods a client will use as well as the order in which they will be tried.

Figure 5–2 Oracle Net8 Assistant Profile/Naming



5.2.1.1 Default Naming Methods

By default, Net8 will attempt to resolve a service name to a network address using the following three naming methods in the order that they appear:

- Local Naming (specified in the Oracle Net8 Assistant as TNSNAMES)
- Centralized Naming Using Oracle Names (specified in the Oracle Net8 Assistant as ONAMES)
- Host Naming (specified in the Oracle Net8 Assistant as HOSTNAME)

5.2.1.1.1 Connection Scenario Using Default Naming Methods If you decide to keep the naming methods that have been set by default, a typical connection scenario would proceed as follows.

A user may attempt to establish a connection to a database service called “hqserv” in the default domain “acme.com” by providing the following information:

User Name: scott

Password: XXXXX

Service Name: hqserv

Net8 would proceed as follows:

1. Verify if a local naming configuration file exists.
   a. If the file exists, Net8 will check to see if “hqserv” is defined as a service name. If “hqserv” exists, it will attempt to make the connection using the connect descriptor provided.
   b. If the file does not exist, Net8 will proceed to use the next naming method.
2. Verify if Oracle Names is in use by attempting to find a Names Server. Net8 will find a Names Server if either a Preferred Names Server is defined in a client profile, or if a Names Server List exists.
   a. If Net8 finds a Names Server, it will query the Names Server to resolve the service name.
   b. If Net8 does not find a Names Server or if a Names Server responds indicating that no such name exists, it will proceed to use the next naming method.

3. Use the host name adapter to resolve “hqserv”. If “hqserv” matches the host name or host name alias of the machine on a network, it will connect to a database on that machine with a global database name of “hqserv”.

5.2.1.2 Adding or Editing Naming Methods

If you wish to change the use or order of a naming method using the Oracle Net8 Assistant, proceed as follows:

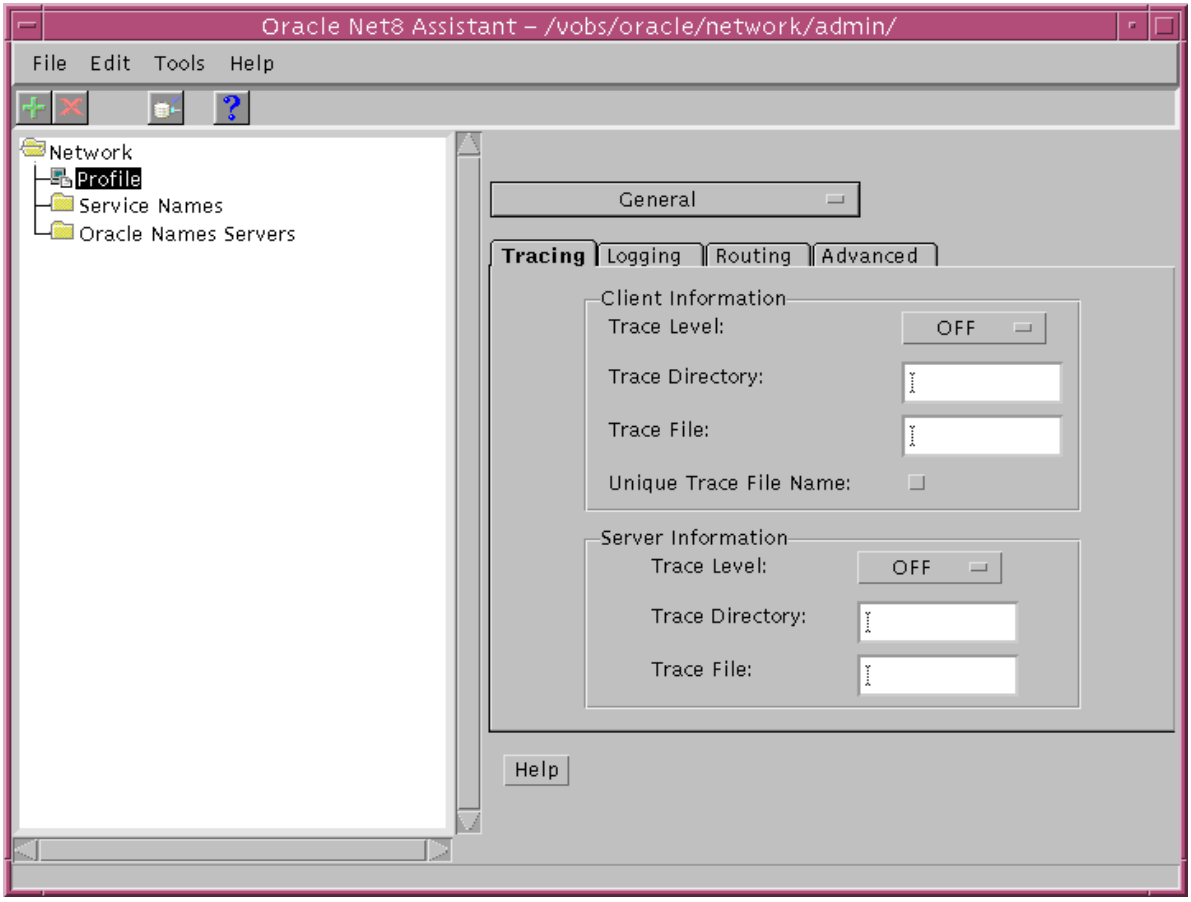
1. From the Oracle Net8 Assistant, click on the **Profile** icon in the directory tree.
2. Select “**Naming**” in the pull down menu.
3. Click on the “**Methods**” tab.
4. Specify the use of a naming method by moving it from the list of available naming methods to the list of selected methods.
5. Specify the order with which the naming methods will be used by promoting or demoting them in the list of selected methods.
6. Save your configuration.

5.2.2 Configuring Tracing Features

Use the Oracle Net8 Assistant to enable tracing on a client. Tracing is a feature that enables you to obtain information about the internal operations of Net8 which may be useful in troubleshooting problems in your networking environment. This feature is disabled by default.

Figure 5–3 depicts the graphical user interface used in the Oracle Net8 Assistant to configure tracing features in a profile.

Figure 5–3 Oracle Net8 Assistant Profile/Tracing



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

1. From the Oracle Net8 Assistant, click on the **Profile** icon in the directory tree.
2. Select "**General**" from the pull down menu.
3. Click on the "**Tracing**" tab.
4. Select a **trace level** from the pull down menu located in the **Client Information** section. You may choose from one of the following values to specify the level of tracing on a client or server:
   - OFF - Tracing disabled
   - USER - Trace information applicable for users
   - ADMIN - Trace information applicable for database administrators
   - SUPPORT - Trace information applicable for customer support staff
5. Enter any valid directory name in the **Trace Directory** field in the **Client Information** section to specify the directory to which trace files will be written on a client or server.
6. Enter any valid file name in the **Trace File** field in the **Client Information** section to specify the name of the trace file on the client or server.
7. Click on the "**Unique File Trace Name**" checkbox, if you want a unique identifier appended to each new trace file created.
8. Save your configuration.

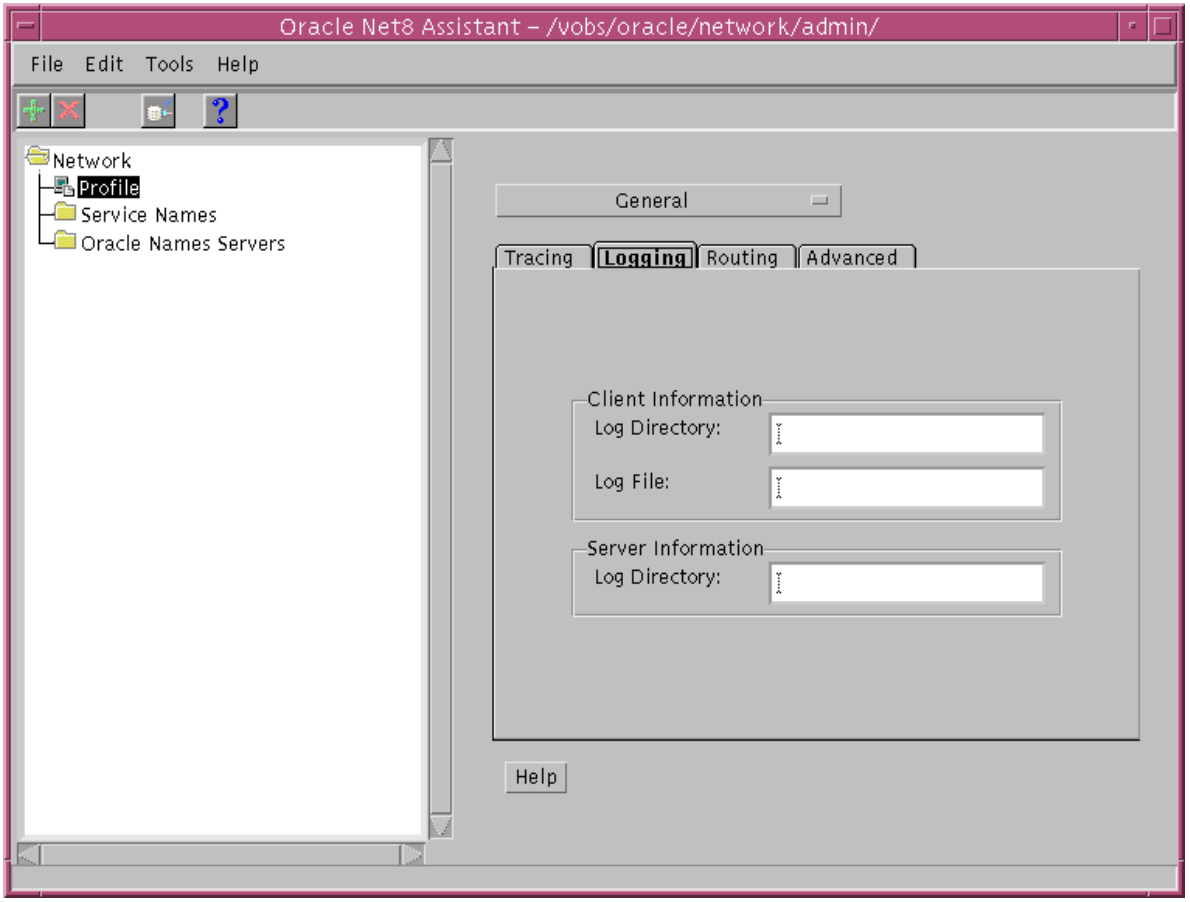
Note: The Server Information field in the Tracing tab panel is appropriate only for tracing networking operations on a server.

5.2.3 Configuring Logging Features

You may also use the Oracle Net8 Assistant to configure logging features on a client. Logging refers to the process by which network components note and append error-specific information to a log file.

Figure 5–4 depicts the graphical user interface used in the Oracle Net8 Assistant to configure logging features in a profile.

Figure 5–4 Oracle Net8 Assistant Profile/Logging



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

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

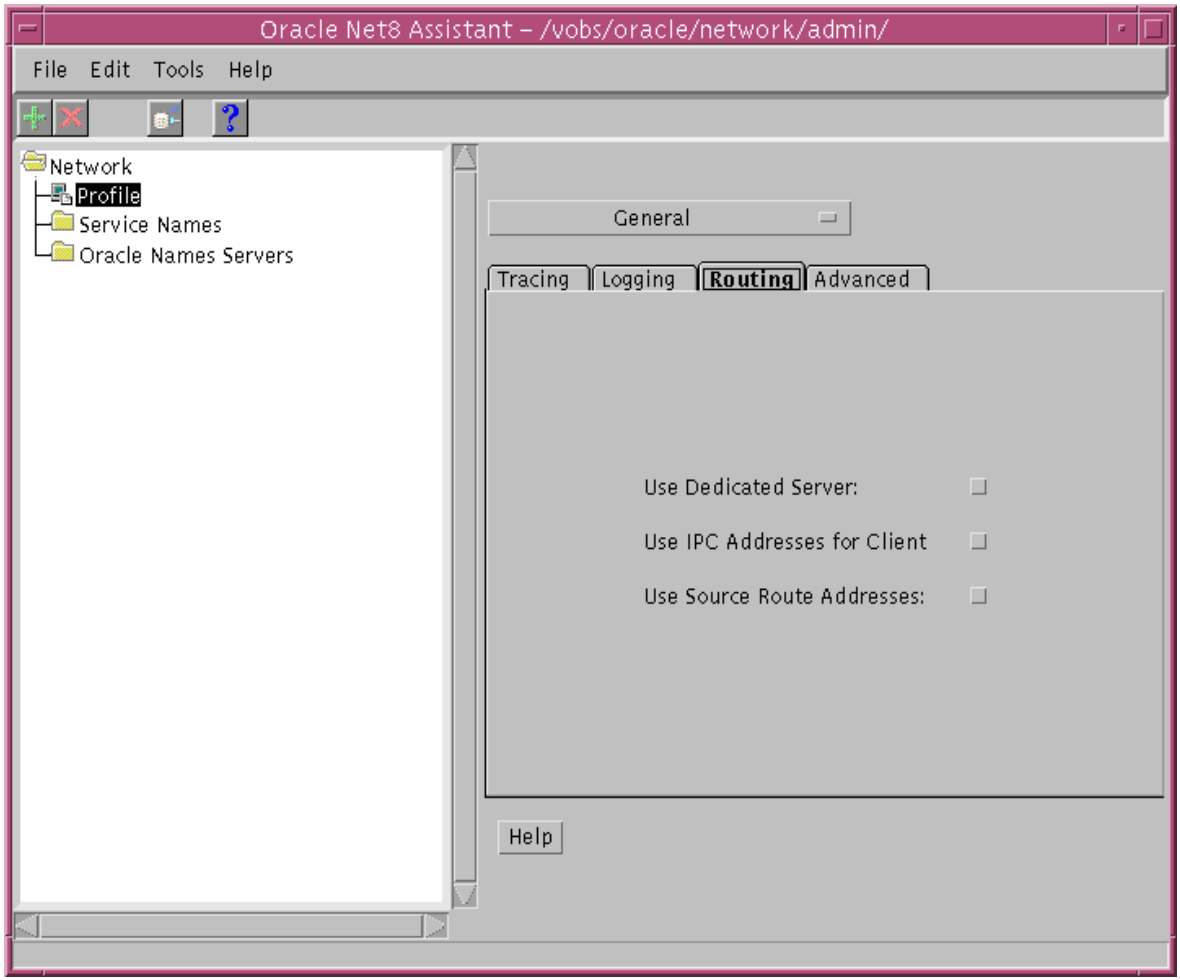
Note: The Server Information field in the Log tab panel is appropriate only for logging networking operations on a server.

5.2.4 Routing Connection Requests

From the General tab panel, you may specify the way that connection requests are routed from the client to a specific process, for example, always use a dedicated server process.

Figure 5–5 depicts the graphical user interface used in the Oracle Net8 Assistant to configure routing in a profile.

Figure 5–5 Oracle Net8 Assistant Profile/Routing



To route connection requests from the client a specific way, using the Oracle Net8 Assistant, proceed as follows:

1. From the Oracle Net8 Assistant, click on the **Profile** icon in the tree directory.
2. Select "**General**" from the pull down menu.
3. Click on the "**Routing**" tab.
4. Click on the "**Use Dedicated Server**" checkbox to cause all connection requests for this client to be made using a dedicated server.
5. Click on the "**Use IPC Addresses for Client**" checkbox to force the client to always attempt a local connection first using interprocess communication (IPC) addresses.
6. Click on the "**Use Source Route Addresses**" checkbox to route connections through Oracle Connection Manager first. For more information on how to route connection requests through Oracle Connection Manager, refer to Section 5.2.4, "Routing Connection Requests".

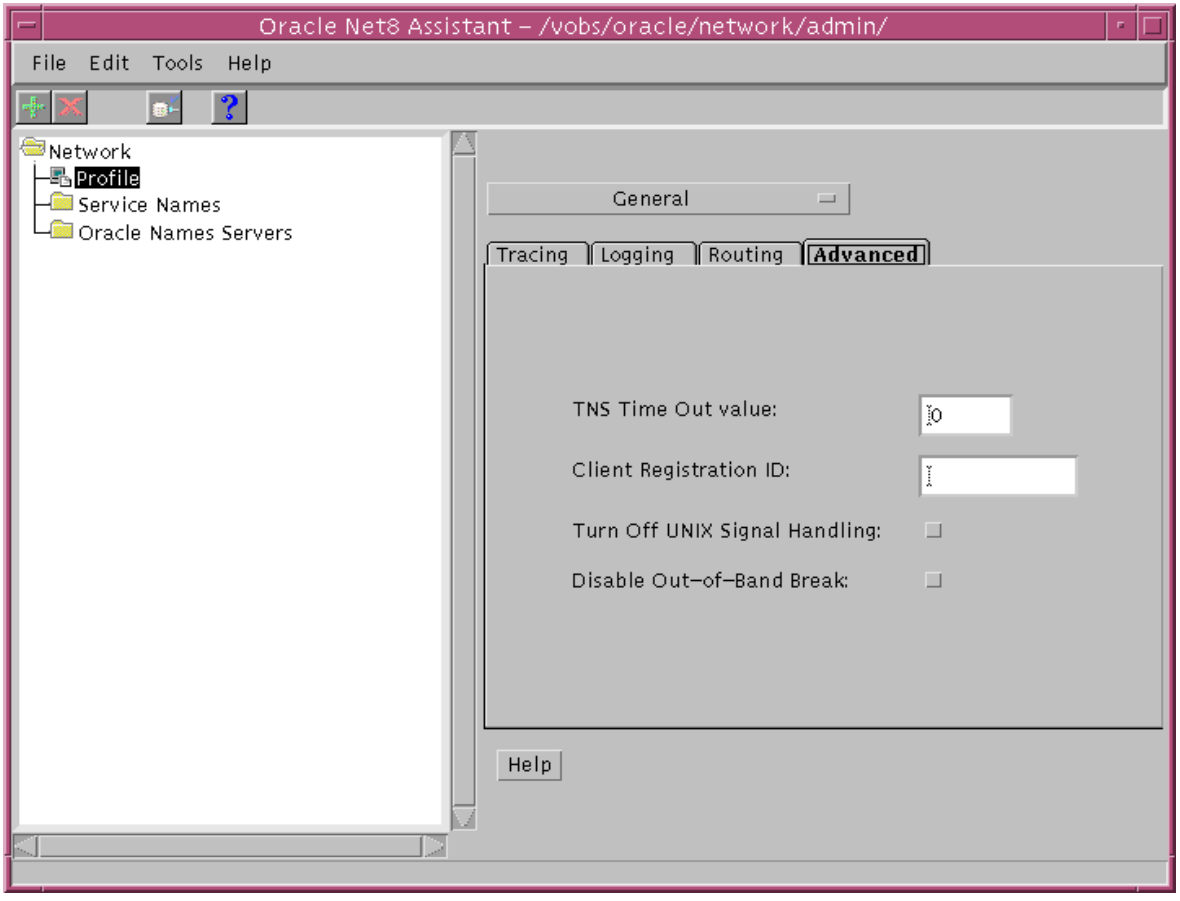
5.2.5 Configuring Advanced Net8 Functionality

Use the Oracle Net8 Assistant to configure the following advanced Net8 functionality:

- TNS Time-Out Value
- Unique Client Identifiers
- Signal Handling
- Out of Band Breaks

Figure 5–6 depicts the graphical user interface used in the Oracle Net8 Assistant to configure advanced Net8 functionality in a profile.

Figure 5–6 Oracle Net8 Assistant Profile/Advanced



5.2.5.1 TNS Time-Out Value

The TNS Time-out value is a Net8 server function.

5.2.5.2 Registering Unique Client Identifiers

From the "**Advanced**" tab panel, you may register a unique client identifier with the listener during a connection request. You can do this by entering any alphanumeric string up to 128 characters long in the **Client Registration ID** field. Net8 disables the unique client registration ID by default.

5.2.5.3 Turning Off Signal Handling

From the "**Advanced**" tab panel, you may turn signal handling off on UNIX systems. You can do this by clicking on the **Turn Off UNIX Signal Handling** checkbox. Net8 turns signal handling off by default.

5.2.5.4 Disabling Out of Band Breaks

From the "**Advanced**" tab panel, you may disable out of band breaks. You can do this by clicking on the **Disable Out of Band Breaks** checkbox. Net8 enables out of band breaks by default.

5.2.6 Configuring Security Features

To configure a client to use security features made available with either Oracle Security Server or Advanced Networking Option, proceed as follows:

1. From the Oracle Net8 Assistant, click on the **Profile** icon in the tree directory.
2. Select either "**Oracle Security Server**" or "**Advanced Networking Options**" from the pull down menu.
3. Add or edit parameters as applicable.
4. Save your configuration.

For more information on Oracle Security Server and Advanced Networking Option parameters, refer to Appendix B, "Configuration Parameters".

5.3 Configuring the Server as a Client

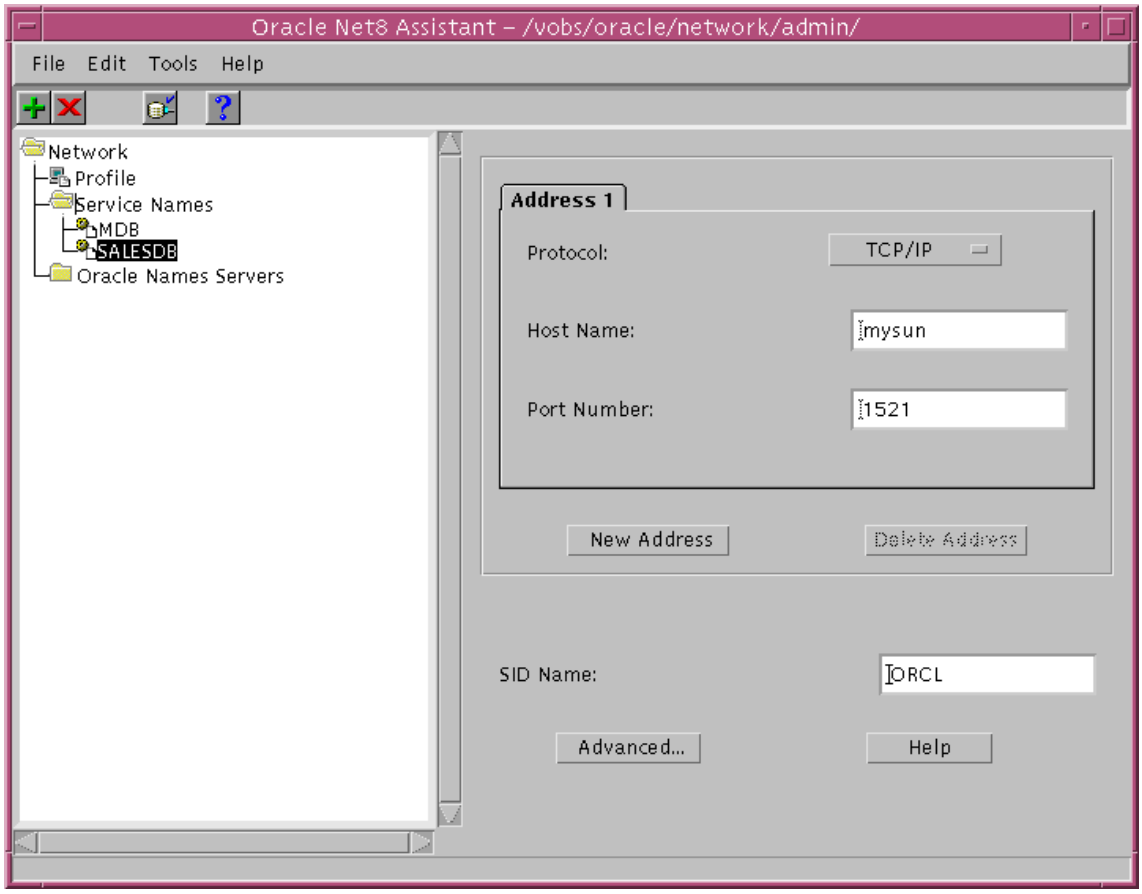
When using database links to initiate connections to other servers, the server needs all of the same configuration information as a client. This includes configuring local naming configuration file and a profile. In most situations, the profile used for the server can also be used for the client.

5.4 Configuring Service Names Using the Oracle Net8 Assistant

To add or modify service names, and to specify required network address and database identification information, as well as optional configuration information, associate with service names, use the local service name editing feature of the Oracle Net8 Assistant. When you save your additions or modifications, the Oracle Net8 Assistant will write this information to the local naming configuration file (TNSNAMES.ORA). This information will be used when local naming (TNSNAMES) is specified as a naming method in a profile.

Figure 5–7 depicts the graphical user interface used in the Oracle Net8 Assistant to configure service names.

Figure 5–7 Oracle Net8 Assistant Service Names Component

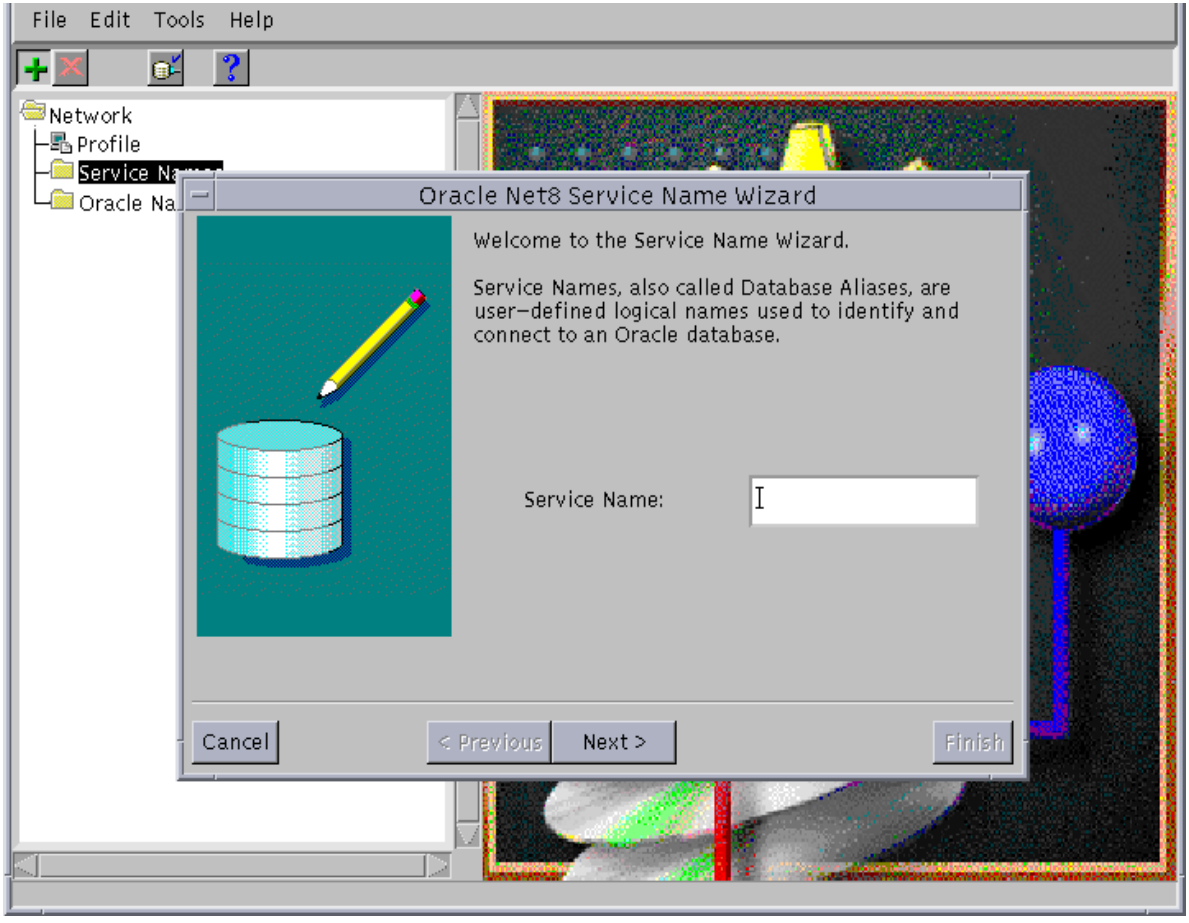


5.4.1 Adding Service Names

To add local service names using the Oracle Net8 Assistant, proceed as follows:

1. From the Oracle Net8 Assistant, double-click on the **Service Names** folder in the directory tree. If local service names exist, they will be displayed in the tree structure.
2. Select "**Create**" from the **Edit** menu or click on the "**+**" button to create a new service name. The Service Name Wizard will appear to guide you through the configuration process. Figure 5–8 depicts the graphical user interface used in the Oracle Net8 Assistant Service Name Wizard.
3. Be prepared to provide the following for each service name:
   - Address Information — which consists of the networking protocol you wish to use to connect and communicate with the database, and any information specific for each protocol. For example, if you were using TCP/IP, you will need the host name of the computer that the database is located on, and the port number of the database listener (this usually defaults to port 1521). For more information on the parameters specific to protocols, refer to Oracle operating system-specific documentation for your platform.
   - Database Identification Information — which consists of a protocol independent keyword indicating the System Identifier (SID) of the database instance.
4. Save your configuration.

Figure 5–8 Oracle Net8 Service Names Wizard



5.4.2 Modifying Service Names

To modify local service names using the Oracle Net8 Assistant, proceed as follows:

1. From the Oracle Net8 Assistant, double-click on the **Service Names** folder in the directory tree. If local service names exist, they will be displayed in the tree structure.
2. Click on any service name object that appears in the **Service Names** folder to modify parameters for a service name.
3. Save you configuration.

5.4.3 Configuring Advanced Service Name Options

You may also configure the following optional information:

- Global Database Name
- Session Data Unit (SDU) Size
- Source Route Addresses

5.4.3.1 Global Database Name

The global database name is the name and domain name of the database as given in the database initialization parameter file. If you want to refer to the database by its global database name on the network, then you must specify it with your service name.

5.4.3.2 Session Data Unit (SDU) Size

If you want to optimize 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. For more information on the session data unit, refer to Section 3.4.5, “Optimizing Data Transfer by Adjusting the Session Data Unit (SDU) Size”.

5.4.3.3 Source Route Addresses

If you wish to use any of the concentration, network access control, or multi-protocol support features provided by Oracle Connection Manager, you will need to configure your service name for source route addresses. This usually consists of the following:

1. configuring a Connection Manager’s listening address as your first address, and the server’s listening address as your second address.
2. specifying that you want to use source routing for these addresses. You can do this from the **Advanced** Service Names tab panel.

5.5 Configuring Clients to Use Oracle Names

To configure clients to use Oracle Names, you will need to perform the following three tasks:

1. Verify that each client is configured to use centralized naming using Oracle Names.
2. Verify that each client has discovered Names Servers on the network.
3. Start Client Cache (optional).

5.5.1 Configuring the Client to Use Centralized Naming

To configure the client to use Oracle Names as a naming method:

1. From the Oracle Net8 Assistant, click on the **Profile** icon in the directory tree.
2. Select "**Naming**" in the pull down menu.
3. Click on the "**Methods**" tab.
4. Verify that "**ONAMES**" is listed in the field of selected naming methods. If it is not, select it from the list of available methods and move it to the field of selected methods.
5. To specify additional preferences for how a client will use Oracle Names, click on the "**Oracle Names**" tab. Figure 5–9 depicts the Oracle Net8 Assistant Oracle Names tab panel. Features that you may configure from this panel include:

   a. Default Domain - indicates the domain name space from which the client will most often request a Names Server. When set, this name will automatically be appended to any unqualified name in an Oracle Names request. Net8 sets the default domain to NULL by default.

   Note: In prior SQL*Net releases, the default domain was set by default to ".world". The ".world" domain was therefore appended to all service names. If you wish to retain this feature with Net8, you will need to change the default setting to the ".world" domain.

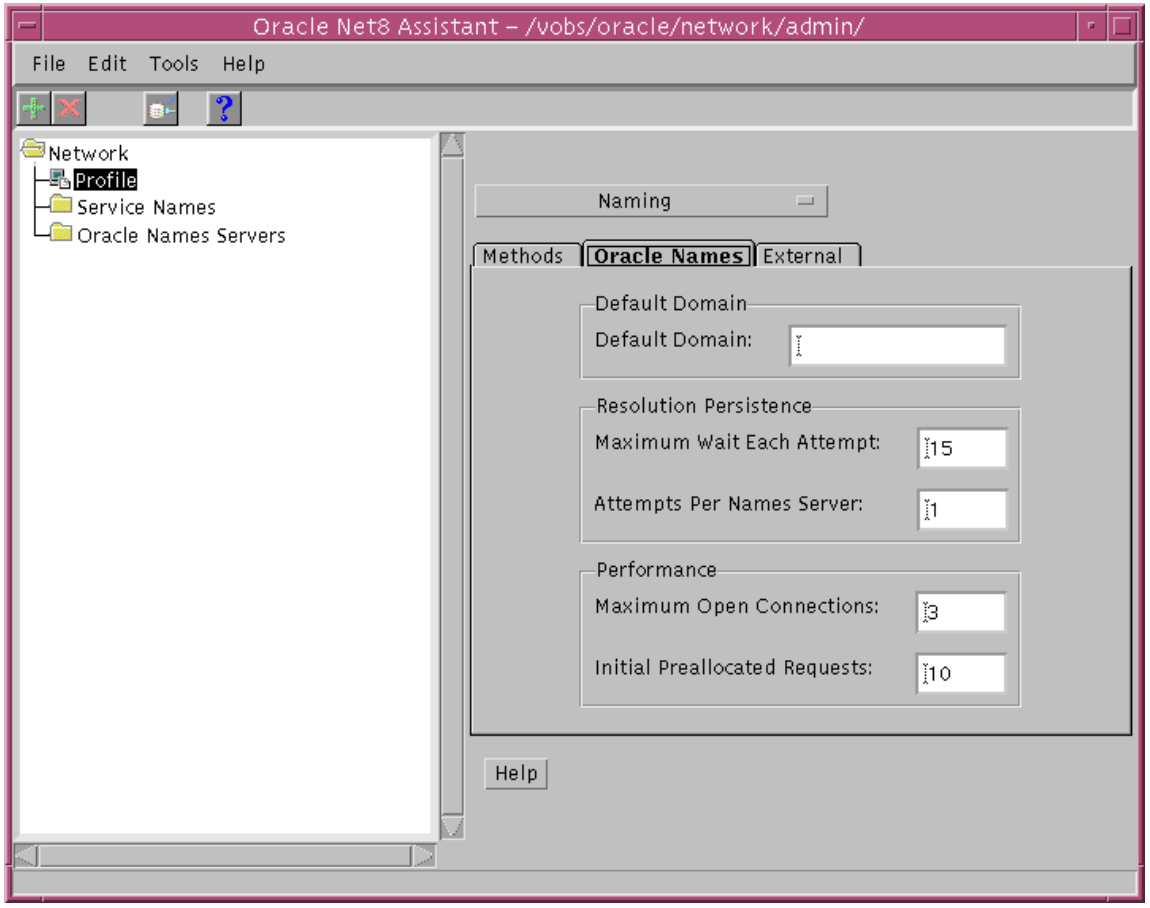
   b. Maximum Wait Each Attempt - specifies how long a client will wait for a response from a Names Server before reiterating the request to another Names Server. Net8 will wait for 15 seconds (operating system dependent) by default.

c. Attempts Per Names Server - specifies the number of times a client will attempt to iterate through the list of Names Servers before allowing the operation to fail. Net8 will attempt to iterate through the list of Names Servers once before allowing the operation to fail by default.

d. Maximum Open Connections - specifies how many connections an Oracle Names client may have open at one time. Net8 allows a client to have 10 connections open at any one time by default. This default value should be sufficient for almost all situations.

e. Initial Preallocated Requests - allows you to pre-allocate an initial number of messages in a client's message pool. These messages may be used for future requests to Names Servers. Net8 allocates 10 messages in the pool by default. This default value should be sufficient for almost all situations.

Figure 5–9 Oracle Net8 Assistant Profile/Naming Oracle Names Tab Panel



5.5.2 Discovering Names Servers on the Network

To discover Names Servers on the network, using the Oracle Net8 Assistant:

1. From the Oracle Net8 Assistant, click on the **Oracle Names Servers** folder.
2. From the **Tools** menu, select "**Discover Oracle Names Servers**". This will automatically find and order Names Servers on the network according to how quickly they respond. It will then generate a Names Server list with these results.
3. If the discovery process fails to find a Names Server on the network, a dialog box will appear requesting you to enter the address of any Names Server that you know.

Note: To initiate the discovery process, you can also issue a **REORDER_NS** command from the Oracle Names Control Utility (NAMESCTL). For more information, refer to the REORDER_NS command in the Section A.2, "Oracle Names Control Utility (NAMESCTL)" of Appendix A, "Control Utility Reference".

5.5.2.1 How the Discovery Process Works

Oracle Names attempts to discover Names Servers on the network by referencing the following:

- Preferred Names Server - which is a Names Servers whose address has been defined in a profile. If your network includes clients that were configured for a previous release of SQL*Net and Oracle Names, your profile may already contain this parameter.

Note: Addresses for Preferred Names Servers in your profile will override the Names Server list of the discovery process. If you have already generated a Names Server list, you should comment out or delete the addresses of any Preferred Names Server in your profile.

- Names Server List - which is created if a REORDER_NS command was executed previously or if the file was copied from another source.

- Well-Known Names Server - A “well known” Names Server is one that listens on a default well-known address. Names Servers started on TCP/IP will listen on port 1575 by default. They become well known if you create an alias through an IP address translation mechanism such as DNS, NIS or a centrally maintained TCP/IP hosts file, from the hostname to one of the following names: oranamesrvr0, oranamesrvr1, oranamesrvr2, oranamesrvr3, oranamesrvr4.

If a Names Server is found, the Oracle Names Control Utility will then send a query for all the Names Servers in the local region. It will then send a 'ping' to each of these servers to determine the time it takes for each Names Server to respond. It will then will sort a list of the Names Servers in increasing order of response time, and create or replace any existing Names Server List file with the sorted list of names and addresses.

5.5.3 Client Cache Daemon Process

Oracle Names version 8 provides a client cache daemon process that allows a client on most platforms to store information retrieved from a Names Server in its local cache. Whenever a client makes a subsequent connection request, it first checks its local cache for the address information. If the information is located, and if it has not exceeded a specified time to live (TTL), the client will use the information to connect to a server, saving the time it would normally take to re-query the Names Server.

The local client-side cache is particularly advantageous if at any time a Names Server is not available. In this case, the local client-side cache has a current list of recently accessed services.

5.5.3.1 Starting the Client Cache Daemon Process

To start the client cache, issue a START_CLIENT_CACHE command from the Oracle Names Control Utility.

For more information about the START_CLIENT_CACHE command, refer to “Oracle Names Control Utility (NAMESCTL)” in Appendix A, “Control Utility Reference”.

Note: Not all platforms support the client-side cache. For more information, refer to your Oracle platform-specific documentation

---

6

Oracle Names

Oracle Names is a distributed naming service developed for Oracle environments to help simplify the setup and administration of global, client/server computing networks.

This chapter describes features and functionality available with the version 8 release of Oracle Names. It also outlines procedures for configuring and controlling Oracle Names using the Oracle Net8 Assistant.

This chapter includes the following sections:

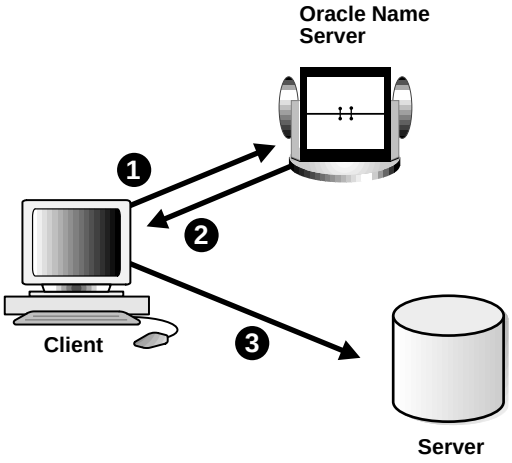
6.1 What Oracle Names Does

Oracle Names establishes and maintains an integrated system of Names Servers which work together like a directory service storing addresses for all the services on a network and making them available to clients wishing to make a connection.

Much like a caller who uses directory assistance to locate a telephone number, clients configured to use Oracle Names will refer their connection requests to a Names Server. The Names Server will attempt to resolve the service name provided by the client to a network address. If the Names Server finds the network address it will then return that information to the client. The client can then use that address to connect to the service.

Figure 6–1 depicts how Oracle Names works to help establish a connection between a client and server.

Figure 6–1 Oracle Names



6.1.1 Why Use Oracle Names?

Oracle Names provides an alternative to file-based or 'local' name resolution methods, where service names and addresses must be configured and maintained with each individual client. By maintaining this information in a central administrative location, Oracle Names reduces the work effort associated with adding or relocating services.

6.2 How Oracle Names Works

Names Servers may be configured and started on any node where Net8 is installed. You may use the Oracle Names Control Utility (NAMESCTL), or the Oracle Net8 Assistant to define service names and aliases and their associated values for use with Oracle Names. Alternatively, you may configure a network listener to register its services automatically with a Names Server. As services are started and shut down, the listener will dynamically register and de-register the connect descriptor and service name to a Names Server appropriately.

6.2.1 Continuous Replication vs. Database Storage of Service Names

Oracle Names also supports different modes for storing service registration data. For smaller workgroup environments where all of the services are registered dynamically, administrators may configure Names Servers to replicate data continuously among themselves. When a network listener registers a new service, information about that service will immediately be passed along to other Names Servers in the administrative region.

Alternatively, administrators in large environments will normally want to store their registration data in an Oracle database. If the Names Servers are configured to use an Oracle database as a repository, all service registrations (both static and dynamic) will be written to the database. Each Names Server in a given administrative region will periodically poll the region database for updated registrations. In this way, new registrations are communicated in a timely manner to all of the Names Servers in a given region. At the same time, it relieves Names Servers of the necessity to communicate directly with each other, as well as provides better reliability.

6.2.2 Single Region vs. Multiple Regions

Oracle Names also provides support for one or more administrative regions. An administrative region consists of a collection of Names Servers which share a common service registry. They do this either through continuous replication between all the Names Servers in the region, or by writing to and reading from a common Oracle database (also called the region database).

Most enterprise environments with multiple data centers and many Oracle instances will probably choose to take advantage of multiple administrative regions. This allows each data center to independently define and manage the services in its own environment. At the same time, all service addresses are continuously available to all of the clients in the environment. Names Servers transparently forward name resolution requests from clients in foreign administrative regions to the proper Names Server.

6.2.3 What Data is Stored in a Names Server

Table 6–1 describes the type of data stored in a Names Server.

Table 6–1 Data Stored by Oracle Names

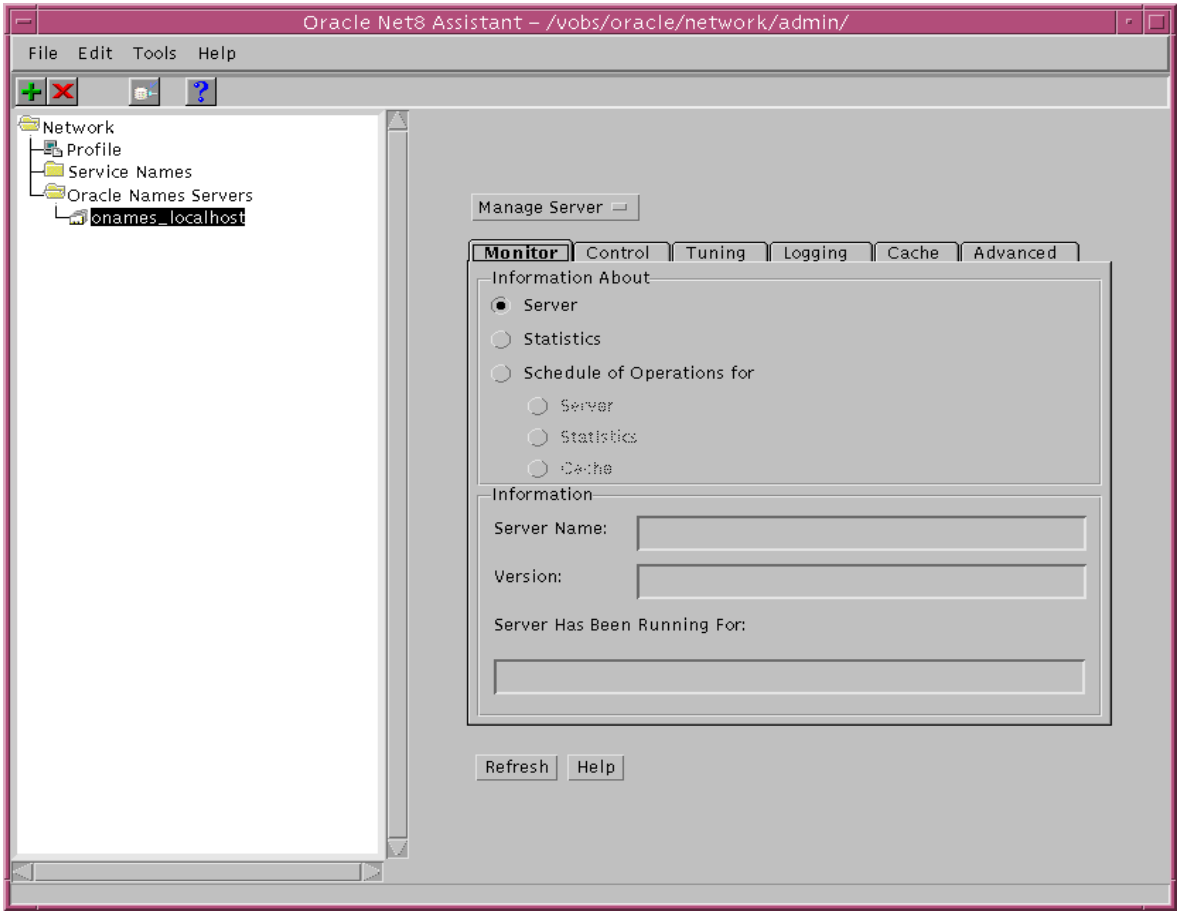
Data	Description
Other Names Server Names and Addresses	A Names Server stores the names and addresses of all other Names Servers in the same region. If there is more than one region in a network, the Names Server will store the name and address of at least one Names Server in the root region and each of the immediate sub-regions.
Service names	A Names Server stores the names and addresses of database services. A Names Server also stores gateways to non-Oracle databases and Oracle RDB databases.
Database links	Once the database address is registered, the name can be used as a global database link. Global database links, by default, use the current username and password and are made available to all users. These global database links may be supplemented by private and public database links created by individual users. For more information about private and public database links, refer to *Oracle8 Distributed Database Systems*.
Aliases	A Names Server stores aliases or alternative service names for any defined database service or database link.
Oracle Connection Managers	A Names Server stores the names and listening addresses of all Connection Managers on the network.

6.3 Using Oracle Names with the Oracle Net8 Assistant

To configure and control a Names Server, use the Oracle Net8 Assistant.

Figure 6–2 depicts the graphical user interface used to manage a Names Server using the Oracle Net8 Assistant.

Figure 6–2 Oracle Net8 Assistant Names Server Component



6.4 Configuring a Names Server

To configure a Names Server using the Oracle Net8 Assistant, proceed as follows:

1. From the Oracle Net8 Assistant, double-click on the **Oracle Names Server** folder in the directory tree.
2. Select "**Create**" from the **Edit** menu or click on the "**+**" button to start a new Names Server. The Oracle Names Server Wizard will appear to guide you through the configuration process.

Preferences will be saved in the Oracle Names configuration file (NAMES.ORA). For a complete list of parameters that are available in your Oracle Names configuration file, refer to "Oracle Names Parameters (NAMES.ORA)" in Appendix B, "Configuration Parameters".

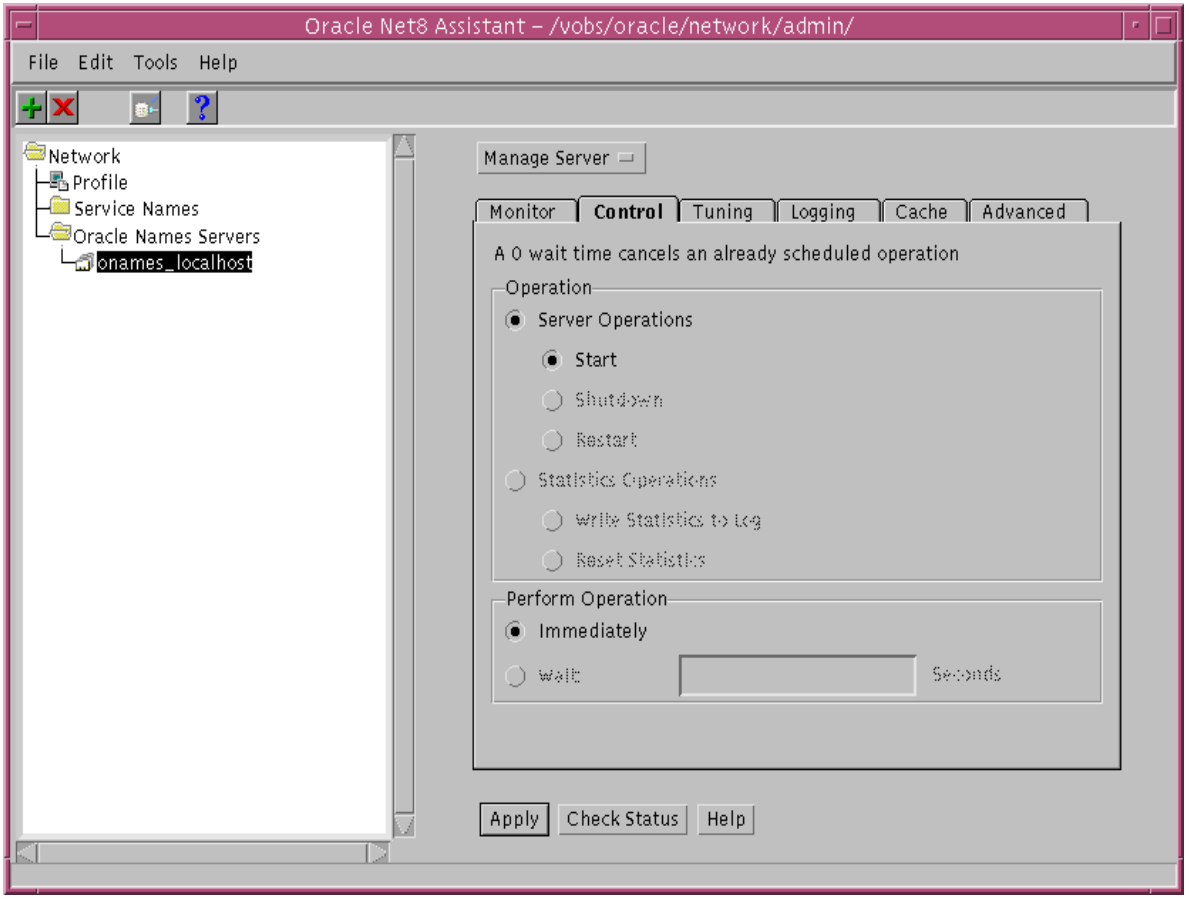
6.5 Starting a Names Server

To start a Names Server using the Oracle Net8 Assistant, proceed as follows:

1. Start Oracle Net8 Assistant.
2. Double-click on the **Oracle Names Server** folder in the directory tree.
3. Select "**Create**" from the **Edit** menu or click on the "**+**" button to start a new Names Server.
4. Select **Manage Server** from the pull down menu.
5. Select the **Control** tab panel.
6. Press the **Start** radio button from the **Server Operations** field.
7. Press **Apply**.

Figure 6–3 depicts the Control tab panel from the Manage Server pull down option.

Figure 6–3 Oracle Net8 Assistant Control Tab Panel From the Manage Server Pull Down Option



6.6 Loading Service Names Information Into a Names Server

To load information from a local naming configuration file into a Names Server, proceed as follows:

1. From the Oracle Net8 Assistant, click on the **Oracle Names Server** folder.
2. Select a Names Server object in the parent region.
3. Select **Manage Data** from the pull down menu.
4. Select the **Service Names** tab panel.
5. Enter the pathname for your current master local naming configuration file in the **Load Service Names from TNSNAMES.ORA File** section.
6. Press **Execute**.

6.7 Creating a Database to Store Names Server Information

If you decide to store your Names Server information in a database, you will need to create the database for the region. To do this, proceed as follows:

1. Use Server Manager and log into the Oracle database that you plan to use as the repository of the region's information, using the account that the Names Server will use to connect to the database. Note you will need to have the right to create tables when you log in.
2. Run the "namesini.sql" script provided with Oracle Names version 8. This script creates the tables needed by Oracle Names to store information.
3. Configure the Names Server with the default name and listening address by selecting "**Create**" from the **Edit** menu or clicking on the "**+**" button to start a new Names Server.
4. Add or edit region database information by selecting **Configure Server** from the pull down menu.
5. Start the Names Server.

6.7.0.1 Creating a Database in a Delegated Region

If you wish to create a database to store Names Server information in a delegated region, proceed as follows:

1. Create the database as you would if you were creating it in a single or root region.
2. Indicate a list of domains for which the Names Server is authoritative. You can do this by proceeding as follows:
   a. From the Oracle Net8 Assistant, click on the **Oracle Names Servers** folder.
   b. Select **Configure Server** from the pull down menu.
   c. Select the **Domains** tab panel.
   d. Enter information about the domains for which the Names Server is authoritative.
3. Indicate the name and address of one Names Server in the root region to the Names Server in the delegated region. You can do this by proceeding as follows:
   a. From the **Oracle Names Servers** component of the Oracle Net8 Assistant, select **Manage Data** from the pull down menu.

b. Select the **Topology** tab panel.

c. Click on the **Domain Hint** radio button.

d. Enter the name and address of one known instance of a Names Server in the root region in the **Topology** field area.

e. Press **Execute**.

4. Define the new region as a delegated sub-region. You can do this by proceeding as follows:

a. From the Oracle Net8 Assistant, click on the **Oracle Names Server** folder.

b. Select a Names Server object in the parent region.

c. Select **Manage Data** from the pull down menu.

d. Select the **Topology** tab panel.

e. Click on the **Delegate Domain** radio button.

f. Enter the name and address of one known instance of a Names Server in the delegated region in the **Topology** field area.

g. Press **Execute**.

6.8 Organizing and Naming Network Components

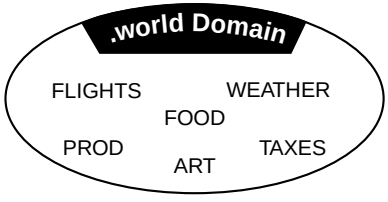
When you use Oracle Names, objects such as databases in a networked environment will need to be named in a way as to ensure that they are unique within the network. There are two basic models for naming objects in a network:

- Single Domain Model
- Hierarchical Naming Model

6.8.1 Single Domain Model

The use of the single domain naming model is useful if your network is small, and there is no duplication of names. Figure 6–4 depicts a typical flat naming structure using a single domain name .WORLD.

Figure 6–4 Single Domain Naming Model

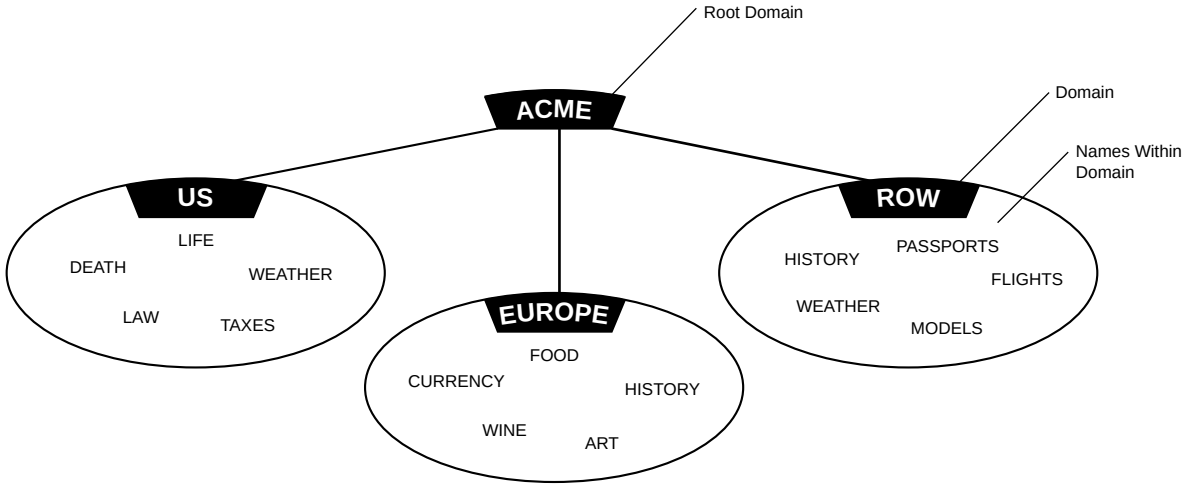


In this environment, database service names will automatically be appended with a ".WORLD" extension (for example, PROD.WORLD, FLIGHTS.WORLD, and so forth).

6.8.2 Hierarchical Naming Model

Hierarchical naming models divide names into a hierarchical structure to allow for future growth or greater naming autonomy. This type of naming model will allow more than one database with the same simple name in different domains. Figure 6–5 depicts a hierarchical structure of domains including the (ROOT) domain, ACME domain, US.ACME, EUROPE.ACME, and ROW.ACME (Rest of World) domains.

Figure 6–5 Hierarchical Naming Model



Notice in Figure 6–5 both WEATHER and HISTORY are repeated, but the global names remain unique (that is, HISTORY.ROW.ACME and HISTORY.EUROPE.ACME).

6.8.2.1 Domains

A domain is a logical group of machines and network services. Within each domain all names must be unique, but across domains simple unqualified names can be repeated.

Network domains are similar to file directories used by many operating systems in that they are hierarchical. Unlike file systems however, network domains may or may not correspond to any physical arrangement of databases or other objects in a network. They are simply names spaces developed to prevent name space conflicts.

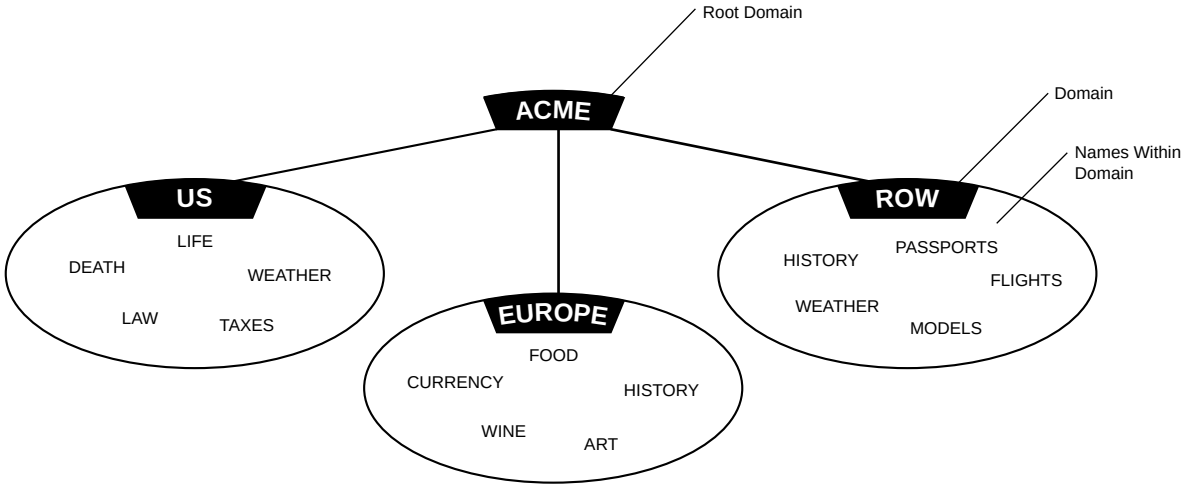
Note: Although they appear similar, the domains of an Oracle network are completely independent of Domain Name Service (DNS) name spaces. For convenience, you may choose to mirror the DNS conventions in your Oracle network.

6.8.2.2 Default Domains

The default domain is the domain within which most of the client's name requests are conducted. This is usually the domain in which the client resides, though it could also be another domain from which the client often requests services. A client can request a network service within its default domain using the service's simple, unqualified name, that is, without specifying a domain name. If a user requests a name without a "." character in it, the default domain name is automatically appended to the database service or database link name requested.

Figure 6–6 depicts a client with a default domain of EUROPE.ACME.COM. When it makes a request for the service name "WINE", the default domain name EUROPE.ACME.COM is appended to the requested name so that the name becomes WINE.EUROPE.ACME.COM.

Figure 6–6 Default Domains



For more information on domain names, refer to *Oracle8 Server Concepts*.

6.8.2.3 Multiple Domains

Multiple domains are related hierarchically to a root domain (the highest-level domain in the hierarchy) in a series of parent-child relationships. For example, under the root might be several domains, one of which is called COM. Under the COM domain might be several more domains, one of which is ACME. Under the ACME domain might be several domains, such as US, EUROPE, and so forth.

6.8.2.4 Using Consistent Domain Names

In previous releases of SQL*Net and Oracle Names, a network with only one domain, would by default be called ".world". This is no longer a requirement with Net8 and Oracle Names version 8. You may, however, want to keep the same convention to be backward compatible, as well as to avoid having to rename all your databases.

6.8.3 Using Regions to Decentralize Administrative Responsibilities

Most networks have one central point of administration, that is, one administrative region. A region is an administrative name space that defines a population of Names Servers. It is used to divide administrative responsibilities.

If you are using Oracle Names and your network is large or widely distributed geographically, you may choose to have several points of network administration. For example, if your network includes both the United States and Europe, you might want to have administrative decisions about the network made locally.

To delegate administrative regions, you must use a hierarchical naming model with each administrative region controlling one or more different domains.

6.8.3.1 How Multiple Region Networks Are Organized

Networks with multiple administrative regions must have one root administrative region and one or more delegated administrative regions.

Root Administrative Regions

The domain at the top of a hierarchy is known as the *root domain*. Similarly, the administrative region that encompasses the root domain is known as the *root administrative region*. The root defines the reference point within which the naming model and the administrative model function. The root administrative region provides a common thread among all delegated administrative regions in a hierarchical naming structure. The root administrative region requires:

- Names Servers in the root region.

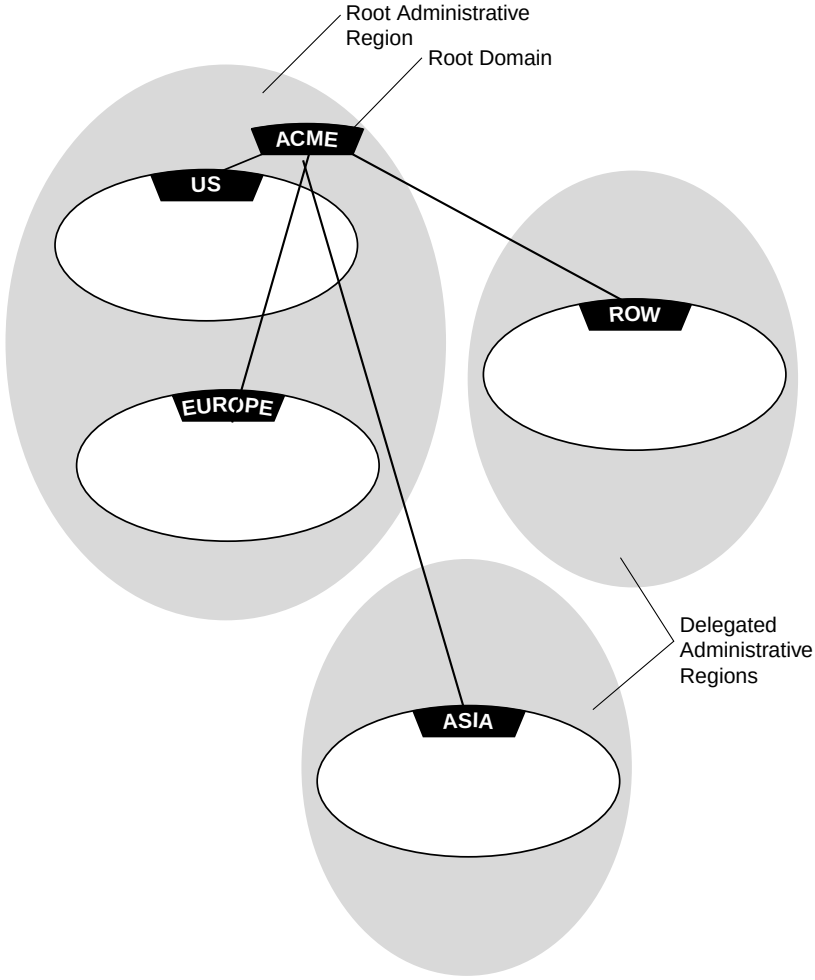
- Domains as they are administered in this region. This is always at least the root domain, and can include other domains.
- Delegated administrative region Names Servers—The domains and Names Server addresses in any alternate regions which act as direct child regions of the root.
- Data definitions for the root region—all of the database service names, database links and aliases associated with the root administrative region.

Delegated Administrative Regions

Administrative regions can be ”delegated” from the top of the hierarchy down to other domains in the naming model. For example, a naming model with ten domains can have between one and ten administrative regions.

All administrative regions other than the root are hierarchically delegated directly or indirectly from it. Figure 6–7 depicts a network with six domains and three administrative regions: the ROOT, and two delegated regions (DR1, DR2).

Figure 6–7 Delegated Administrative Regions



Delegated Administrative Regions Below Root

All administrative regions below the root are considered delegated administrative regions. Each delegated region must know:

- All Name Servers and domains in the region
- Domains and Names Server addresses in any of this administrative region's child regions
- Addresses of the Names Servers in the root region. Having this data allows Names Servers in delegated regions to contact any other region (through the root region)
- Data definitions—All of the database service names, database links, and aliases for all of the domains in this local (delegated) administrative region

7

Oracle Connection Manager

Oracle Connection Manager is a multipurpose, networking service that provides connection concentration, network access control, and multiprotocol connectivity functionality. It is available only with the Oracle8 Enterprise Edition.

This chapter describes the features and functionality available with Oracle Connection Manager. It also outlines installation notes and configuration procedures.

This chapter includes the following sections:

7.1 What Oracle Connection Manager Does

Oracle Connection Manager is a new application that is available with Oracle8 Enterprise Edition. It provides support for the following features:

- "Connection Concentration"
- "Network Access Control"
- "Multiple Protocol Support"

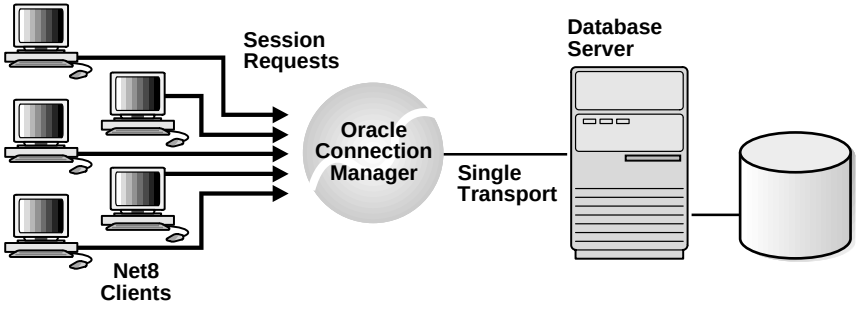
7.1.1 Connection Concentration

Oracle Connection Manager enables you to take advantage of Net8's ability to multiplex or funnel multiple logical client sessions through a single transport connection to a multi-threaded server destination. This is accomplished through Oracle Connection Manager's connection concentration feature.

Concentration reduces the demand on resources needed to maintain multiple connections between two processes by enabling the server to use fewer connection end points for incoming requests. This enables you to increase the total number of sessions that a server can handle. By using multiple Connection Managers, it is possible for thousands of concurrent users to connect to a server.

Figure 7–1 depicts how concentration works.

Figure 7–1 Connection Concentration Through Oracle Connection Manager



7.1.2 Network Access Control

Oracle Connection Manager also includes a feature which you can use to control client access to designated servers in a TCP/IP environment. By specifying certain filtering rules you may allow or restrict specific clients access to a server based on the following criteria:

- Source hostname(s) or IP address(es) for clients
- Destination hostname(s) or IP address(es) for servers
- Destination database server identifier

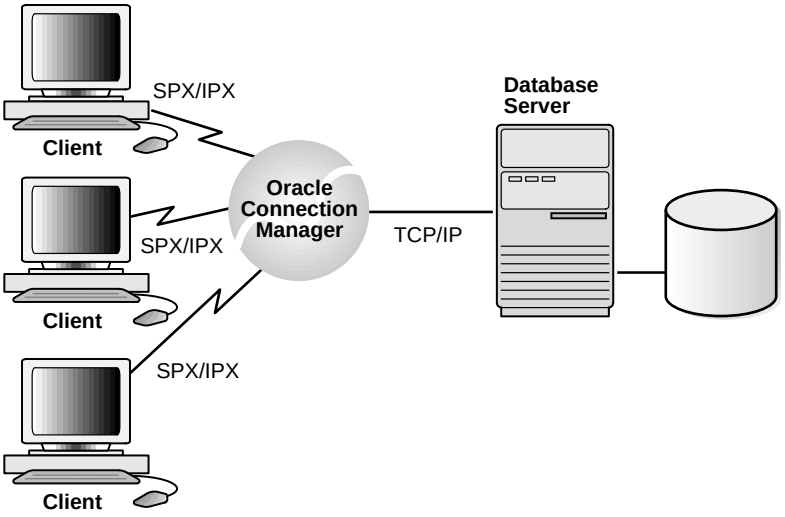
7.1.3 Multiple Protocol Support

Oracle Connection Manager also provides multiple protocol support enabling a client and server with different networking protocols to communicate with each other. This feature replaces functionality previously provided by the Oracle MultiProtocol Interchange with SQL*Net version 2.

Net8 can traverse as many networking protocol stacks as can be installed and supported. In fact, the number of networking protocols supported is limited only by those restrictions imposed by the specific node's hardware, memory and operating system.

Figure 7–2 depicts how a client in an SPX/IPX network can route its session to a server over a TCP/IP transport through Oracle Connection Manager.

Figure 7–2 Multiprotocol Support Through Oracle Connection Manager



7.2 How Oracle Connection Manager Works

Oracle Connection Manager acts like a router through which client connection requests may either be sent to its next hop or directly to a server. Clients who route their connection requests through a Connection Manager may then take advantage of the connection concentration, network access control, or multiprotocol support features configured on that Connection Manager.

7.2.1 Connection Manager Processes

There are three main processes associated with Oracle Connection Manager:

- CMGW
- CMADM
- CMCTL

Note: On desktop platforms, the executables are CMGW80, CMADM80 and CMCTL80.

7.2.1.1 CMGW

CMGW is a gateway process acting as a hub for the Connection Manager. This process is responsible for the following:

- registering with the Connection Manager Administration process
- listening for incoming SQL*Net 2.x or NET 8.x connection requests. By default it listens on port 1600
- initiating connection requests to listeners for clients
- relaying data between the client and server
- answering requests initiated by CMCTL

7.2.1.2 CMADM

CMADM is a multi-threaded process that is responsible for all administrative issues of the Connection Manager. Its primary function is to maintain address information in the Oracle Names Server for the SQL*Net 2.x and NET 8.x clients. Other responsibilities include:

- processing the CMGW registration

- locating local Oracle Names Servers
- identifying all listeners serving at least one database instance
- registering address information about the CMGW and listeners
- monitoring changes in the network and update the Names Server
- answering requests initiated by CMCTL

Communication between CMGW and CMADM is done via interprocess communications. The Connection Manager periodically goes to the Names Server to update its cache of available services.

7.2.1.3 CMCTL

CMCTL is the executable name for the Connection Manager control utility. It provides administrative access to CMADM and CMGW. For more information on the Connection Manager Control Utility, refer to Section A.3, “Connection Manager Control Utility (CMCTL)” in Appendix A, “Control Utility Reference”.

7.3 Configuring Oracle Connection Manager

You may configure features specifying how the Connection Manager will work by adding or editing parameters in the Connection Manager configuration file (CMAN.ORA). This file consists of three sections:

- CMAN (which defines the listening address for the Connection Manager)
- CMAN_PROFILE (which defines general configuration parameters)
- CMAN_RULES (which defines rules for filtering incoming connection requests)

7.3.1 Configuring the Connection Manager to Listen on Multiple Addresses

Connection Manager will listen by default on Port 1600, using TCP/IP. The Connection Manager may also be configured to listen on other listening addresses. To do this, you will need to configure those addresses in the CMAN parameter.

Example 7–1 depicts how the Connection Manager configuration file would look if you configured it to listen on two addresses, SPX and TCP/IP. Note that a Connection Manager can listen on any protocol that Oracle supports.

Example 7–1 CMAN Parameter Configured for Multiple Listening Addresses

```
CMAN=
    (ADDRESS_LIST=
       (ADDRESS=
            (PROTOCOL=SPX)
            (SERVICE=CMAN)
       )
       (ADDRESS=
           (PROTOCOL=TCP)
           (HOST=CMAN.US.ORACLE.COM)
           (PORT=1600)
       )
    )
```

7.3.2 Enabling Connection Concentration Features

To use Oracle Connection Manager's connection concentration feature, proceed as follows:

1. verify that the destination server is configured as a Multi-Threaded Server and that the multiplexing feature is turned on. This is done by setting the MTS_DISPATCHERS parameter in the database initialization file (INIT.ORA) for that instance as follows.

```
MTS_DISPATCHERS = (MULT=ON)
```

2. route client connection requests you want concentrated to the database server through that Connection Manager. For more information on routing your connection requests through Connection Manager, refer to Section 7.3.4, “Configuring Clients to Use Oracle Connection Manager”.

For more detailed information on the Multi-Threaded Server, refer to the *Oracle8 Concepts* manual.

7.3.3 Specifying Network Access Control Rules

To configure the Connection Manager to control access to your database server, proceed as follows:

1. configure values in the CMAN_RULES parameter
2. route client connection requests you want subject to network access control rules to the database server through that Connection Manager. For more information on routing your connection requests through Connection Manager, refer to Section 7.3.4, “Configuring Clients to Use Oracle Connection Manager”.

Example 7–2 depicts filtering rules specifying network access control in a CMAN_RULES parameter. Note that parameter fields in the RULES section are defined as follows: shost is the source hostname or IP address of session request (client); dhost is the destination hostname or IP address (server); services is the SID name; and ACT: specifies whether or not the Connection Manager will either accept or reject the incoming requests based on the above three parameters.

Example 7–2 Filtering Rules in CMAN_RULES

```
CMAN_RULES=
   (RULE_LIST=
        (RULE=
            (SRC = shost)
            (DST = dhost)
            (SRV = services)
            (ACT = accept | reject)
        )
    )
```

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

Note that Oracle Connection Manager can only apply access control in a TCP/IP network.

7.3.4 Configuring Clients to Use Oracle Connection Manager

If you want to use any of the features included with Oracle Connection Manager, including connection concentration, network access control, or multiprotocol support, you will need to:

1. modify the client's profile to route connection requests through the Connection Manager. For more information on routing connections through the Connection Manager, refer to Section 5.2.4, "Routing Connection Requests".
2. verify that you are using either centralized naming using Oracle Names or local naming as a naming method.
   - If you are using Oracle Names, the Connection Manager will automatically update addresses in the Names Server as a source route address, by inserting the address for the Connection Manager into the existing address.

     Note: If more than one Connection Manager is used in the connection path (more than 1 hop), you cannot use Oracle Names to connect clients through it.

   - if you are using local naming, you will need to specify a path of addresses through the Connection Manager as a source route address. For more information on configuring source route addresses, refer to Section 5.4.3, "Configuring Advanced Service Name Options".

7.4 Starting Oracle Connection Manager

You may start Oracle Connection Manager from any node where Net8 is installed. To start Oracle Connection Manager, use the Connection Manager Control Utility (CMCTL) to issue a START command as follows:

```
CMCTL START CMAN
```

A status message will appear if Oracle Connection Manager has started successfully. For more information on the Connection Manager Control Utility, refer to Section 8.3.3, “Using the Connection Manager Control Utility (CMCTL)”.

8

Using Net8

Once you have completed configuring your network, you should start and test each component to ensure that they are functioning properly. Net8 provides a variety of tools to help you start, test and control a Names Server, network listener, and Connection Manager.

This chapter outlines procedures to use Net8's control utilities to operate each component once they have been configured. This chapter contains the following sections:

8.1 Procedures to Get the Network Running

After installing and configuring all the network components, you need to start them to make the network functional. Following is an outline of the steps you should take to start the network components. For more detailed information about using the control utilities, refer to Appendix A, "Control Utility Reference"

1. If your network uses Oracle Names, start the Names Servers, using either the Oracle Net8 Assistant or the Oracle Names Control Utility.
   a. If the Names Servers use a database to store the network information, you will need to start the database first, then start the listener.
   b. If the Names Servers are not using a database, start a Names Server called "onames_oranamesrvr0", then issue a command to discover Names Servers on each Names Server node.
   c. Start each Names Server.
   d. Discover Names Servers on all client nodes.
2. Start the listeners using the Listener Control Utility. On each listener node, enter the following command:

   ```
   lsnrctl start
   ```

3. Start the databases, using the tool of your choice.
4. If your network includes Oracle Connection Managers, start them using the Connection Manager Control Utility. On each Oracle Connection Manager node, enter the following command:

   ```
   cmctl start cman
   ```

You should now be able to make connections across the network.

The remainder of this chapter provides more information about the control utilities, how to test connections between components, and how to resolve the most common problems.

8.2 Net8 Component Testing Methodology

The preferred sequence for testing the network is as follows:

- start and test each Names Server (if included in your network layout)
- start and test each listener
- start and test each Oracle Connection Manager (if included in your network layout)
- test each client by initiating a connection from the client to the server

8.3 Net8 Control Utilities

Net8 provides the following tools to help you start, test and control each network component.

- Oracle Names Control Utility — NAMESCTL
- Listener Control Utility — LSNRCTL
- Connection Manager Control Utility — CMCTL

For more information about Net8's component control utilities and their commands, refer to Appendix A, "Control Utility Reference".

Net8 Tools

In addition, Net8 provides the following tools to help you evaluate network connectivity:

- TNSPING
- TRCROUTE

8.3.1 Using the Oracle Names Control Utility (NAMESCTL)

The Oracle Names Control Utility, NAMESCTL, is a tool that you run from the operating system prompt to start and control the Names Server.

The general form of the Oracle Names Control Utility is:

`NAMESCTL` *`command`*

You can also issue NAMESCTL commands at the program prompt. When you enter NAMESCTL on the command line, the program is opened. You can then enter the desired commands from the program prompt. For example, the following command starts the Names Server.

```
NAMESCTL> startup
```

The STARTUP command of NAMESCTL loads the Names Server into memory and tells it to begin executing. At startup, the Names Server loads its configuration and data. On each Names Server node, invoke NAMESCTL from the prompt to start the utility.

For additional information about NAMESCTL and its commands, refer to Appendix A, "Control Utility Reference".

8.3.1.1 Starting a Names Server

To start a Names Server proceed as follows:

1. From the command line prompt type:

   ```
   NAMESCTL STARTUP
   ```

2. Or, from the NAMESCTL prompt, type:

   ```
   NAMESCTL> STARTUP
   ```

8.3.1.2 Testing a Names Server

To test a Names Server, use the NAMESCTL PING command. Following are two ways to PING the Names Server LABRADOR in the US.ACME domain.

1. From the NAMESCTL prompt, type:

   ```
   NAMESCTL> PING LABRADOR.US.ACME
   ```

2. Or, from the NAMESCTL prompt, type:

   ```
   NAMESCTL> SET SERVER LABRADOR.US.ACME
   NAMESCTL> PING
   ```

You can test several Names Servers with the same PING command. For example:

```
NAMESCTL>PING HUEY.UK.ACME DUEY.UK.ACME LOUIE.UK.ACME
```

PING responds with the time it takes to contact the Names Server and return an acknowledgment. If PING fails, make sure the Names Server is started or double-check the configured address of the Names Server.

8.3.1.3 Test Network Objects Using NAMESCTL

To determine whether your listener, service name, database link, or any other network object has successfully registered itself with or become known to the Names Server, use the QUERY command.

From the NAMESCTL prompt, type:

```
NAMESCTL> QUERY global_object_name type
```

Database service names have the type A.SMD, and database links have the type DL.RDBMS.OMD. The following example shows a query of the database service name BUGSY in the MACS.ACME domain.

```
NAMESCTL> QUERY BUGSY.MACS.ACME A.SMD
```

The QUERY command returns the amount of time the transaction took and information about the network object.

8.3.2 Using the Listener Control Utility (LSNRCTL)

The Listener Control Utility, LSNRCTL, is a tool that you run from the operating system prompt to start and control the listener. The general form of the Listener Control Utility is:

```
LSNRCTL command [listener_name] [args]
```

You can also issue Listener Control Utility commands at the program prompt. When you enter LSNRCTL on the command line, the program is opened. You can then enter the desired commands from the program prompt. For example, the following command determines the amount of time in seconds the listener will wait for a valid connection request after a connection has been started.

```
LSNRCTL> set connect_timeout 20
```

For more information about LSNRCTL and its commands, refer to Section A.1 in Appendix A, “Control Utility Reference”.

8.3.2.1 Starting a Listener

You may start a listener on any node where Net8 is installed. To start a listener, use the Listener Control Utility, LSNRCTL, to issue a START command as follows:

```
LSNRCTL START
```

To start a non-default listener:

```
LSNRCTL START listener_name
```

LSNRCTL will display a status message indicating that the listener has started successfully. Check that all expected SIDs for that listener are listed in the services summary in the status message.

8.3.2.2 Test a Listener

To test a listener, initiate a connection from a client to any active database controlled by that listener. If the only clients available to access the listener are on a different protocol, you must use an Oracle Connection Manager to access the listener.

8.3.3 Using the Connection Manager Control Utility (CMCTL)

The Connection Manager Control Utility, CMCTL, is a tool that you run from the operating system prompt to start and control Oracle Connection Manager. The general form of the Connection Manager Control Utility is:

```
CMCTL command
```

You can also issue CMCTL commands at the program prompt. When you enter CMCTL on the command line, the program is opened. You can then enter the desired commands from the program prompt. For example, the following command starts Oracle Connection Manager.

```
CMCTL> start
```

For more information about CMCTL and its commands, refer to Appendix A, “Control Utility Reference”.

8.3.3.1 Starting Oracle Connection Manager

You may start Oracle Connection Manager from any node where Net8 is installed. To start Oracle Connection Manager, use CMCTL to issue a START command as follows:

```
CMCTL START CMAN
```

CMCTL displays a status message indicating that Oracle Connection Manager has started successfully.

8.3.3.2 Testing Oracle Connection Manager

To test Oracle Connection Manager, initiate a connection from a client to any active database for which a source route address has been created.

8.3.4 Using TNSPING

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

If you can connect successfully from a client to a server (or a server to another server) using TNSPING, it displays an estimate of the round trip time (in milliseconds) it takes to reach the Net8 service.

If it fails, it displays a message describing the error that occurred. This allows you to see the network error that is occurring without the overhead of a database connection.

8.3.4.1 Starting TNSPING

To invoke the TNSPING utility, proceed as follows:

From the command line, type:

```
tnsping service_name [count]
```

Note: Different platforms may have different interfaces, but the program accepts the same arguments. Invoke TNSPING for the display of the proper interface requirements.

- service name: must exist in TNSNAMES.ORA, Oracle Names, or the name service in use, such as NIS or DCE's CDS.
- count (optional): determines how many times the program attempts to reach the server.

If the service name specified is a database name, TNSPING attempts to contact the corresponding network listener. It does not actually determine whether or not the database itself is running. Use Server Manager to attempt a connection to the database.

8.3.4.2 TNSPING Examples

Following are some examples of TNSPING. For example, to connect to a database named SPOTDB, the command:

```
tnsping spotdb
```

produces the following message:

```
TNS Ping Utility for SunOS:
Copyright (c) Oracle Corporation 1997.  All rights reserved.
Attempting to contact
(ADDRESS=(PROTOCOL=TCP)(HOST=spot)(PORT=1521))
OK (50msec)
```

To check whether a Names Server can be reached, use a command using the Net8 address as in the following:

```
tnsping (ADDRESS=(PROTOCOL=TCP)(HOST=fido)(PORT=1575))
```

A message similar to the following will be returned to the user:

```
TNS Ping Utility for SunOS:
Copyright (c) Oracle Corporation 1997. All rights reserved.
Attempting to contact (ADDRESS=(PROTOCOL=TCP)(HOST=fido)(PORT=1575))
OK (70 msec)
```

To determine whether the STPRD database can be connected to, and to specify that TNSPING try to connect 10 times and then give up, use the following command:

```
tnsping stprd 10
```

This command produces the following message:

```
TNS Ping Utility for SunOS:
Copyright (c) Oracle Corporation 1997. All rights reserved.
Attempting to contact (ADDRESS=(PROTOCOL=TCP)(HOST=spot)(PORT=1521))
OK (290 msec)
OK (100 msec)
OK (70 msec)
OK (70 msec)
OK (60 msec)
OK (70 msec)
OK (70 msec)
```

```
OK (80 msec)
OK (180 msec
OK (340 msec)
```

Below is an example of TNSPING attempting to connect to an invalid service name:

```
tnsping bad_db
```

This attempt produces the following message:

```
TNS Ping Utility for SunOS:
Copyright (c) Oracle Corporation 1997.  All rights reserved.
TNS-03505: Failed to resolve name
```

Following is an example of using TNSPING to connect to a name that is valid, but that resolves to an address where no listener is located (for example, the listener may not be started):

```
tnsping testing
```

The following message is returned:

```
TNS Ping Utility for SunOS:
Copyright (c) Oracle Corporation 1997.  All rights reserved.
Attempting to contact (ADDRESS=(PROTOCOL=tcp)(HOST=spot)(PORT=1521))
TNS-12541: TNS:no listener
```

8.3.5 Using TRCROUTE

The Trace Route Utility (TRCROUTE) enables administrators to discover what path or route a connection is taking from a client to a server. If TRCROUTE encounters a problem, 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, TRCROUTE collects error information that shows where the error occurred. The Trace Route Utility displays the information collected on the client screen. You can redirect the TRCROUTE output to a file, and print it if you wish.

8.3.5.1 Requirements

Trace Route works only over Net8 and SQL*Net version 2.3 and later. Every node along the route from client to server must use SQL*Net version 2.3 or later. If a pre-2.3 node is on the path, the following error is displayed:

```
TNS-03603:  Encountered a node with pre-2.3 version of SQL*Net
```

TRCROUTE shows what node along the path is responsible for any errors.

8.3.5.2 Effect on Performance

The Trace Route Utility 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 up any new processes or deal with dummy connections.

8.3.5.3 Starting the Trace Route Utility

To invoke TRCROUTE, type the following from the command line:

```
trcroute service_name
```

If you have configured your network to use listener load balancing, there may be more than one listener on different nodes for a database. If so, the Trace Route Utility might use any of the listeners, just as a regular connection request might. The output it returns shows you what listener node it used.

8.3.5.4 Examples of Trace Route Output

The following are two examples of trace route output:

- a successful Trace Route packet that traveled from a client to a listener
- an unsuccessful Trace Route packet that could not reach the listener because the listener was not up.

Example 8–1 Successful Trace Route

```
%trcroute tcp_direct
Trace Route Utility for Solaris:
Copyright (c) Oracle Corporation 1997.  All rights reserved.
```

```
Route of TRCROUTE:------------------
Node: Client            Time and address of entry into node:
-------------------------------------------------------------
01-DEC-96 13:26:36 ADDRESS= PROTOCOL=TCP  Host=shining-sun  Port=1581
Node: Server            Time and address of entry into node:
-------------------------------------------------------------
01-DEC-96 13:27:20 ADDRESS= PROTOCOL=TCP  Host=setting-sun  Port=1521
```

Example 8–2 Trace Route with Error

```
% trcroute tcp_direct

Trace Route Utility for SVR4:
Copyright (c) Oracle Corporation 1996.  All rights reserved.

Route of TRCROUTE:------------------
Node: Client            Time and address of entry into node:
-------------------------------------------------------------
01-DEC-96 11:12:34 ADDRESS= PROTOCOL=TCP  Host=shining-sun  Port=1581
TNS-12224: TNS:no listener
TNS-12541: TNS:no listener
TNS-12560: TNS:protocol adapter error
TNS-03601: Failed in route information collection
```

8.3.6 Testing a Client

To test several different clients in your network, initiate a connection to a server from each of them. If the connection is unsuccessful, use logging and tracing (including TRCROUTE), to find the cause of the problem.

There are a number of ways to initiate a connection to an Oracle server. Commonly used methods include:

- the operating system command line
- a tool logon screen
- a 3GL program
- special commands within certain tools

The specifics of use are slightly different in each case. Each of the general methods listed is briefly covered here. To identify the method used in a specific tool, refer to the tool's user's guide.

8.3.6.1 Connecting from the Operating System to Test a Client

The general form of connecting an application to a database server from the command line is:

```
tool username/password@service_name
```

To prevent the password from displaying during a logon, you can leave out the password parameter on the command line; you will be prompted to enter your password without it showing on screen.

Most Oracle tools can use the operating system command line to connect; some provide alternatives.

8.3.6.2 Connecting from the Tool Logon Screen to Test a Client

Some tools provide a logon screen as an alternative form of logon. A user can log on to a database server by identifying both the username and service name (*username@service_name*) in the username field of the tool logon screen, and typing the password as usual in the password field.

8.3.6.3 Connecting from 3GL to Test a Client

In applications written using 3GL, the program must establish a connection to a server using the following syntax:

```
EXEC SQL CONNECT :username IDENTIFIED BY :password
```

In this connection request, the :*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@service_name
```

The :*password* variable contains the password for the database account being connected to.

8.3.6.4 Connecting Using Special Commands within Tools

Some Oracle tools have commands for database connection, once the tool has been started, to allow an alternative username to be specified without leaving the tool. Both SQL*Plus and SQL*DBA allow the CONNECT command using the following syntax:

```
SQL> CONNECT username/password@service_name
```

For example:

```
SQL> CONNECT SCOTT/TIGER@SERVERX
```

This is very similar to the operating system command line method, except that it is entered in response to the tool prompt instead of the operating system prompt.

Other Oracle tools use slightly different methods specific to their function or interface. For example, Oracle CDE tools use logon buttons and a pop-up window with the username, password, and remote database ID field. For more information on connecting to Oracle with a specific tool, refer to the tool's user guide.

8.4 Checklist for Troubleshooting Common Startup Problems

The following checklist is provided to help you troubleshoot common problems you may encounter when starting Net8 components.

Table 8–1 Common Problems Encountered When Starting Net8 Components

Cause	Action
Inactive Components	**1.** Verify that you have installed a transport layer protocol and an Oracle Protocol Adapter. **2.** Verify that you have started a listener for any server you intend to contact. **3.** Start Oracle Connection Manager if you are routing sessions across protocols.
Syntax errors in your configuration files	**1.** Use the Oracle Net8 Assistant whenever possible to avoid syntax errors. If you have manually created or edited configuration files, check them carefully to ensure that all the appropriate parentheses are in place, that the lines are indented to show their logical structure, and that there are no typographical errors. For details on the syntax of these files, refer to the section titled, “Syntax Rules for Configuration Files” in Appendix B, “Configuration Parameters”. **2.** Verify that all service names are mapped to connect descriptors in any applicable local naming configuration file. **3.** Verify that an invalid listener name was not typed in the LSNRCTL START command. **4.** Check your typing. Verify that the listener name you are using matches the name specified in your listener configuration file (LISTENER.ORA).

Table 8–1 Common Problems Encountered When Starting Net8 Components

Cause	Action
Files are incorrectly placed.	**1.** The listener will indicate that it cannot start because configuration files could not be found. Normally, all configuration files are stored in $ORACLE_HOME/network/admin. However, the environment variable, TNS_ADMIN, can be set to point to a different location. If $TNS_ADMIN is set to a different directory, Net8 will expect your configuration files to exist in that directory. Secondly, Net8 looks in a "home" directory for configuration files. The "home" directory is different for each operating system. The best way to tell the location of the files is to look in your SQLNET.LOG file or in the header information in your trace file if you have asked for tracing to be turned on. **2.** If you are using Names Servers, verify that they have been started. Also make sure that your profile contains a preferred Names Server parameter so that a Names List file exists. **3.** If a native naming service such as NIS is in use, make sure that the appropriate Native Naming adapter has been installed on clients and servers, and that service names have been properly loaded into it. Refer to your operating system-specific documentation for information.
The address is already in use.	Another process may already be using the address listed in the listener configuration file (LISTENER.ORA). On some protocols such as TCP/IP, DECnet, and OSI, each network service on a node must use a unique port or socket. On other network protocols such as SPX/IPX or NetBIOS, each network service name must be unique for the entire network. Another network service may be using the same configuration. Contact your network administrator to evaluate whether the network address is available.

Table 8–1 Common Problems Encountered When Starting Net8 Components

Cause	Action
When trying to connect to a database, you may get the message `ORA-12203: "TNS:Unable to connect to destination".`	Use the Listener Control Utility (LSNRCTL) to start the listener.
When trying to make a connection from a client, you may get the message `ORA-12154:"TNS:Could not resolve service name".`	**1.** The service name you requested is not defined in your Oracle Names Server, Native Naming adapter, or local naming configuration file, or the local naming file can not be found as expected. **2.** If you are using Oracle Names, this problem may indicate a Names Server definition problem. Verify that your profile contains a NAMES.DIRECTORY_PATH parameter for a list of naming methods. **3.** Server is not running **4.** A Names Server List file does not exist. Issue a REORDER_NS command from NAMESCTL. **5.** The NAMES.PREFERRED_SERVER parameter is not configured correctly in a local profile. Verify that the parameter is configured correctly in the local profile.
When trying to connect to a database, you may get the message `ORA-1034: "Oracle Not Available"`, or `ORA-12505: "Listener could not resolve SID given in connect descriptor".`	The database is not running on the server machine. A listener alone does not provide a database connection; the database instance must also be started.

Table 8–1 Common Problems Encountered When Starting Net8 Components

Cause	Action
A client returns the message "ORA-12541: No Listener".	Connect requests that come in too quickly for a listener to handle, and which exceed the listener's backlog (determined by QUEUESIZE parameter in LISTENER.ORA and NAMES.ORA), are returned with an ECONNREFUSED error. A client encountering this error returns the message `"ORA-12541: No Listener"` and the client log or trace files will show the ECONNREFUSED message. To correct this problem, follow these steps: **1.** Stop the listener. **2.** Reconfigure QUEUESIZE in your listener (LISTENER.ORA), Oracle Connection Manager (CMAN.ORA), or Names Server (NAMES.ORA) configuration files to be a larger value (based on anticipated simultaneous connect requests). **3.** Restart the listener. **4.** Try to connect again.
When attempting to stop the listener, you may get the message `TNS-01169: "The listener has not recognized the password."`	Enter the SET PASSWORD command from within the Listener Control Utility (LSNRCTL), and then enter the STOP command to stop the listener.

9

Migrating to Net8

This chapter outlines migration paths and considerations for upgrading previous releases of SQL*Net version 2 to Net8. It includes the following section:

- Section 9.1, “Migrating from SQL*Net version 2”
- Section 9.2, “Why Migrate to Net8?”
- Section 9.3, “Considerations for Migrating to Oracle Names version 8”
- Section 9.4, “Using Oracle Connection Manager instead of Oracle MultiProtocol Interchange”
- Section 9.5, “Migration Scenarios”

9.1 Migrating from SQL*Net version 2

Net8 is backwards compatible with SQL*Net version 2 (release 2.0.14 and later). This means that Net8 clients can transparently connect to an Oracle7 database. It also means that existing SQL*Net version 2 clients can connect to an Oracle8 database.

Oracle Corporation no longer supports SQL*Net version 1.

9.2 Why Migrate to Net8?

Though it is not required, upgrading your network to Net8 is recommended for the following reasons:

- Minimal configuration. Net8 simplifies the process of setting up your network components. With Net8, you can start a client, network listener, Names Server, and Oracle Connection Manager with default settings. This minimizes the need to create and maintain configuration files. With Net8, the Oracle Net8 Assistant replaces most of the functionality previously provided with Oracle Network Manager. Use the Oracle Net8 Assistant to create or modify your existing local naming files, profile and Oracle Names configuration file.
- Extended network functionality. Net8 has replaced many of the features previously available with SQL*Net version 2 with equivalent or enhanced functionality.

  Table 9–1 lists the networking features supported in each Oracle release.

Table 9–1 Network Products Compatibility

Oracle Release	7.1.4	7.1.5	7.1.6	7.2.2	7.2.3	7.3.2	7.3.3	8.0.3
Net8								8.0.3
Oracle Connection Manager								8.0
Oracle Advanced Networking Option						2.3.2	2.3.3	8.0
SQL*Net	2.1.4	2.1.5	2.1.6	2.2.2	2.2.3	2.3.2	2.3.3	
Oracle Multiprotocol Interchange					2.0	2.0.2	2.0.3	
Secure Network Services	1.0.1	1.0.2	1.0.3	1.1	2.0			
Oracle Names	1.0	1.0	1.0	1.1	1.1	2.0.2	2.0.3	8.0

* The functionality of Multiprotocol Interchange is now included with Oracle Connection Manager. The functionality of Secure Network Services and SQL*Net/DCE are now included in the Oracle Advanced Networking Option.

9.3 Considerations for Migrating to Oracle Names version 8

Oracle Names version 8 is backward compatible with SQL*Net version 2. This implies that clients running on SQL*Net version 2 can access Names Servers using Oracle Names version 8 to connect to an Oracle8 server.

If you wish to take advantage of the new features provided with Oracle Names version 8, you must upgrade all of your existing Names Servers in a region to version 8 by installing Oracle Names version 8 on every existing Names Server node.

9.3.1 Migrating from Oracle Names version 2 using a Database

To upgrade and transfer data from an existing Names Server database to a version 8 database, run the namesupg.sql script on the node where Oracle Network Manager stored your network definition.

9.3.2 Migrating from Oracle Names version 2 using the Dynamic Discovery Option

If you were previously running Oracle Names version 2 using the Dynamic Discovery Option, and you want to configure a database as a repository for your Oracle Names information, you will need to:

1. Run the namesini.sql script on the Names Server node where you want the database to reside.
2. Use the Oracle Net8 Assistant to configure a NAMES.ADMIN_REGION parameter in every Names Server configuration file (NAMES.ORA).
3. Reload Names Server information into at least one Names Server by issuing a NAMESCTL REORDER_NS command on every listener node.

If you were previously running Oracle Names version 2 using the Dynamic Discovery Option, and you do not want to configure a database, you must issue a REORDER_NS command on every node.

9.3.3 Checklist for Ensuring Proper Migration to Oracle Names version 8

The following checklist is provided to ensure proper migration to Oracle Names version 8.

Table 9–2 Checklist for Ensuring Proper Migration to Oracle Names version 8

Task	Description
1. Upgrade all Names Servers in each region to the same version.	
2. Set up at least two Names Servers in each region to provide for fault tolerance.	
3. Change or delete the following parameters in any existing profile.	■ NAMES.PREFERRED_SERVERS. If you have already generated a Names Server Lists file or if one already exists, and you wish to continue using Names Servers discovered by it, you should delete or comment out the NAMES.PREFERRED_SERVERS parameter. Any Names Servers specified in the NAMES.PREFERRED_SERVERS parameter will override the results of the discovery process. ■ NAMES.DIRECTORY_PATH. If you have upgraded your network to Net8 and wish to use Oracle Names version 8, edit the NAMES.DIRECTORY_PATH parameter to specify that Oracle Names is the first naming method attempted. ■ NAMES.DEFAULT_DOMAIN. If you upgrade all or part of your network to Net8 and Oracle Names, you will need to decide whether to continue to use the .WORLD domain set by default. If you choose to use it, add the NAMES.DEFAULT_DOMAIN parameter to all new Net8 clients which will contact existing services. To maintain consistent naming conventions, be sure that the .WORLD domain is part of the names of the services that register themselves.

9.3.4 Other Obsolete Parameters

If you are upgrading your entire environment to Net8, you will not need the following configuration parameters:

- COMMUNITY
- NAMES.DEFAULT_ZONE
- NAME.PREFERRED_SERVERS

You may choose to delete these parameters or leave them. Though they are no longer required, they will not interfere with any network operations.

The COMMUNITY parameter used to a required part of all network service addresses. Thus, it appears anywhere you might find an address (for example, local naming and listener configuration files).

The NAMES.DEFAULT_ZONE and NAME.PREFERRED_SERVERS used to be included in profiles as slight variants of the NAMES.DEFAULT_DOMAIN and NAMES.PREFERRED_SERVERS parameters.

9.4 Using Oracle Connection Manager instead of Oracle MultiProtocol Interchange

If you have migrated your clients to Net8, and you still require multiprotocol support, you will need to install Oracle Connection Manager, and route your sessions through it. Oracle Multiprotocol Interchange is no longer supported past SQL*Net release 2.3.

To ensure proper functioning of Oracle Connection Manager, verify that you have deleted the following files on those nodes:

- TNSNET.ORA
- TNSNAV.ORA
- INTCHG.ORA

Other migration considerations are specific to your network configuration.

9.5 Migration Scenarios

The following migration scenarios are provided to help you understand the migration issues when:

- Migrating an existing Oracle7 Database to Oracle8
- Installing a new Oracle8 database in an existing Oracle7 network
- Migrating SQL*Net v2 clients to Net8
- Migrating to Oracle8 with Oracle Names

9.5.1 Migrating an existing Oracle7 Database to Oracle8

If you plan to upgrade an existing Oracle7 database to Oracle8, but do not wish to upgrade your SQL*Net v2 clients, proceed as follows:

1. Run a migration utility to install the Oracle8 tables on your server node.
2. Start the database.
3. Continue to use Oracle Network Manager to configure naming methods.

9.5.2 Installing a new Oracle8 database in an existing Oracle7 network

If you plan to install a new Oracle8 database in your existing Oracle7 network, and do not wish to upgrade your SQL*Net v2 clients, proceed as follows:

1. Install the Oracle8 database.
2. Start a network listener.
3. Start the Oracle8 database.
4. Continue to use Oracle Network Manager to configure naming methods.

9.5.3 Migrating SQL*Net v2 clients to Net8

If you plan to upgrade existing SQL*Net v2 clients to Net8 to connect to both Oracle7 and Oracle8 servers, proceed as follows:

1. Install Net8 on each client node.
2. Use the Oracle Net8 Assistant to modify any existing client profiles.

3. If you are using local naming, use the Oracle Net8 Assistant to modify your existing local naming configuration files to include new service addresses.

Note: If you are using the Oracle Net8 Assistant to configure default domains, keep in mind that "null" is the new default value. Oracle Network Manager appended a ".world" extension to every address by default.

9.5.4 Migrating to Oracle8 with Oracle Names

If you plan to upgrade both an existing Oracle7 database to Oracle8, and Oracle Names version 2 to version 8, proceed as follows:

1. Install Oracle8.
2. Install Oracle Names version 8 on each node with an existing Names Server.
3. If you were previously running Oracle Names version 2, and you want to update your database as a repository for your Oracle Names information, you will need to run the namesupg.sql script on the node where your network definition is stored.
4. If you were previously running Oracle Names version 2 using the Dynamic Discovery Option, and you want to configure a database as a repository for your Oracle Names information, you will need to:
   a. Run the namesini.sql script on the node where you wish to install the database.
   b. Use the Oracle Net8 Assistant to configure a NAMES.ADMIN_REGION parameter in every Names Server configuration file. For more information about the NAMES.ADMIN_REGION parameter, refer to Appendix B, "Configuration Parameters".
5. Verify that a well-known Names Server exists in your local region. If a well-known Names Server does not exist in your local region, you may need to configure the address of a preferred names server in your client profile.
6. Execute a REORDER_NS command from the Oracle Names Control Utility (NAMESCTL) on every node.
7. Delete or comment out any remaining parameters specifying a Preferred Names Server in all client profiles. This will ensure that each client will use the results of the discovery process to find a Names Server.

10

Troubleshooting Net8

Net8 provides methods for understanding and resolving network problems through the use of log and trace files. These files keep track of the interaction between network components as errors occur. Evaluating this information will help you to diagnose and troubleshoot even the most complex network problems.

This chapter describes common network errors and outlines procedures for resolving them. It also describes methods for logging and tracing error information to diagnose and troubleshoot more complex network problems. This chapter contains the following sections:

10.1 Troubleshooting Common Network Errors

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, applications such as SQL*Plus and SQL*Forms, which depend on network services from Net8, will normally generate an error message.

A list of the most common network error messages follows:

- ORA-12154 "TNS:could not resolve service name"
- ORA-12198 "TNS:could not find path to destination"
- ORA-12203 "TNS:unable to connect to destination"
- ORA-12224 "TNS:no listener"
- ORA-12500 "TNS:listener failed to start a dedicated server process"
- ORA-12533 "TNS:illegal ADDRESS parameters"
- ORA-12545 "TNS:name lookup failure"
- ORA-12560 "TNS:protocol adapter error"
- ORA- 3113 "TNS:End-of-file on communication channel"

Table 10–1 describes each network error and outlines procedures to troubleshoot them.

Table 10–1 Common Network Errors and Troubleshooting Procedures

Error #: Message	Description/Troubleshooting Procedures
ORA-12154: "TNS:could not resolve service name"	**Cause**: Net8 could not locate the service name specified in the TNSNAMES.ORA configuration file. **Actions**: **1.** Verify that a TNSNAMES.ORA file exists and that it is accessible. **2.** Verify that there are not multiple copies of the TNSNAMES.ORA file. **3.** In your TNSNAMES.ORA file, verify that the service name specified in your connect string is mapped to a connect descriptor in the TNSNAMES.ORA file. Also, verify that there are no syntax errors in the file. **4.** Verify that there are no duplicate copies of the SQLNET.ORA file. **5.** If you are using domain names, verify that your SQLNET.ORA file contains a NAMES.DEFAULT_DOMAIN parameter. If this parameter does not exist, you must specify the domain name in your connect string. If you are not using domain names, and this parameter exists, delete it or disable it by commenting it out. **6.** If you are connecting from a login box, verify that you are not placing an "@" symbol before your connect service name.

Table 10–1 Common Network Errors and Troubleshooting Procedures

Error #: Message	Description/Troubleshooting Procedures
ORA-12198: ‘‘TNS:could not find path to destination” and ORA-12203 ‘‘TNS:unable to connect to destination”	**Cause**: The client is not able to find the desired database. **Actions**: **1.** Verify that you have entered the service name you wish to reach correctly. **2.** Verify that the service name ADDRESS parameters in the connect descriptor of your TNSNAMES.ORA file are correct. **3.** Verify that your TNSNAMES.ORA file is stored in the correct directory. **4.** Verify that the listener on the remote node has started and is running. If not, start the listener by using the Listener Control Utility. **5.** If you are connecting from a login box, verify that you are not placing an “@” symbol before your connect service name.
ORA-12224:“TNS:no listener””	**Cause**: The connection request could not be completed because the listener is not running. **Actions**: **1.** Ensure that the supplied destination address matches one of the addresses used by the listener. **2.** Verify also that this is not a version compatibility problem.
ORA-12500: “TNS:listener failed to start a dedicated server process”	**Cause**: The listener was unable to start a process connecting the user to the database server. **Actions**: **1.** Verify that the SID_LIST section of the LISTENER.ORA file and the system identifier (SID) in the CONNECT DATA section of the TNSNAMES.ORA file are correct. **2.** Verify that the user has adequate privileges to access the database.
ORA-12533: “TNS:illegal ADDRESS parameters”	**Cause**: The protocol specific parameters in the ADDRESS section of the designated connect descriptor in your TNSNAMES.ORA file are incorrect. **Action**: For more information about protocol specific keywords, refer to the Oracle operating system specific documentation for your platform.

Table 10–1 Common Network Errors and Troubleshooting Procedures

Error #: Message	Description/Troubleshooting Procedures
ORA-12545: "TNS:name lookup failure"	**Cause**: The listener on the remote node cannot be contacted. **Actions**: 1. Verify that the ADDRESS in the TNSNAMES.ORA file or the LISTENER.ORA file is correct. 2. Verify that the listener on the remote node has been started. You may check its status with the STATUS command of the Listener Control Utility, and start it with the START command if necessary.
ORA-12560: "TNS:protocol adapter error"	**Cause**: The listener was unable to start a process connecting the user to the database server. **Actions**: 1. Turn on tracing and re-execute the operation. 2. Evaluate the contents of the trace file to diagnose the problem.
ORA-3113: "TNS:End of file on communication channel"	**Cause**: 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; it may indicate that the server has gone down. **Action**: You may need to modify your re-transmission count. For more information about troubleshooting this error, refer to the appropriate Oracle operating system specific documentation.

10.2 Troubleshooting Network Problems Using Log and Trace Files

Oracle Network Products provide detailed information about the source and context of problems as they arise. This information is generated and stored in log and trace files. The process of logging and tracing error information will help you to diagnose and resolve network problems.

10.3 Logging Error Information

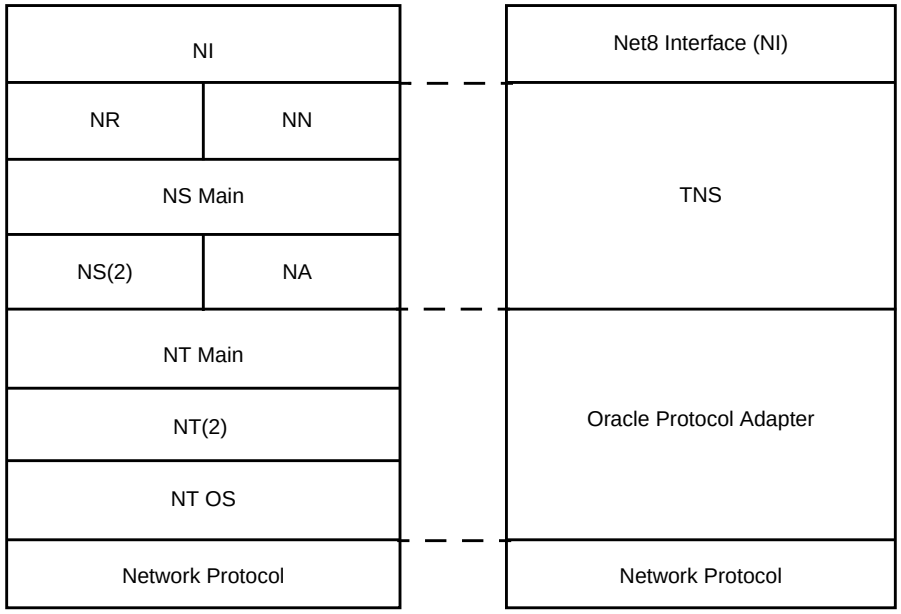
Logging refers to the process by which network components note and append error-specific information to a log file. Each Net8 component produces its own log file to describe the state of the software at various communication layers as an error occurs. To ensure that all errors are recorded, logging cannot be disabled on clients or Names Servers. Furthermore, only an administrator may replace or erase log files. The log file for the listener also includes Audit Trail information about every client connection request, as well as most listener control commands.

10.3.1 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.

Figure 10–1 depicts the relationship among Oracle network products as they might appear in an error stack:

Figure 10–1 Network Products and Error Stack Component



The layers in Figure 8-1 are as follows:

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

Your network may or may not include all of these components.

Error Example

As an example, suppose that a user of a client application tries to establish a connection with a database server using Net8 and TCP/IP, and the user enters:

```
sqlplus scott/tiger@hrserver.world
```

The SQL*Plus banner is displayed on the screen, and the following error is displayed:

```
ORA-12203: 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, a log file called SQLNET.LOG, contains an error stack corresponding to the ORA-12203 error as follows:

Example 10–1 Typical Error Stack

```
************************************************************

Fatal OSN connect error 12203, connecting to:
 (DESCRIPTION=(CONNECT_DATA=(SID=trace)(CID=(PROGRAM=)
   (HOST=lala)(USER=sviavant)))(ADDRESS_LIST=(ADDRESS=
   (PROTOCOL=ipc)(KEY=trace))(ADDRESS=(PROTOCOL=tcp)
   (HOST=lala)(PORT=1521))))

VERSION INFORMATION:
TNS for SunOS:
Oracle Bequeath NT Protocol Adapter for SunOS:
Unix Domain Socket IPC NT Protocol Adaptor for SunOS:
TCP/IP NT Protocol Adapter for SunOS:
  Tracing to file: /home/sviavant/trace_admin.trc
  Tns error struct:
    nr err code: 12203
    TNS-12203: 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
```

10.3.2 Log Filenames

Each Net8 component produces its own log file. Table 10–2 provides the default filenames and a description of the information they contain:

Table 10–2 Log File Component Information

Log File	Contains Error Information about the...
SQLNET.LOG	Client and / or server
LISTENER.LOG	Listener
NAMES.LOG	Names Server
CMAN.LOG	Oracle Connection Manager's main process
CMADM.LOG	Oracle Connection Manager's administration process

10.3.3 Setting Log Parameters

Parameters that control logging, including the type and amount of information logged, as well as the location where the files are stored, are set in the configuration file of each network component as follows:

Table 10–3 Setting Log Parameters

Log Parameters Corresponding to the...	...are set in the following Configuration Files
client	SQLNET.ORA
server	SQLNET.ORA
listener	LISTENER.ORA
Names Server	NAMES.ORA
Oracle Connection Manager	CMAN.ORA

Although there are operating system specific default names and locations for the log files, you can specify alternative names and locations if you wish. Each component may have log parameters that determine the following:

- the log file name (optional)
- the log file directory (optional)

10.3.3.1 Changing Log File Names

To change log file names, edit the following parameter in the appropriate component configuration file:

```
LOG_FILE_component = string
```

For example, the following parameter in your listener configuration file would send listener log output to a file called TEST.LOG on the server machine.

```
LOG_FILE_LISTENER = TEST
```

Note: On most operating systems, the *.log* suffix is automatically appended to the log filename. For a full description of configuration parameters, refer to Appendix B, "Configuration Parameters". Some platforms have restrictions on the properties of a filename. See your Oracle operating system-specific documentation for platform-specific restrictions.

10.3.3.2 Changing Log File Directories

To change the location of where the log file for each component is stored, edit the following parameter in the appropriate component configuration file:

```
LOG_DIRECTORY_component = valid directory
```

Examples are specific to different operating systems. An example on UNIX might be:

```
LOG_DIRECTORY_LISTENER = /tmp/log
```

Some platforms have restrictions on the properties of a directory. For more information about platform-specific restrictions, refer to your Oracle operating system-specific documentation.

10.3.4 Using Log Files

To use a log file to diagnose a network error:

1. Review the log file for the most recent error number you received from the application. Note that this is almost always the last entry in the log file.

2. Starting from the bottom of the file, locate the first non-zero entry in the error report. This is usually the actual cause.
3. If that error does not provide the desired information, review the next error in the stack until you locate the correct error information.
4. If the cause of the error is still not clear, turn on tracing and re-execute the statement that produced the error message.

10.3.5 Listener's Log Audit Trail

The listener log file contains Audit Trail information that allows you to gather and analyze network usage statistics, as well as information indicating the following:

- a client connection request
- a start, stop, status, reload or service command issued by the Listener Control Utility

Note that you cannot turn this feature off.

10.3.5.1 Format of the Listener's Log Audit Trail

The Audit Trail formats text into the following fields: *Timestamp, Connect Data, [Protocol Info], event, [sid], return Code*. Properties of the Audit Trail are as follows:

- Each field is delimited by an asterisk (*)
- PROTOCOL INFO and SID appear only when a connection is attempted
- A successful connection or command returns a code of 0
- A failure produces a code that maps to an error message

Typical output to the log file upon a reload request is as follows:

Example 10–2 Typical Audit Trail Information for Successful Reload Request

```
10-MAY-96 14:16:21 *(CONNECT_DATA=(CID=(PROGRAM=)(HOST=roach)(USER=reltest)
(COMMAND=reload)(ARGUMENTS=64)(SERVICE=LISTENER)
(VERSION=36704256))*reload*0
```

Typical output to the log file upon a connection request is as follows:

Example 10–3 Typical Audit Trail Information for Successful Connection Request

```
10-MAY-96 14:16:21*(CONNECT_DATA=(SID=reltest)(CID=
(PROGRAM=C:\ORAWIN\BIN\PLUS31.EXE)
(HOST=WINDOWSPC)(USER=CCLOW))*(ADDRESS=(PROTOCOL=tcp)
(HOST=144.25.23.246)(PORT=3366))
*establish*reltest*0
```

Notice that the user ID is recorded as well as the platform, protocol, and software used to make the connection.

10.3.5.2 Using Audit Trail Information

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

10.4 Tracing Error Information

The trace facility produces a detailed sequence of statements that describe network events as they are executed. “Tracing” an operation allows you to obtain more information on the internal operations of the components of Net8 than is provided in a log file. This information is output to files that can be evaluated to identify the events that led to an error.

CAUTION: The trace facility uses a large amount of disk space and may have a significant impact upon system performance. Therefore, you should enable tracing only when necessary

Components that can be traced using the trace facility are:

- Network listener
- Net8 components on the client and server
- Oracle Connection Manager components
- Oracle Names Server
- Oracle Names Control Utility
- TNSPING utility

10.4.1 Setting Tracing Parameters

To enable tracing as well as to set specific trace parameters, you may use either:

- component configuration files
- component control utilities
- Oracle Trace

10.4.1.1 Setting Trace Parameters Using Component Configuration Files

To set tracing parameters using component configuration files:

1. Specify the following parameters in the component configuration file:

   ```
   TRACE_LEVEL_component name=(OFF/USER/ADMIN/SUPPORT)
   TRACE_DIRECTORY_component name=directory name
   ```

2. If you modified the configuration files while the component was running, start or restart the component to enable the changed parameters.

10.4.1.2 Setting Trace Parameters Using Component Control Utilities

To set trace parameters using component control utilities:

- For the listener, use the TRACE command from the Listener Control Utility, to set the trace level while the listener is running;
- For Oracle Names, use the SET TRACE_LEVEL command from the Names Server Control Utility, to set the trace level while the Names Server is running.

> **Note:** For Oracle Connection Manager, the trace level can only be set from the Connection Manager configuration file.

10.4.1.3 Setting Trace Parameters Using Oracle Trace

Oracle Trace is a new tool that can be used with Oracle Enterprise Manager or stand alone enabling you to format and correlate data from two different trace files. Use Oracle Trace in situations where you will need to compare and / or correlate the trace information produced individually by Net8 and the server, or by Net8 as it interacts between the client and server. For more information on enabling Oracle Trace, refer to the *Net8 Release Notes*.

10.4.2 Evaluating Net8 Traces

Evaluating trace files either manually, or by using the Trace Assistant tool will help you to diagnose and troubleshoot network problems by giving you a better understanding of the following:

- the flow of packets between network nodes
- which component of Net8 is failing
- pertinent error codes

10.4.2.1 Understanding the Flow of Data Packets Between Network Nodes

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

10.4.2.1.1 Understanding 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.

Table 10–4 provides a list of the Net8 packet keywords and describes the types of packets they represent:

Table 10–4 Keyword and Packet Types

Keyword	Packet Type
NSPTCN	Connect
NSPTAC	Accept
NSPTRF	Refuse
NSPTRS	Resend
NSPDA	Data
NSPCNL	Control
NSPTMK	Marker

Note: This data is not viewable if you are using encryption through an Oracle network product or through EBCDIC data.

For example, the following line describes a procedure called “nscon” sending a NSPTCN packet over the network:

```
nscon: sending NSPTCN packet
```

Each packet has a keyword that denotes the packet type. All packet types begin with the prefix “NSP”. It is helpful to remember this when reviewing trace files for specific packet information

Example 10–4 provides typical packet information:

Example 10–4 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  |ap207sun|
nspsend:29 28 55 53 45 52 3D 6D  |)(USER=m|
nspsend:77 61 72 72 65 6E 29 29  |warren))|
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=ap34|
nspsend:37 73 75 6E 29 28 50 4F  |7sun)(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)
```

10.4.2.2 Understanding Pertinent Error Output

Every time a problem occurs with the connection in Net8, the error code is logged in the trace file with the prefix of **<ERROR>** or **<FATAL>**. Example 10–5 depicts typical trace file error output.

Example 10–5 Trace File Error Output

```
nspsend: entry
```

```
nspsend: plen=244, type=6
ntpwr: entry
ntpwr: exit
-<ERROR>- nspsend: transport write error
nspsend: error exit
nserror: entry
-<ERROR>- nserror: nsres: id=0, op=65, ns=12541, ns2=12560; nt[0]=511,
nt[1]=61,nt[2]=0
-<ERROR>- nsopen: unable to open transport
nricdt: Call failed...
nricdt: exit
-<ERROR>- osnqper:  error from nricall
-<ERROR>- osnqper:  nr err code: 12203
-<ERROR>- osnqper:  ns main err code: 12541
-<ERROR>- osnqper:  ns (2)  err code: 12560
-<ERROR>- osnqper:  nt main err code: 511
-<ERROR>- osnqper:  nt (2)  err code: 61
-<ERROR>- osnqper:  nt OS   err code: 0
osnqme: entry
osnqme:  reporting nr (1) error: (12203) as rdbms err (12203)
osnqme: exit
-<ERROR>- onstns: Couldn't connect, returning 12203
nricall: Exiting NRICALL with following termination result -1
nricall: exit
osnqme: entry
osnqme:  reporting nr (1) error: (12203) as rdbms err (12203)
osnqme: exit
-<ERROR>- onstns: Couldn't connect, returning 12203
-<ERROR>- osnqper:  error from nricall
```

The most efficient way to evaluate error codes is to find the most recent NS error code logged. This is because 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 your connection.

For information about the specific return codes, use the Oracle UNIX error tool "oerr". Use the "oerr" tool to discover more information about Net8 return codes, by entering the following at any command line prompt:

```
oerr tns error_number
```

10.4.3 Using the Trace Assistant to Examine Your Trace Files

Net8 provides a tool called the Trace Assistant to help you understand the information provided in your trace files by converting existing lines of trace file text into a more readable paragraph. Note that the Trace Assistant runs against only a **level 16 (SUPPORT)** SQL*Net or Net8 trace file.

To run the Trace Assistant, type the following at any command line prompt:

```
trcasst [options] filename
```

Table 10–5 describes the options that are available.

Table 10–5 Trace Assistant Text Formatting Options

Option	Description
-o	Displays connectivity and Two Task Common (TTC) information. After the -o the following options may be used: ■ c (for summary connectivity information) ■ d (for detailed connectivity information) ■ u (for summary TTC information) ■ t (for detailed TTC information) ■ q (displays SQL commands enhancing summary TTC information)
-p	Oracle Internal Use Only
-s	Displays statistical information
-e	Enables display of error information After the -e, zero or one 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)

If no options are provided, then the default is -odt -e -s, providing detailed connectivity, detailed Two-Task Common, error level 0, and statistics.

Example 10–6 depicts how Trace Assistant converts trace file information into a more readable format.

Example 10–6 Typical Trace Assistant Conversion

Trace File	Converted by Trace Assistant with option -e0 or -e1
nsc2addr: normal exit nsopen: entry nsmal: 404 bytes at 0x10d5a48 nsopen: opening transport... -<ERROR>- ntus2err: sd=13, op=1, resnt[0]=511, resnt[1]=2, resnt[2]=0 -<ERROR>- nserror: nsres: id=0, op=65, ns=12541, ns2=12560; nt[0]=511, nt[1]=2, nt[2]=0 -<ERROR>- nsopen: unable to open transport	Error found. Error Stack follows: id: 00000 Operation code: 00065 NS Error 1: 12541 NS Error 2: 12560 NT Generic Error: 00511 Protocol Error: 00146 OS Error: 00000 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.

However, other errors may also exist within the trace file that were not logged from the nserror function.

10.4.3.1 Understanding Information Traversing the Network in Net8 Packets

Trace Assistant also allows you to view data packets from both the Net8 and Two Task Common communication layers. Trace Assistant offers you two options to view these packets:

- summary connectivity (using option -oc)
- detailed connectivity (using option -od)

Net8 Packet Examples

The following examples depict how Trace Assistant presents various packets as they are sent to and from the Net8 layer in a variety of transactions:

- bequeathed connection
- redirected connection
- data packet

Note that 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 server. The connection request is always sent by the client, but received by the server (or listener).

Example 10–7 Summary Data Packets Sent in a Bequeathed Connection

Using trcasst -oc <filename>

```
---> Send     192 bytes - Connect packet
       Connect data length: 142
(DESCRIPTION=(ADDRESS=(PROTOCOL=TCP)(Host=dlsun)(Port=1521))(CONNECT_DATA=(SID=db1)(CID=(PROGRAM=)(HO
ST=dlsun)(USER=use1))))
<---  Received 24 bytes - Accept packet
       Accept data length: 0
```

This example shows two packets. The first is the connect packet that is sent from the client to the listener. The second is the accept packet coming back from the server.

Example 10–8 Detailed Data Packets Sent in a Bequeathed Connection

Using trcasst -od <filename>

```
--->  Send     50 bytes - Connect packet
Current NS version number is: 309.
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: 2048
        Maximum TDU size: 5120
        NT protocol characteristics:
                Test for more data
                Spawner is running
                Hang on to Listener connection
                Full duplex I/O
                Urgent data support
                Generate SIGURG signal
                Handoff connection to another
        Line turnaround value: 0
        Connect data length: 234
        Connect data offset: 50
        Connect data maximum size: 2048
                Native Services wanted
                Native Services wanted
Cross facility item 1: 0
        Cross facility item 2: 0
        Connection id: Ox0000000000000000
        Packet data is in the following data packet
--->  Send     244 bytes - Data packet
(DESCRIPTION=(ADDRESS=(PROTOCOL=beq)(PROGRAM=/private/oracle/bin/
oracle)(ARGV0=oracle)(ARGS='(DESCRIPTION=(LOCAL=YES)
(ADDRESS=(PROTOCOL=beq))))(DETACH=NO))(CONNECT_DATA=(CID=(PROGRAM=)(HOST=dlsun)(USER=use1))))
<---  Received 24 bytes - Accept packet
        Accepted NS version number is: 307.
Global options for the connection:
        no attention processing
        Don't care
        Accepted maximum SDU size: 2048
        Accepted maximum TDU size: 4096
        Connect data length: 0
                Native Services wanted
                Native Services wanted
```

This example shows all of the details sent along with the connect data in negotiating a connection.

Example 10–9 Summary Data Packets Sent in a Redirected Connection

Using trcasst -oc <filename>

```
--->  Send     187 bytes - Connect packet
        Connect data length: 153
(DESCRIPTION=(CONNECT_DATA=(SID=ap347db1)(CID=(PROGRAM=)(HOST=apsun)(USER=use2)))(ADDRESS_LIST=(ADDRE
SS=(PROTOCOL=tcp)(HOST=apsun)(PORT=1521))))
<---  Received 8 bytes - Resend packet
--->  Send     187 bytes - Connect packet
        Connect data length: 153
(DESCRIPTION=(CONNECT_DATA=(SID=apdb1)(CID=(PROGRAM=)(HOST=apsun)(USER=use2)))(ADDRESS_LIST=(ADDRESS=
(PROTOCOL=tcp)(HOST=apsun)(PORT=1521))))
<---  Received 24 bytes - Accept packet
        Accept data length: 0
```

Example 10–10 Data Packet

Using trcasst -oc <filename> or -od <filename>

```
--->  Send     30 bytes - Data packet
<---  Received 201 bytes - Data packet
--->  Send     439 bytes - Data packet
<---  Received 400 bytes - Data packet
```

Once the connection is established, data is given to Net8 from the Two-Task Common layer to be sent to the other node. Both summary and detailed views yield the same summary information.

Two Task Common Packet Examples

Two-Task Common handles requests such as open cursor, select rows, and update rows that are directed to the database. All requests are answered by the server. If you request to logon, a response is returned from the database that the request was completed. Example 10–11 and Example 10–12 show the type of information you can expect.

Summary information for Two-Task Common 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 Two-Task Common operates.

Example 10–11 Two Task Common Summary Information

Using trcasst -ou <filename>

```
(O3LOGA)        1st half of challenge-response logon                  80      78
(O3LOGON)       2nd half of challenge-response logon                  97      59
(OOPEN)         # 1                                                  21      16
(OPARSEX)       # 1                                                  245      59
(OCLOSE)        # 1                                                  17      11
(OVERSION)                                                           29      16
(OOPEN)         # 2                                                   21      16
(OALL7)         # 2 Parse Can Defn=2 Exec Fetch “SELECT A.V          268     100
(OOPEN)         # 3                                                  21      16
(OALL7)         # 3 Parse Exec=1 “SELECT USER FROM SYS.DUAL          152     70
(OALL7)         # 3 Defn=1 Fetc                                      117     88
(OCLOSE)        # 3                                                  17      11
```

On each line, the first item displayed is the actual request made.The second item is a cursor number, if one is involved with the transaction. The third item is either a listing of the flags or the SQL command that is being answered. The flag indicates that a request has the following characteristics:

!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

The number of bytes sent and received are displayed at the far right.

The OOPEN on line three is a prime example of how the output displays the request/response pairs. The OOPEN appears with a #1 following it indicating that an “Open cursor” request was sent from the client and the server responded with the cursor number 1 that it opened. Because a request/response pairing is placed on one line, you should not combine this option with any of the connectivity options.

Example 10–12 Two Task Common Summary Information

Using trcasst -ot <filename>

```
start of user function (TTIFUN)
        1st half of challenge-response logon (O3LOGA)
          Username: applsys
          Terminal: ttyp5
          Machine: ap207sun
          System User: mwarren
          Process: 24459
          Program: aiap45@ap207sun (TNS interface)
return opi parameter (TTIRPA)
        OPI parameter: 3309B1A977A62A3C
start of user function (TTIFUN)
        2nd half of challenge-response logon (O3LOGON)
          Username: applsys
          Terminal: ttyp5
          Machine: ap207sun
          System User: mwarren
          Process: 24459
          Program: aiap45@ap207sun (TNS interface)
ORACLE function complete (TTIOER)
start of user function (TTIFUN)
        Open a cursor
return opi parameter (TTIRPA)
        Cursor #: 1
start of user function (TTIFUN)
        Parse and Execute (OPARSEX) Cursor # 1
alter session set nls_language= 'AMERICAN' nls_territory= 'AMERICA' nls_currency= '$'
nls_iso_currency= 'AMERICA' nls_numeric_characters= '.,' nls_date_format= 'DD-MON-YY'
nls_date_language= 'AMERICAN' nls_sort= 'BINARY'
ORACLE function complete (TTIOER)
start of user function (TTIFUN)
        Close cursor (OCLOSE) Cursor # 1
V6 Oracle func complete (TTISTA)
        Succeeded
```

Example 10–13 Detailed SQL information on top of summary Two-Task

Using trcasst -ouq <filename>

(O3LOGA)	1st half of challenge-response logon	180	78
(O3LOGON)	2nd half of challenge-response logon	197	59
(OOPEN)	# 1	21	16

Add q to your summary Two-Task Command to display the detailed SQL information given automatically in the detailed Two-Task option.

Example 10–13 Detailed SQL information on top of summary Two-Task

Using trcasst -ouq <filename>

(OPARSEX)	# 1 alter session set nls_language= 'AMERICAN' nls_territory= 'AMERICA' nls_currency= '$' nls_iso_currency= 'AMERICA' nls_numeric_characters= '.,' nls_date_format= 'DD-MON-YY' nls_date_language= 'AMERICAN' nls_sort= 'BINARY'	245	59
(OCLOSE)	# 1	17	11
(O71SESOPN)	(get session ID)	47	18
(OOPEN)	# 1	21	16
(OVERSION)	Oracle7 Server Release 7.1.4.1.0 - Production Release with the distributed and parallel query optionsPL/SQL Release 2.1.4.0.0 - Production	29	157
(O71SESOPN)	(get session ID)	47	18

Add q to your summary Two-Task Command to display the detailed SQL information given automatically in the detailed Two-Task option.

10.4.3.2 Analyze the Data Collected into Appropriate Statistics

The type of statistics gathered is on the order of how many calls (TTC), packets and bytes were sent and received between the network partners. The following example depicts typical trace file statistics:

Example 10–14 Typical Trace File Statistics

Using trcasst -s <*filename*>

```
================================================================================
     Trace File Statistics:
    ------------------------------------------------------------------------
     SQL*Net:
          Total Calls:       466 sent,       491 received,       423 upi
          Total Bytes:    119214 sent,     86614 received
       Average Bytes:       255 sent,       176 received
       Maximum Bytes:      2048 sent,      2048 received
     GRAND TOTAL PACKETS  sent:    466     received:    491
```

10.4.3.3 Example of a Trace File

The following example shows a full trace file decoded. This example was created using the Oracle client tool SVRMGRL with the request:

```
connect scott/tiger@june
```

The message `ORA-12154: TNS:could not resolve service name` was displayed on the screen.

Example 10–15 Trace File Example

Description	Trace File Information

Note Trace level and location of the trace file in the Trace Configuration Information section.	`-- TRACE CONFIGURATION INFORMATION FOLLOWS ---`
	`New trace stream is "C:\ORAWIN\network\trace\sqlnet7.trc"`
	`New trace level is 16`
	`--- TRACE CONFIGURATION INFORMATION ENDS ---`

The Network Names component cannot find service name "june.world". Note client adds ".world" extension to service name "june".	`nnfotran: tnsname.ora entry for name "june.world" not found`
	`nnftqnm: Error querying june.world of attribute A.SMD`
	`errcode 406`
	`nnfgrwsp: Query unsuccessful, skipping to next adapter`

Client attempts to access a Names Server (oranamesrvr0) to resolve service name address.	`nnfgrwsp: Switching to ONAMES adapter`
	`nnfgrwsp: Original name: june`
	`nnfgrwsp: Qualified name: june.world`
	`nngsget_get_stream: looking for`
	`"(DESCRIPTION=(CONNECT_DATA=(RPC=ON))(ADDRESS=(PROTOCOL=tcp)(`
	`HOST=oranamesrvr0)(PORT=1575)))"`
	`nngsget_get_stream: cache miss, opening new stream`
	`nngsnad_new_stream_addr: "(DESCRIP`
	`TION=(CONNECT_DATA=(RPC=ON))(ADDRESS=(PROTOCOL`
	`=tcp)(HOST=oranamesrvr0)(PORT=1575)))"`
	`nngsget_get_stream: no caller address will be sent to callee`

Example 10–15 Trace File Example

Description	Trace File Information
Network Routing (nr) performs routing to Names Server (oranamesrvr0).	nricall: entry nric2a: entry nric2a: Getting local community information nriglp: entry nriglp: Looking for local addresses setup by nrigla nriglp: No addresses in the preferred address list nriglp: exit nric2a: TNSNAV.ORA is not present. No local communities entry. nrigla: entry nrigla: Getting local address information nrigla: Simple address... nrigla: No community component so just use straight address nrigla: exit nridst: entry nridst: Resolving address to use to call destination or next hop nridst: Found destination address nridst: Local address nridst: Local destination community found nridst: exit nric2a: This is a local community access nric2a: exit nricall: Got routable address information nricall: Making call with following address information: (DESCRIPTION=(CONNECT_DATA=(RPC=ON))(ADDRESS=(PROTOCOL=tcp)(HOST=oranamesrvr0)(PORT=1575))) nricdt: entry nricdt: Calling with outgoing connect data (DESCRIPTION=(CONNECT_DATA=(RPC=ON))(ADDRESS=(PROTOCOL=tcp)(HOST=oranamesrvr0)(PORT=1575)))
Network Session (ns) sets up the session to the Name Server.	nscall: entry nscall: connecting... nsc2addr: entry nsc2addr: (DESCRIPTION=(CONNECT_DATA=(RPC=ON))(ADDRESS=(PROTOCOL=tcp)(HOST=oranamesrvr0)(PORT=1575)))

Example 10–15 Trace File Example

Description	Trace File Information
Network Transport (nt) sets up the transport session.	nttbnd2addr: entry nttbnd2addr: port resolved to 1575 nttbnd2addr: looking up IP addr for host: oranamesrvr0 nttbnd2addr: exitnsopen: entry nsmal: entry nsmal: 330 bytes at 0x30d76e74 nsmal: normal exit nsopen: opening transport... nttcon: entry nttcon: toc = 1 nttcnp: entry nttcnp: creating a socket. nttcnp: exit nttcni: entry nttcni: trying to connect to socket 1. ntt2err: entry
Network Transport (nt) returns the error "no listener" as the Names Server is not running.	-<ERROR>- ntt2err: soc 1 error - operation=1, ntresnt[0]=511, ntresnt[1]=61 ntresnt[2]=0 ntt2err: exit nttcni: exit nttcon: exit nserror: entry
The error is propagated to the next layer (ns).	-<ERROR>- nserror: nsres: id=0, op=65, ns=12541, ns2=12560; nt[0]=511, nt[1]=61,nt[2]=0 -<ERROR>- nsopen: unable to open transport nsmfr: entry nsmfr: 330 bytes at 0x30d76e74 nsmfr: normal exit nsopen: error exit nscall: error exit nricdt: Call failed... nricfg: entry nricfg: exit nricdt: Call made to destination nricdt: exit

Example 10–15 Trace File Example

Description	Trace File Information
	nricall: Failed to copy originating community name value binding nricall: Exiting NRICALL with following termination result -1 nricall: exit nngsfad_free_stream_addr: "(DESCRIPTION=(CONNECT_DATA=(RPC=ON))(ADDRESS=(PROTOCOL=tcp)(HOST=oranamesrvr0)(PORT=1575)))" -<ERROR>- nngsget_get_stream: open failure, error stack follows
The errors are propagated to the next layer (TNS)	TNS-12224: TNS:no listener TNS-12541: TNS:no listener TNS-12560: TNS:protocol adapter error TNS-00511: No listener nnfgrwsp: Query unsuccessful, skipping to next adapter
The address is not found on any Names Server as no Names Server is available.	nnfun2a: address for name "june" not found nngsfad_free_stream_addr: "(DESCRIPTION = (CONNECT_DATA=(RPC=ON)) (ADDRESS=(PROTOCL=tcp) (HOST=oranamesrvr0) (PORT=1575)))" nngtdei_deinit_msg: free message pool block nngtfms_free_msg: message ID -10429 nngtfms_free_msg: message was a request nngtfms_free_msg: message free, type 100 nngtfoa_free_objarr: free message object array nngtfmt_free_msg_type: type-specific message free, type 100 nngtfoa_free_objarr: free message object array nngtfms_free_msg: message ID 0 nngtfms_free_msg: message was a response nngtfms_free_msg: message free, type 0 nngsdei_deinit_streams: deinit nngscls_close_stream: UID 11 not established, ignored nngscls_close_stream: UID 0 not established, ignored osnqrn: Return code from nnfsn2a is 409
Error is returned to the user.	-<ERROR>- onstns: Couldn't connect, returning 12154 onstns: exit osnqtg: Count in the OSN global area is now 0 rigbd: entry nrigbd: exit osnqtg: Count in the NL global area is now 0

Trace File Example Summary

This trace file provides a summary of what occurs with Net8 when you encounter the error “Could not resolve service name”. In this example, a client is unsuccessful in making a connection to service name “june”. This is because a “names.default_domain = world” parameter exists in the profile. This parameter adds the “.world” extension to all service names requested, including the service name “june”. Unfortunately, this service name is defined in neither the client’s local naming configuration file, nor a Names Server. To troubleshoot this problem, the user should:

- edit the profile to remove the NAMES.DEFAULT_DOMAIN configuration parameter; or
- request a connection to “june.world” instead of “june”

10.5 Contacting Oracle Customer Support

If you are still unable to resolve your problems or if you are requested to contact Oracle Customer Support to report the error, please have the following information at hand:

- The hardware, operating system, and release number of the operating system on which your application(s) is running.

  Example: "My client application runs on a Sun workstation running SUN OS4.1.3 and the server application runs on a VAX machine running VMS version 5.4. The protocol is TCP / IP."

- The release numbers of all the Oracle networking products involved in the current problem.
- If you encountered one or more error codes or messages, the exact code numbers and message texts in the order they appeared.
- The problem severity according to the following codes:

  1 = Program not usable. Critical impact 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.
- A description of the problem, including any unusual conditions.
- You will also be expected to provide your:
  - name
  - company's name
  - company's Oracle customer support ID (CSI) number
  - phone number

11

Net8 Enhancements for Programmers

Net8 includes an application program interface (API) called Net8 OPEN allowing programmers to develop both database and non-database applications. In addition, Net8 contains several new benefits for programmers including UNIX client programming, signal handler and alarm programming, Bequeath Adapter and child process termination.

This chapter contains the following sections:

11.1 Net8 OPEN

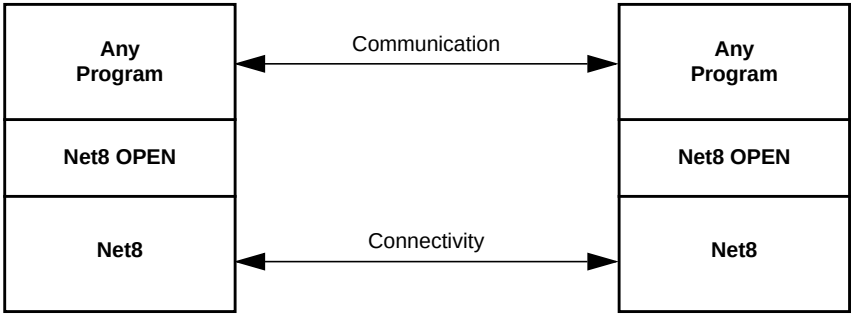
Net8 includes an application program interface (API) called Net8 OPEN, which enables programmers to:

- develop both database and non-database applications that make use of the Net8 network already deployed in their environment
- deploy an application developed on one machine to another without having to modify their calls to the network interface

Net8 OPEN provides applications a single common interface to all industry standard network protocols.

The relationship of Net8 OPEN to other products is shown in Figure 11–1.

Figure 11–1 Net8 OPEN



Using Net8 OPEN, you can solve a number of problems, such as:

- Three-tier connectivity (client / agent / server) - Use any application to communicate with an agent. For example, the agent might be an application server that allows simultaneous connectivity to Oracle and non-Oracle data sources, such as remote information servers.
- Distributed applications - Build distributed applications that can run over an existing Oracle network without the requirements of a database or additional middleware.
- Enhanced clients - Integrate non-SQL information with SQL applications. For example, a process control application can communicate with a non-SQL application such as a sensor.

11.1.1 Net8 OPEN API Function Calls

In contrast to a remote procedure call interface, Net8 OPEN provides a byte stream-oriented API that can be used to develop basic applications which send and receive data. Applications developed with Net8 OPEN must ensure that values sent across the network are interpreted correctly at the receiving end.

The Net8 OPEN API consists of five function calls:

- TNSOPEN
- TNSCLOSE
- TNSSEND
- TNSRECV
- TNSCONTROL

Table 11–1 Net8 OPEN API Function Call Summary

TNSOPEN	
Description:	Initializes the Net8 OPEN API per-connection handle. This function must be the first Net8 OPEN call that a user makes. Note that tnsopen() does not establish a connection; connections are established by the send and receive calls as needed. If you are writing a client program (which will initiate the connection), "name" contains a service name in the same format as those in a local naming configuration file. If you are writing a server program, "name" should be NULL. Your server program will pick up the connection automatically when you make the first tnsrecv() call to receive data (see the section on tnsrecv).

Synopsis:	`int tnsopen(handlep, name)`
	`void **handlep;`
	`const char *name;`

Parameters:	`handlep (IN/OUT)`
	- Address to receive Net8 connection handle
	`name (IN)`
	- Service name

Prerequisites:	The handlep parameter must not be NULL
Returns:	Upon successful completion a zero value is returned. Otherwise, a positive Net8 API error is returned.

TNSCLOSE	
Description:	Shuts down the connection. The user must call this function to close the connection and release the handle properly.
Synopsis:	`int tnsclose(handlep)` `void **handlep;`
Parameters:	`handlep(IN/OUT)` - Address of a pointer to a Net8 connection handle
Prerequisites:	The handlep parameter must not be NULL.
Returns:	Upon successful completion a zero value is returned, and *handlep is set to NULL. Otherwise, a positive Net8 API error number is returned.

TNSSEND	
Description:	Sends data to the Net8 connection handle. In the first call to tnssend() on the client side, the connection is established before any data is sent to the handle. The client application must first call tnssend() after tnsopen() to establish a connection to the server. It is an error if the client application calls tnsrecv() first, or if the server program calls tnssend() first. Note that this also means that the tnssend() call may return errors related to connection establishment - so the first indication you get that, for instance, you have given the incorrect TNS address, is that an error occurs on the first tnssend() call, rather than on the tnsopen() call as you may first expect.

Synopsis:	`int tnssend(handle, data, length)`
	`void *handle;`
	`const void *data;`
	`size_t *length;`

Parameters:	`handle(IN/OUT) - pointer to Net8 connection handle returned by tnsopen()`
	`data(IN) - pointer to data to be sent`
	`length(IN/OUT) - pointer to the length of data to be sent in bytes and the actual number of bytes written on return.`

Prerequisites:	The parameters must not be NULL.
Returns:	Upon successful completion a zero value is returned, and the actual number of bytes written is returned as the value pointed to by the length parameter. Otherwise, a positive Net8 API error number is returned.

TNSRECV

Description:	Receives data from the Net8 connection handle. In the first call to tnsrecv() on the server side, the connection is established before data is received from the handle. The server must first call tnsrecv() after tnsopen() to accept the connection from the client. Incoming connections are accepted by the Net8 Listener (tnslsnr), which automatically spawns off a copy of your server program when needed, and hands it the new connection. You must configure the network listener with the location of your server program for this to occur — see the section on configuration below.

Synopsis:	`int tnsrecv(handle, data, length)`
	`void *handle;`
	`void *data;`
	`size_t *length;`

Parameters:	`handle(IN/OUT) - pointer to Net8 connection handle returned by tnsopen()`
	`data(IN/OUT) - pointer to buffer to receive data`
	`length(IN/OUT) - pointer to the length of buffer to receive data and actual number of bytes received on return`

Prerequisites:	All parameters must not be NULL.
Returns:	Upon successful completion a zero value is returned, and the actual number of bytes received is returned as the value pointed to by the length parameter. Otherwise, a positive Net8 API error number is returned.

TNSCONTROL	
Description:	Sets the connection to blocking or nonblocking mode.

Synopsis:	`int tnscontrol(handle, cmd)`
	`void *handle;`
	`int *cmd;`

Parameters:	`handle(IN) - pointer to Net8 connection handle returned by tnsopen()`
	`cmd(IN) - option to apply to the connection. Currently supported values are:`
	`TNSAPINONBLOCKING - set connection into nonblocking mode`
	`TNSAPIBLOCKING - set connection into blocking mode`

Prerequisites:	The handle must not be NULL, and cmd must be one of the supported commands.
Returns:	A zero value is returned if the option is successfully set.

11.1.2 Finding the Net8 OPEN Applications Program Interface

The applications program interface is provided as part of the standard Net8 installation. To use it, you need the following:

- TNSAPI.H - this is the header file which describes the API interfaces and errors. This file is located in the Oracle home directory for your platform. On UNIX, it is provided in $ORACLE_HOME/network/public.
- The Net8 OPEN library - located with other Oracle networking libraries, and contains the name "TNSAPI". Note that the name of the library varies by platform. On UNIX, it is in your $ORACLE_HOME/network/lib directory and is named libtnsapi.a. On Windows platforms, the Oracle directories contain the files TNSAPI.DLL and TNSAPI.LIB.
- Sample makefiles - are provided for your platform in your network directory. They can be used to determine the appropriate link line to build your application.

11.1.3 Building Your Own Application

Modules which make reference to Net8 OPEN functions should include TNSAPI.H, as follows:

```
#include <tnsapi.h>
```

Your makefile (or other location for your build command) should ensure that the include path is set properly so that it can find TNSAPI.H. Refer to the sample makefiles provided in your installation.

11.1.4 Configuring the System to Use Your Net8 OPEN Application

To configure Net8 to recognize your Net8 OPEN application, proceed as follows:

1. Add the location of your server program to your listener configuration file (LISTENER.ORA), so that the network listener knows to start your server if a connection request is received.

   To do this, choose a system identifier (SID) name for your service similar to that of an Oracle database. Do not pick the same SID as your database.

   For example, if you are configuring a "chat" program, you could call the SID "chatsid". Place the program into the same place as the Oracle server executable, which is normally $ORACLE_HOME/bin, which we will assume here is in /usr/oracle/bin.

You would place the following entry in a listener configuration file as follows:

```
SID_LIST_LISTENER =
  (SID_LIST =
    (SID_DESC =
        (SID_NAME = chatsid)/*your SID name*/(ORACLE_HOME = /usr/oracle/
        bin)/*$ORACLE_HOME bin directory*/(PROGRAM = chatsvr)/*the name of
        your server program*/)
```

You need to restart the listener, so it will recognize the new service.

2. Add the address of your application server to your local names configuration file (TNSNAMES.ORA).

   For example, if your listener is listening on the following address:

   ```
   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=unixhost)(PORT=1521)))
   ```

   And you want people to refer to the service you created above as ”chat”.

   You would add the following parameter to your local naming configuration file:

   ```
   chat=
   (DESCRIPTION=
       (ADDRESS=(PROTOCOL=tcp)(HOST=unixhost)(PORT=1521)
       )
       (CONNECT_DATA=(SID=chatsid)
       )
   )
   ```

   Note that the address contains the SID you configured in the LISTENER.ORA file above. Also note that the second line started with at least one space character, which indicates that it is a continuation line.

   If you have domains in your network, you need to name your service accordingly. For instance, use chat.acme.com if the domain is acme.com. Again, use the existing local naming configuration file as a template — if all the other service names end in a domain, you need to name your service similarly.

3. Place the executable for your service in the same directory as your Oracle Server executable. On UNIX platforms, place the executable in the $ORACLE_HOME/bin directory indicated in your LISTENER.ORA file. In this example, you would place the program ”chatsvr” in the location /usr/oracle/bin/chatsvr.

If needed on your operating system, you also must ensure that you have permission to execute your program.

11.1.5 Sample Programs

Two sample applications are provided with Net8 OPEN:

- finger - this is a utility that connects to the server that retrieves information about who is logged in. This utility includes a pair of programs which demonstrate the basic steps involved in building a distributed application. The client program runs on both Solaris and Windows NT; the server is UNIX specific.
- tftp - this sample client and server program is implemented in UNIX to help you with simple file transfers using the tftp protocol.

11.1.6 Net8 OPEN API Errors

This section lists the error numbers which can be returned if one of the above function calls fails. Note that in some cases, connection-related errors may come back from a send or receive call, if the connection has not yet been established at that time.

```
20002 - SDFAIL_TNSAPIE - The underlying "send" command failed in tnssend().
20003 - RECVFAIL_TNSAPIE - The underlying "receive" command failed in tnsrecv().
20004 - INVSVROP_TNSAPIE - Operation is invalid as the server.
20005 - INVCLIOP_TNSAPIE - Operation is invalid as the client.
20006 - HDLUNINI_TNSAPIE - The connection should be initialized by calling
tnsopen().
20007 - INHFAIL_TNSAPIE - Server failed in inheriting the connection from the
listener.
20008 - ACPTFAIL_TNSAPIE - Server failed in accepting the connection request
from the client.
20009 - NULHDL_TNSAPIE - A null handle was passed into the call, which is not
allowed.
20010 - INVOP_TNSAPIE - An invalid operation called was passed into the call.
20011 - MALFAIL_TNSAPIE - A malloc failed in TNS API call.
20012 - NLINIFAIL_TNSAPIE - Failed in NL initialization.
20013 - NMTOOLONG_TNSAPIE - Service name is too long.
20014 - CONFAIL_TNSAPIE - Client connect request failed.
20015 - LSNFAIL_TNSAPIE - Server failed to listen for connect request.
20016 - ANSFAIL_TNSAPIE - Server failed to answer connect request.
20017 - NMRESFAIL_TNSAPIE - Failed to resolve service name.
20018 - WOULDBLOCK_TNSAPIE - Operation would block.
20019 - CTLFAIL_TNSAPIE - Control call failed.
20020 - TNSAPIE_ERROR - TNS error occurred.
```

```
20021 - INVCTL_TNSAPIE - Invalid operation request in control call.
```

11.2 UNIX Client Programming

Event programming in UNIX requires the use of a UNIX signal. When an event occurs, a signal flags a process. The process executes code that is relevant to the particular signal generated. UNIX does not allow a single process to set more than one signal handler or alarm for a particular signal call. If a process sets a second signal handler or alarm request on a signal like SIGCHLD (signal on a child process' status change), UNIX nullifies and loses the previous request for the SIGCHLD.

If any part of your application issues one of these requests, signal handling or alarms may cause the system to lose and never respond to that particular request. Depending on the signal requested, the system may not clean up defunct processes properly because of a signal handler problem.

Net8 provides two solutions to allow for the use of signal handling and alarms in tandem with Oracle's usage of those requests:

- Signal Handler and Alarm Programming
- Bequeath Adapter

11.2.1 Signal Handler and Alarm Programming

Net8 provides an operating system dependent (OSD) call that keeps a table of all signal handler or alarm requests for each signal. Any program that uses the signal handler or alarm is now required to use the Oracle OSD calls. This provides a solution for programmers in UNIX who are not allowed to set more than one signal handler or alarm for a particular call. Any program that uses the signal handler or alarm must use the Oracle OSD calls. This is however, currently available only for internal use. In the near future, an externalized version of the OSD calls for client application usage will be released.

Until then, if you set all of the client's signal handlers before making any database connections, the OSD call will remember the last signal handler set for the signal and will add it to the signal handler table. Note that by doing this, you cannot disable the signal handler.

11.2.1.1 Oracle OSD Signal Handling Rules

To use the table-driven shared OSD signal handler for all SIGCHLD calls, you must observe the following rules:

- Know your child process IDs so you can clean up the correct process.
- Use the `waitpid`() call instead of `wait`() on the correct child process ID.
- The `waitpid`() call must be non-blocking.

11.2.2 Bequeath Adapter

This section is for UNIX application programmers who use both the UNIX signal handler for tracking child process status changes with the SIGCHLD call and Net8 for the networking portion of their application.

When a client application is directed to communicate with an Oracle database on the same machine, it uses Net8's Bequeath Adapter to establish the connection. The Bequeath Adapter enables the client to retrieve information from the database without using the network Listener. The Bequeath Adapter internally spawns a server process for each client application. In a sense, it performs locally the same operation that a remote listener does for your connection.

11.2.2.1 Child Process Termination

Since the client application spawns a server process internally through the Bequeath Adapter as a child process, 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 defunct process. Signal handlers are responsible for cleaning up these defunct processes. Alternatively, you may configure your client profile to pass this process to the UNIX init process by disabling signal handlers.

Use the Oracle Net8 Assistant to configure a client to disable the UNIX signal handler. The profile parameter set to disable is as follows:

```
BEQUEATH_DETACH=YES
```

This parameter causes all child processes to be passed over to the UNIX init process (pid = 1). The init process automatically checks for "defunct" child processes and terminates them.

The Bequeath Adapter automatically chooses to use a signal handler in tracking child process status changes. If your application does not use any signal handling, then this default does not affect you.

12

Extending Net8 Functionality

Several related Oracle products extend Net8 functionality. Those products include:

- Section 12.1, “Oracle Enterprise Manager”
- Section 12.2, “Oracle Advanced Networking Option”
- Section 12.3, “Oracle Security Server”

12.1 Oracle Enterprise Manager

Oracle Enterprise Manager is included with the Oracle8 Server. It combines a graphical console, agents, common services, and tools to provide an integrated, comprehensive systems management platform for managing Oracle products. From the Oracle Enterprise Manager console you can:

- administer, diagnose, and tune multiple databases
- distribute software to multiple servers and clients
- schedule jobs on multiple nodes at varying time intervals
- monitor objects and events throughout the network
- integrate participating third-party tools

Oracle Enterprise Manager requires SQL*Net or Net8 to communicate across the network. In turn, some Net8 features are controlled by Oracle Enterprise Manager including Simple Network Management Protocol (SNMP) Support.

SNMP Support

Support for the Simple Network Management Protocol (SNMP) allows a database and other network objects to be remotely monitored by any SNMP-capable management software in a TCP/IP network. SNMP is a de facto standard underlying many popular network management systems such as Hewlett Packard's OpenView, Novell's Network Management System, IBM's NetView/6000, and Sun Solstice. It enables Oracle products such as the Net8 network listener, Oracle8 Server, and Names Servers to be located, identified, and monitored by a management station running at one or more centrally located nodes.

For more information about SNMP Support and the information it provides, refer to the *Oracle SNMP Support Reference Guide*.

12.2 Oracle Advanced Networking Option

Oracle Advanced Networking Option is an optional product that works with Net8 and SQL*Net release 2.1.4 and later. It includes the following features:

- Security Services
- Authentication Services
- Oracle DCE Integration

Security Services

Oracle Advanced Networking Option enables Net8 and related products to use network data encryption and checksumming so that data cannot be read or altered. It protects data from unauthorized viewing by using the RSA Data Security RC4™ or the Data Encryption Standard (DES) encryption algorithm. To ensure that data has not been modified, deleted, or replayed during transmission, the security services of Oracle Advanced Networking Option can generate a cryptographically secure message digest and include it with each packet sent across the network.

Oracle Advanced Networking Option is supported by Oracle Connection Manager. Clients and servers using different protocols can securely transfer data across network protocol boundaries. For example, clients using LAN protocols such as NetWare (SPX / IPX) can share data securely with large servers using different protocols such as LU6.2, TCP / IP, or DECnet.

Authentication Services

Oracle Advanced Networking Option includes enhanced user authentication services such as support for single sign-on. These authentication services enhance the existing security facilities of Oracle7 and Oracle8 such as secure network access control logon, roles, and auditing by providing reliable user identification. No changes to applications are required. Oracle Advanced Networking Option works over all protocols, operating systems, and name services. It also supports token authentication through Security Dynamics ACE Server, Kerberos, and DCE Security Server, and biometrics authentication through Identix.

These services are available to most products that implement Net8, including the Oracle8 Server, Developer 2000 tools, and any other Oracle or third-party products that support Net8.

Oracle DCE Integration

Oracle Distributed Computing Environment (DCE) Integration is an optional product that works with Net8 and SQL*Net 2.1.6 and later. It enables users to transparently use Oracle tools and applications to access Oracle7 and Oracle8 Servers in a DCE environment. It provides authenticated RPC (Remote Procedure Call) as the transport mechanism, which enables multi-vendor interoperability. The DCE security service enables a user logged onto DCE to securely access any Oracle application without having to specify a username or password. This is sometimes referred to as "external authentication," formerly referred to as "OPS$ support".

Oracle DCE Integration also provides support for DCE Cell Directory Service (CDS), which allows Oracle7 and Oracle8 services to be transparently accessed throughout the DCE environment. Users can connect to Oracle database servers in a DCE environment using familiar Oracle service names. Oracle service names can be managed from a central location with standard DCE tools. For more information, refer to the *Oracle Advanced Networking Option Administrator's Guide* and your Oracle platform-specific documentation.

12.3 Oracle Security Server

The Oracle Security Server is included with the Oracle8 Server. Oracle Security Server is a security product based on public-key cryptography that supports authentication and authorization in an Oracle networking environment. It requires Net8 functionality for communication between its major modules.

Oracle Security Server features:

- a global centralized authentication and authorization framework that is based on public-key cryptography. It contains two modules: Oracle Security Adapter and the Oracle Security Repository. This framework supports X.509 certificates, an industry standard method of authentication.
- the Oracle Security Manager, a management tool that configures the framework.
- the Oracle Cryptographic Toolkit, a programmer's toolkit that accesses the authentication and authorization resources provided by the Oracle Security Server framework. This toolkit is a set of application programming interfaces (APIs) that are available in two forms: Oracle Call Interfaces (OCI) and PL/SQL code. These APIs enable application programs to access cryptographic functions, such as generating and validating digital signatures. They provide assurance to a wide variety of user applications, for example, electronic mail and electronic commerce. This toolkit is accessible to both Oracle and non-Ora-

cle applications on Oracle networks, and to Oracle applications on non-Oracle networks, such as the World Wide Web and the Internet.

For more information on Oracle Security Server, refer to the *Oracle Security Server Guide*. For more information on the Oracle Cryptographic Toolkit, refer to the *Oracle Cryptographic Toolkit Programmer's Guide*.

A

Control Utility Reference

Net8 provides you with utilities through which you can control each networking component. This appendix describes the control utilities for the listener, Oracle Names and Oracle Connection Manager. It also lists the commands that are available with each utility, including any applicable prerequisites, passwords, syntax or argument rules, usage notes or examples to help you use them.

The three control utilities described in this appendix are:

- Section A.1, “Listener Control Utility (LSNRCTL)”
- Section A.2, “Oracle Names Control Utility (NAMESCTL)”
- Section A.3, “Connection Manager Control Utility (CMCTL)”

A.1 Listener Control Utility (LSNRCTL)

The Listener Control Utility (LSNRCTL)is a tool that you run from the operating system prompt to start and control the listener. The general syntax of the Listener Control Utility is as follows:

```
LSNRCTL command [listener_name]
```

You can also issue Listener Control Utility commands at the program prompt. When you enter LSNRCTL on the command line, the program is opened. You can then enter the desired commands from the program prompt.

A.1.1 LSNRCTL Commands

The following commands are available through the Listener Control Utility (LSNRCTL).

CHANGE_PASSWORD	
Purpose:	This command allows you to dynamically change the password of a listener. This will not change unencrypted passwords already established in a listener configuration file. It only establishes a new password, or changes a password that has been encrypted in the listener configuration file.
Prerequisites:	None
Password Required:	Yes
Syntax:	`change_password [listener_name]`
Arguments:	listener_name
Usage Notes:	The control utility prompts you for your old password, then for the new one. It asks you to re-enter the new one, then changes it. Neither the old nor the new password displays during this procedure.

Example:	`LSNRCTL> change_password`
	`Old password:`
	`New password:`
	`Reenter new password:`
	`Connecting to (ADDRESS=(PROTOCOL=ipc)(KEY=iris))`
	`Password changed for LISTENER`
	`The command completed successfully`

DBSNMP_START

Purpose:	This command starts the SNMP subagent for an Oracle database running on the same node.
Prerequisites:	Must be run locally
Password Required:	No
Syntax:	`LSNRCTL> dbsnmp_start`
Arguments:	None
Usage Notes:	None
Example:	`LSNRCTL> dbsnmp_start`

DBSNMP_STATUS

Purpose:	This command verifies whether the SNMP subagent for an Oracle database is running.
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> dbsnmp_status`
Arguments:	None
Usage Notes:	DBSNMP STATUS must be run on the same node the Oracle database is on.
Example:	`LSNRCTL> dbsnmp_status`

DBSNMP_STOP	
Purpose:	This command stops the SNMP subagent for an Oracle database running on the same node.
Prerequisites:	Must be run locally
Password Required:	No
Syntax:	`LSNRCTL> dbsnmp_stop`
Arguments:	None
Usage Notes:	None
Example:	`LSNRCTL> dbsnmp_stop`

EXIT	
Purpose:	This command quits LSNRCTL and returns you to the operating system prompt.
Prerequisites:	None
Password Required:	Yes
Syntax:	`LSNRCTL> EXIT`
Arguments:	None
Usage Notes:	This command is identical to QUIT
Example:	`LSNRCTL> EXIT`

HELP

Purpose:	Provides a list of all the LSNRCTL commands available
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> help`
Arguments:	None

Example:	`LSNRCTL> help`
	`The following operations are available`
	`An asterisk (*) denotes a modifier or extended command:`
	`start stop status`
	`services version reload`
	`trace spawn dbsnmp_start`
	`dbsnmp_stop dbsnmp_status change_password`
	`quit exit set* show*`

QUIT

Purpose:	This command quits LSNRCTL and returns you to the operating system prompt
Prerequisites:	None
Password Required:	Yes
Syntax:	`LSNRCTL> QUIT`
Arguments:	None
Usage Notes:	This command is identical to EXIT
Example:	`LSNRCTL> QUIT`

RELOAD

Purpose:	This command shuts down everything except listener addresses, and re-reads the LISTENER.ORA file. This command enables you to add or change services without actually stopping the listener.
Prerequisites:	This will not work on valid nodes. In this case, the listener must be stopped and restarted.
Password Required:	Yes
Syntax:	`LSNRCTL> reload [`*`listener name`*`]`
Arguments:	Name of listener
Usage Notes:	If there are any passwords in the listener configuration file, you must use the SET PASSWORD command before you can use the RELOAD command. You must set the password from within the LSNRCTL program; you cannot set it from the operating system command line. The method for setting the password depends on whether you are using the encrypted password feature. If you are *not* using an encrypted password, enter the password on the LSNRCTL command line.
Example:	`LSNRCTL> reload`

SAVE_CONFIG

Purpose:	This command creates a backup of your listener configuration file (called LISTENER.BAK) and updates the actual configuration file (LISTENER.ORA) itself to reflect any changes.
Password Required:	Yes
Syntax:	`LSNRCTL> save_config [`*`listener name`*`]`
Arguments:	Name of listener
Usage Notes:	This is used by an administrator to save all on-line configuration changes to the listener configuration file.
Example:	`LSNRCTL> SAVE_CONFIG LISTENER`

SERVICES

Purpose:	This command provides detailed information about the services the listener listens for. For example, how many connections have been established, how many refused. It displays three different types of services (dedicated servers from LISTENER.ORA, dispatcher information, and prespawned shadows.)
Prerequisites:	None
Password Required:	Yes
Syntax:	`LSNRCTL> services [`*`listener name`*`]`
Arguments:	Name of listener
Usage Notes:	This is used by a database administrator to get information about the services of the listener.
Example:	`LSNRCTL> SERVICES [`*`listener name`*`]` The output of a LSNRCTL SERVICES command follows:

```
LSNRCTL for SunOS: Version 2.1.3.0.0 -
Production on 10-FEB-94 07:14:55
Copyright (c) Oracle Corporation 1993.  All
rights reserved.
Connecting to
(ADDRESS=(PROTOCOL=IPC)(KEY=ruth))
Services Summary...
ruth has 1 service handlers
DEDICATED SERVER established:99 refused:0
```

SET

Purpose:	This command lists the parameter values that can be set using the SET command.
Prerequisites:	None
Password Required:	Yes
Syntax:	`LSNRCTL> set`
Arguments:	None
Usage Notes:	You must have set a valid password to be able to use this command if one is listed in the listener configuration parameter, PASSWORDS_*listener_name*. If there are any passwords in the listener configuration file, you must use the SET PASSWORD command before you can use the SET command. You must set the password from within the LSNRCTL program; you cannot set it from the operating system command line. The method for setting the password depends on whether you are using the encrypted password feature. If you are *not* using an encrypted password, enter the password on the LSNRCTL command line.
Example:	`LSNRCTL> set` The following operations are available after set An asterisk (*) denotes a modifier or extended command: password trc_file trc_directory trc_level log_file log_directory log_status current_listener connect_timeout save_config_on_stop startup_waittime use_plugandplay

SET CONNECT_TIMEOUT

Purpose:	This command determines the amount of time the listener will wait for a valid connection request after a connection has been started.
Prerequisites:	None
Password Required:	No
Syntax:	LSNRCTL> set connect_timeout *time*
Arguments:	*time in seconds*
Usage Notes:	None
Example:	LSNRCTL> set connect_timeout 20 Connecting to (ADDRESS=(PROTOCOL=ipc)(KEY=iris)) LISTENER parameter ”connect_timeout” set to 20 The command completed successfully

SET CURRENT_LISTENER

Purpose:	This command sets or shows parameters for multiple listeners.
Prerequisites:	You must enter SET CURRENT_LISTENER from within the LSNRCTL utility
Password Required:	No
Syntax:	`LSNRCTL> set current_listener [`*`listener name`*`]`
Arguments:	[*listener name*]
Default Argument	LISTENER
Usage Notes:	If there is more than one listener on a node, any LSNRCTL command acts on the default listener (LISTENER) unless another listener has been set. Any subsequent LSNRCTL commands within the same LSNRCTL session would then apply to the second listener, unless `CURRENT_LISTENER` were reset. You can also display the current listener by using the LSNRCTL `SHOW CURRENT_LISTENER` command. You must enter SET CURRENT_LISTENER from within the LSNRCTL utility. When you exit the utility, the setting will be lost.
Example:	`LSNRCTL> set current_listener [`*`listener name`*`]`

SET LOG_DIRECTORY

Purpose:	This command allows you to change the default directory where log files for the listener process are written.
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> set log_directory [`*`directory`*`]`
Arguments:	[*directory*]

Example:	`LSNRCTL> set log_directory /usr/oracle/admin`
	`Connecting to`
	`(ADDRESS=(PROTOCOL=ipc)(KEY=iris))LISTENER`
	`parameter "log_directory" set to /usr/oracle/admin`
	`The command completed successfully`

SET LOG_FILE

Purpose:	This command sets a non-default name for the log file
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> set log_file [filename]`
Arguments:	[filename]

Example:	`LSNRCTL> set log_file list.log`
	`Connecting to`
	`(ADDRESS=(PROTOCOL=ipc)(KEY=iris))LISTENER`
	`parameter "log_file" set to list.log`
	`The command completed successfully`

SET LOG_STATUS

Purpose:	This command turn listener logging on or off
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> set log_status [ON | OFF]`
Arguments:	ON or OFF
Example:	`LSNRCTL> set log_status on`

SET PASSWORD

Purpose:	This command changes the password sent from the LSNRCTL utility to the listener process for authentication purposes only. To change the password on the listener itself, use the change_password command.
Password Required:	Yes
Syntax:	`LSNRCTL> set password`
Arguments:	[*password*]
Usage Notes:	You may enter this command when you start up the shell or any time during your session. (You must enter the SET PASSWORD command before you can stop the listener.) The preferred, secure way to enter your password is in interactive mode. The listener supports encrypted and unencrypted passwords.
Example:	`LSNRCTL> SET PASSWORD` `enter listener password: password`

SET SAVE_CONFIG_ON_STOP

Purpose:	This command saves any changes made by the LSNRCTL SET command permanently if parameter is ON. The saving of all parameters occurs right before the listener exits. To have all parameters saved right away, use the SAVE_CONFIG command.
Password Required:	Yes
Syntax:	`LSNRCTL> set save_config_on_stop [ON | OFF]`
Arguments:	ON or OFF
Example:	`LSNRCTL> set save_config_on_stop on`

SET STARTUP_WAITTIME

Purpose:	This command sets the amount of time the listener sleeps before responding to a START command:
Prerequisites:	None
Password Required:	No
Syntax:	LSNRCTL> set startup_waittime *[time]*
Arguments:	*time in seconds*
Example:	LSNRCTL> set startup_waittime 10 Connecting to (ADDRESS=(PROTOCOL=ipc)(KEY=iris))LISTENER parameter "startup_waittime" set to 10 The command completed successfully

SET TRC_DIRECTORY

Purpose:	This command allows you to change the default location where trace files for the listener process will be written.
Prerequisites:	None
Password Required:	No
Syntax:	LSNRCTL> set trc_directory [*directory*]
Arguments:	[*directory*]
Example:	LSNRCTL> set trc_directory /usr/oracle/admin Connecting to (ADDRESS=(PROTOCOL=ipc)(KEY=iris)) LISTENER parameter "trc_directory" set to /usr/oracle/admin The command completed successfully

SET TRC_FILE

Purpose:	This command sets a non-default name for the trace file
Prerequisites:	None
Password Required:	No
Syntax:	LSNRCTL> set trc_file [*filename*]
Arguments:	[*filename*]
Example:	LSNRCTL> set trc_file list.trc Connecting to (ADDRESS=(PROTOCOL=ipc)(KEY=iris)) LISTENER parameter "trc_file" set to list.trc The command completed successfully

SET TRC_LEVEL

Purpose:	This command turns on tracing for the listener.
Prerequisites:	None
Password Required:	Yes
Syntax:	LSNRCTL> SET TRC_LEVEL *level*
Arguments:	OFF, USER, ADMIN, SUPPORT or 0, 4, 10, 16.
Usage Notes:	Selecting USER provides a limited level of tracing; ADMIN provides a more detailed trace. This command overrides the setting in the LISTENER.ORA file. You must have set a valid password, if one is listed in the LISTENER.ORA file parameter PASSWORDS_*listener_name* to be able to use this command This command is identical to TRACE.
Example:	LSNRCTL> SET TRC_LEVEL ADMIN

SET_USE_PLUG_AND_PLAY	
Purpose:	This command instructs the listener to register with a well-known Names Server. The listener will continue to look for a well-known Names Server until one is found.
Password Required:	Yes
Syntax:	`LSNRCTL> set use_plug_and_play [ON | OFF]`
Arguments:	ON or OFF
Example:	`LSNRCTL> set use_plug_and_play on`

SHOW

Purpose:	All of the SET commands listed except `SET PASSWORD` have equivalent SHOW commands. In response to one of the SHOW commands, LSNRCTL displays the current setting of the listener for that parameter.
Prerequisites:	None
Password Required:	see equivalent SET commands
Syntax:	`LSNRCTL> show [`*`listener name`*`] subcommand`
Arguments:	[*listener name*] subcommand
Usage Notes:	The SHOW parameter can be shown, but not set, through LSNRCTL: `SHOW [`*`listener name`*`] SNMP_VISIBLE` displays whether the listener is accessible to SNMP clients
Example:	`LSNRCTL> show` The following operations are available after show An asterisk (*) denotes a modifier or extended command: ■ current_listener ■ connect_timeout ■ log_file ■ log_directory ■ log_status ■ save_config_on_stop ■ snmp_visible ■ startup_waittime ■ trc_file ■ trc_directory ■ trc_level ■ use_plugandplay

SPAWN	
Purpose:	The SPAWN command starts a program stored on the machine on which the listener runs, and which is listed with an alias in the LISTENER.ORA file.
Prerequisites:	None
Password Required:	No
Syntax:	LSNRCTL> spawn [*listener name*] *alias* (ARGUMENTS=*arg1,arg2,*)
Arguments:	[*listener name*] - listener name, if any *alias* - the alias of the program as listed in the listener configuration file *arg1* - the arguments sent to the program that is to be spawned
Example:	LSNRCTL> spawn nstest_alias (ARGUMENTS='')

START

Purpose:	This command starts the named listener.
Prerequisites:	Listener must be stopped
Password Required:	No
Syntax:	`LSNRCTL> start [`*`listener name`*`]`
Arguments:	Name of the listener. If no listener name is entered, LISTENER is started by default.

Usage Notes:	To start a listener configured in the LISTENER.ORA file with a name other than LISTENER, include that name.
	For example, if the listener name is TCP_LSNR, enter:
	`LSNRCTL START TCP_LSNR`
	Or, from the LSNRCTL program prompt, enter:
	`LSNRCTL> START TCP_LSNR`

Example:	`LSNRCTL> start listener`

STATUS

Purpose:	This command displays basic information: version, start time, uptime, what LISTENER.ORA file is used, and whether tracing is turned on.
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> status [`*`listener name`*`]`
Arguments:	Name of listener.
Usage Notes:	The status command allows you to perform the following: ■ check the current setting of the logging and tracing options. ■ the list of database SIDs available through this listener. These are defined in the SID mapping in LISTENER.ORA. ■ whether a password is encrypted in LISTENER.ORA. (If you encrypt the listener password you can have only one password.) ■ whether the network listener can respond to queries from an SNMP-based network management system ■ the address(es) the TNSLSNR is listening on

Example:	`LSNRCTL> status [`*`listener name`*`]`
	`LSNRCTL for SunOS:`
	`Copyright (c) Oracle Corporation 1997. All rights reserved.`
	`Connecting to`
	`(ADDRESS=(PROTOCOL=IPC)(HOST=orchid)(port=1334))`
	`STATUS of the LISTENER`
	`------------------------`
	`Alias LISTENER`
	`Version TNSLSNR for SunOS:`
	`Start Date 10-FEB-97 07:06:34`
	`Uptime 0 days 0 hr. 0 min. 44 sec`
	`Trace Level ADMIN`
	`Security ON`
	`...`
	`The command completed successfully`

STOP

Purpose:	This command stops the named listener.
Prerequisites:	The listener must be running
Password Required:	Yes
Syntax:	`LSNRCTL> stop [`*`listener name`*`]`
Arguments:	Name of listener.
Usage Notes:	If you have configured passwords, you must use the SET PASSWORD command before you can use the STOP command. You must set the password from within the LSNRCTL program; you cannot set it from the operating system command line. The method for setting the password depends on whether you are using the encrypted password feature. If you are *not* using an encrypted password, enter the password on the LSNRCTL command line. Be careful when stopping a listener. On some platforms and with some protocols, when a listener is stopped any Net8 connections currently running are shut down. In some situations the connections continue, but it is then not possible to start the listener again until the running processes have been closed. It is good practice to send a warning message to all network users before stopping a listener.
Example:	`LSNRCTL> stop listener`

TRACE

Purpose:	This command turns on tracing for the listener.
Prerequisites:	valid password required
Password Required:	Yes
Syntax:	`LSNRCTL> trace [OFF|USER|ADMIN|SUPPORT] [listener name]`
Arguments:	[OFF | USER | ADMIN | SUPPORT] [listener name]
Usage Notes:	USER provides a limited level of tracing. ADMIN provides a more detailed trace. This command overrides the setting in the LISTENER.ORA file. This command has the same functionality as SET TRC_LEVEL.
Example:	`LSNRCTL> trace admin listener`

VERSION

Purpose:	Displays the current TNS listener, and protocol adapter version.
Prerequisites:	None
Password Required:	No
Syntax:	`LSNRCTL> version [`*`listener name`*`]`
Arguments:	Name of service
Example:	`LSNRCTL> version listener`

A.2 Oracle Names Control Utility (NAMESCTL)

The Oracle Names Control Utility (NAMESCTL) is a tool that you run from the operating system prompt to start and control Names Servers. It contains several types of commands:

- Operational commands such as STARTUP, SHUTDOWN, RESTART, and so forth.
- Modifier commands, such as SET *property*.
- Informational commands, such as STATUS, SHOW *property*, and PING.
- Command utility operational commands such as EXIT, QUIT, and HELP.

You can use the any of these utilities to perform basic management functions on one or more Names Servers. By using this tool, you can execute such commands as STARTUP, SHUTDOWN, and STATUS. Additionally, you can view and change Names Server parameter settings such as RESET_STATS_INTERVAL and TRACE_LEVEL.

A.2.1 NAMESCTL Operating Modes

You can run NAMESCTL in one of three modes:

- Interpreter mode - NAMESCTL is loaded from the command line. When loaded, the program displays the following prompt:

  ```
  NAMESCTL>
  ```

- Command line mode - You can also execute most commands from the operating system command line by running the NAMESCTL program with a complete command as a parameter to the program. In this case, NAMESCTL will load and execute the command, then return the operating system prompt. Sample commands are:

  ```
  NAMESCTL START

  NAMESCTL STATUS CHEDDAR.ACME
  ```

- Batch command mode - You can combine commands in a standard text file, then run them as a sequence of commands. To execute in batch mode, use the format:

  ```
  NAMESCTL @file_name
  ```

  You can use either REM or # to identify Comments in the batch script; all other lines are considered commands. Any commands that would typically require confirmation do not require confirmation during batch execution.

A.2.2 NAMESCTL Parameter Options

When loading NAMESCTL, any valid parameter settings can be passed to the program to override the default or configured settings. For example:

```
NAMESCTL NAMESCTL.TRACE_LEVEL=ADMIN
```

would load NAMESCTL and turn on tracing to the ADMIN level, regardless of the currently configured value of NAMESCTL.TRACE_LEVEL.

A.2.3 NAMESCTL SET and SHOW Modifiers

You can use the modifier SET to change some parameter values of the Names Server or the Oracle Names Control Utility environment. For example, the following sequence sets the node to control and changes its trace level.

```
NAMESCTL>  SET SERVER DOLPHIN.WORLD
NAMESCTL>  SET TRACE_LEVEL ADMIN
```

The first modifier sets the node to DOLPHIN.WORLD. Subsequent commands are directed to DOLPHIN.WORLD. The second modifier sets the server DOLPHIN.WORLD's trace level. The server will then begin tracing at the ADMIN level.

A.2.4 NAMESCTL's Distributed Operation

The Oracle Names Control Utility operates on a Names Server on the same machine as any other Names Servers in the network. This is very useful when a single administrator is managing all of the Names Servers in a region, or wants to check the availability of a specific Names Server.

Most commands accept the name of a Names Server as the last argument indicating which Names Server to perform the command against. If omitted, the current SET Names Server is used. For example:

```
SHOW SYSTEM_QUERIES DOLPHIN.ACME
```

will display the system queries on the Names Server DOLPHIN.ACME and when they will next occur. To perform a series of commands against an individual Names Server, type

```
NAMESCTL> SET SERVER server_name
```

then perform the commands.

A.2.5 NAMESCTL Security

You have the option of configuring a Names Server to require a password for any NAMESCTL command that alters how it operates.

The value for PASSWORD is set to the value specified for the NAMESCTL.SERVER_PASSWORD parameter in the SQLNET.ORA file on the node running NAMESCTL. This is the password of the first Names Server listed in the NAMES.PREFERRED_SERVERS list. The current setting for PASSWORD must match the value in the NAMES.PASSWORD parameter in the NAMES.ORA file on the current Names Server.

If you are concerned with the security implications of explicitly putting a Names Server password in the administrator's client SQLNET.ORA file, you can omit the parameter and always use the command:

```
NAMESCTL> SET PASSWORD
```

You will be prompted for the password. When passed over the network, the password is ALWAYS encrypted, regardless of how it is set in NAMESCTL.

A.2.6 Confirmation Mode in NAMESCTL

Some of the NAMESCTL commands require your confirmation before they are executed. When you issue the command, you are prompted:

```
confirm:[yes or no]
```

Type "yes" to execute the command; type "no" to cancel the command.

You can turn confirmation mode off by using by setting the parameter

```
NAMESCTL.NOCONFIRM = TRUE
```

in a profile (SQLNET.ORA). Note that with this parameter set to OFF, all commands execute without asking for confirmation.

A.2.7 NAMESCTL Commands

The following commands are available through the Oracle Names Control Utility (NAMESCTL).

DELEGATE_DOMAIN	
Purpose:	Defines a domain as the start of a subregion of the current region
Prerequisites:	none
Password Required:	No
Syntax:	From the operating system prompt: `NAMESCTL _DELEGATE_DOMAIN` *`domain_name ns_name ns_addr`* From the NAMESCTL program: `_DELEGATE_DOMAIN` *`domain_name ns_name ns_addr`*
Arguments:	*domain_name ns_name ns_addr* are mandatory. *domain_name* and *ns_name* must be legal domain names. *ns_addr* must be a legal TNS address.
Usage Notes:	This command provides a dynamic way to subdivide the namespace. The domain *domain_name* is defined with *ns_name* as the name of the Names Server. *ns_name* is defined with its address set to *ns_addr*. Unless a domain is delegated from a region, the servers in that region will assume authority over all sub-domains. Once a domain is delegated, a new region is created and the servers in the current region are no longer authoritative for the new domain or any sub-domain of the new domain. After delegation, Names Servers in the current domain will forward subsequent operations for objects in the new domain to its Names Servers.
Examples:	`NAMESCTL> _DELEGATE_DOMAIN webwidgets.acme.com ns1.webwidgets.acme.com (address=(protocol=tcp) (host=fred.webwidgets.acme.com) (port=1575))`

DOMAIN_HINT

Purpose:	Provides the Names Servers in the current region with the name and address of a Names Server in another region.
Prerequisites:	None
Password Required:	No
Syntax:	From the operating system: `NAMESCTL _DOMAIN_HINT` *`domain_name ns_name ns_addr`* From the NAMESCTL program: `_DOMAIN_HINT` *`domain_name ns_name ns_addr`*
Arguments:	*domain_name ns_name* and *ns_addr* are mandatory. *domain_name* and *ns_name* must be legal domain names. *ns_addr* must be a legal TNS address.
Usage Notes:	This command provides a dynamic way to define the path to other regions in the namespace. The domain *domain_name* is defined with *ns_name* as the name of its Names Server. *ns_name* is defined with its address set to *ns_addr*. Any region that is not the root region will need at least the root region defined using this command in order to find objects in any other region. You may provide additional hints as optimizations to provide local Names Servers with direct access to certain other regions.
Examples:	`NAMESCTL> _DOMAIN_HINT acme.com ns0.acme.com (address=(protocol=tcp) (host=top.acme.com) (port=1575))`

EXIT	
Purpose:	The EXIT command closes the NAMESCTL program.
Prerequisites:	The NAMESCTL program must be loaded.
Password Required:	No
Syntax:	From the NAMESCTL program: `EXIT`
Arguments:	None
Usage Notes:	EXIT has no effect on any Names Servers. It affects only the NAMESCTL program. The EXIT command is identical to the QUIT command.
Example:	`NAMSCTL> EXIT` `NAMESCTL finished.`

FLUSH

Purpose:	Drops all stored non-authoritative data from the Names Server cache.
Prerequisites:	Only relevant with an environment with multiple regions. (In central administration there is no non-authoritative data.)
Password Required:	Yes

Syntax:	From the operating system prompt:
	`NAMESCTL FLUSH [server] ...`
	From the NAMESCTL program:
	`FLUSH [server] ...`

Arguments:	Zero or more Server names separated by a space. When no arguments are supplied, only the current Names Server's cache is flushed of the foreign names
Usage Notes:	FLUSH erases *all foreign data* that has been cached. Typically, you should flush the foreign data cache when: - A large volume of data changes in the network and the normal TTL aging mechanism will take too long. - When unidentifiable errors in name resolution of cached foreign data are occurring. Flushing all foreign data from the cache forces it to be looked up again when it is requested the next time.
Examples:	`NAMESCTL>FLUSH` `Confirm [yes or no]: yes`

FLUSH_NAME	
Purpose:	Drops one or more specific non-authoritative names from the current Names Server's cache.
Prerequisites:	Only meaningful with an environment with multiple regions. (In central administration, there is no non-authoritative data.)
Password Required:	Yes

Syntax:	From the operating system prompt:
	`NAMESCTL FLUSH_NAME name`
	From the NAMESCTL program:
	`FLUSH_NAME name`

Arguments:	A single name
Usage Notes:	FLUSH_NAME erases only data cached from outside the Names Server's region (that is, non-authoritative data). It is typically flushed when a name is behaving unusually, suggesting the source copy may have changed. FLUSH_NAME removes the name from the current foreign data cache as well as any other Servers between the current region and the authoritative region. Names are flushed from the current Names Server. The current Names Server is either the default preferred Names Server or the one set by using the SET SERVER command.
Examples:	`NAMESCTL>FLUSH_NAME MOUNTAIN.ACME.COM`

HELP

Purpose:	Provides details of the NAMESCTL commands.
Prerequisites:	None.
Password Required:	No

Syntax:	From the operating system prompt:
	`NAMESCTL HELP [command]`
	From the NAMESCTL program:
	`HELP [command]`

Arguments:	commands
Usage Notes:	Help provides brief reminders of the function of each command in NAMESCTL. When no arguments are supplied, help shows the list of valid commands. When you supply an argument, a one line description of that command's function is displayed.

Example:	`NAMESCTL> HELP`
	The following operations are available
	`An asterisk (*) denotes a modifier or extended command:`
	`exit`
	`flush`
	`flush_name`
	`log_stats`
	`ping`
	`query`
	`quitreload`
	`repeat*`
	`reset_stats`
	`restart set*`
	`show`
	`shutdown`
	`start`
	`startup`
	`status`
	`stop version`

LOG_STATS

Purpose:	Logs the current set of Names Server statistics to the configured log file for that Names Server.
Prerequisites:	None
Password Required:	Yes

Syntax:	From the operating system prompt:
	`NAMESCTL LOG_STATS [`*server*`] ...`
	From the NAMESCTL program:
	`LOG_STATS [`*server*`]`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the statistics for the current Names Server are reset.
Usage Notes:	Statistics may be logged if the STATUS command or other behavior indicates some data that you would like to capture in the log. LOG_STATS does not affect the current LOG_STATS_INTERVAL.
Example:	`NAMESCTL>LOG_STATS` `Statistics counters logged.`

PASSWORD

Purpose:	Registers the password for privileged Names Server operations such as RELOAD and STOP.
Prerequisites:	The NAMESCTL program must be loaded.
Password Required:	N/A
Syntax:	From the NAMESCTL program: `PASSWORD [`*`password`*`]`
Arguments:	Text string matching the value encrypted in the current Names Server parameter NAMES.PASSWORD.
Usage Notes:	PASSWORD does not change the Names Server's password. It simply sets a NAMESCTL variable. Then, the value stored is sent from NAMESCTL with any command request to the Names Server, and the value is compared to the value configured on the Names Server. If they match, operations requiring passwords are allowed. Only "privileged" operations are affected, that is, operations that alter the functioning of the Names Server. Operations such as SHOW or STATUS are not considered privileged, and do not require a password. The password can either be passed as an argument of the PASSWORD command, or if no argument is given, it will be prompted for. Note that the input is not displayed on the screen as it is typed. When passed over the network the password is ALWAYS encrypted, regardless of how it is set.

Example:	`NAMESCTL> PASSWORD OPEN_SESAME`
	`NAMESCTL> PASSWORD`
	`Enter name server password:`

PING	
Purpose:	Contacts the current Names Server, or named server(s), and display the request/response time.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system prompt:
	`NAMESCTL PING [server_name] ...`
	From the NAMESCTL program:
	`PING [server_name] ...`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the current Names Server is pinged.
Usage Notes:	Ping ensures that a Names Server is functioning and shows typical response times from the location of the NAMESCTL user to a Names Server.
Example:	`NAMESCTL> ping NSERVER.world` `Round trip time is 0.04 seconds`

QUERY

Purpose:	Tests or retrieves the contents of a network object stored in the Names Server.
Prerequisites:	None
Password Required:	No

Syntax	From the operating system prompt:
	`NAMESCTL QUERY object_name [object_type] [modifiers]`
	From the NAMESCTL program:
	`QUERY name [type] [modifiers]`
	
	VALID TYPES:
	■ `A.SMD`: Network addresses, as with database service definitions.
	■ `CNAME.SMD:` Alias name (sometimes referred to as ”canonical name”).
	■ `DL.RDBMS.OMD:` Database link.
	■ `NS.SMD`: Names Server addresses. System data used to communicate between Names Servers.
	■ `V1ADD.NPO.OMD`: SQL*Net Version 1 connect string
	VALID MODIFIERS:
	■ `AUTHORITY -` Forces the query to be resolved at the source of the data (in the administrative region where the data is considered local) even if the data is in the local cache. This could be used if the administrator suspects that the data has changed at the source.
	■ `NOFORWARD -` Query for the data, but don’t forward the request. When the data is not local, and noforward is specified, the query will not be resolved.
	■ `TRACE -` Allows a trace of the path to the answer. This is useful whenever you want to find out which Names Servers the request went to.

Arguments:	Mandatory network object name and network object type

QUERY

Usage Notes:

QUERY can be used to test that a defined piece of data can be found, and that the contents are correct.

The QUERY command always operates on the current Names Server, either the default, or as specified using the SET SERVER command.

If the QUERY command is used with just a name as a parameter, the Names Server responds with the number of pieces of data with that name, and the time required to complete the operation.

If the QUERY command is used with the name and type supplied as arguments; the specific name is looked up and returned to the user.

The QUERY command can take multiple arguments if appropriate. For example:

```
QUERY NAME.WORLD A.SMD AUTHORITY TRACE
```

Example:

```
NAMESCTL> QUERY BONES.DEM.MEDICINE A.SMD
Total response time:0.04 seconds
Response status:normal, successful completion
Authoritative answer:yes
Number of answers:1
Canonical name:bones.dem.medicine
TTL: 1 day
Alias translations:
    from:bones.dem.medicine
    to: bones.dem.medicine
Answers:
    data type is ”a.smd”
        Syntax is
ADDR:...(DESCRIPTION=(ADDRESS=
(PROTOCOL=TCP)(Host=cowboy)
(Port=1522))(CONNECT_DATA=(SID=rodeo)))
```

QUIT	
Purpose:	Quits the NAMESCTL program.
Prerequisites:	The NAMESCTL program must be loaded.
Password Required:	No
Syntax:	From the NAMESCTL program: `QUIT`
Arguments:	None
Usage Notes:	QUIT has no effect on any Names Servers. It affects only the NAMESCTL program. The QUIT command is functionally equivalent to the EXIT command.
Example:	`NAMESCTL> EXIT` `NL-00851: NAMESCTL Finished`

REGISTER	
Purpose:	Registers a network object to a Names Server
Prerequisites:	None
Syntax:	From the NAMESCTL program: register *object_name* [-t *type_of_service*] [-d *address_data*] [-h *hostname*] [-l *listener_name*]
Arguments:	Mandatory object name. The service, address data, and host are not necessary to make the registration process appear to work. However, they are necessary to make the registration useful. In other words, an object name registered without an address cannot be used.
Usage Notes:	Provides a manual mechanism for registering a service, its type, its hostname, and its address. Both the type of service and the data may be any valid string, but the typical registration has either ”database” or ”listener” as type of service, and the TNS address as the data. The object registration is propagated to all other well known Names Servers in the region.
Example:	NAMESCTL> register parts -t oracle_database -d (DESCRIPTION= (ADDRESS= (PROTOCOL=TCP)(HOST=nineva)(PORT=1387)) (CONNECT_DATA=(SID=db3)))

RELOAD

Purpose:	Forces the server to check immediately for data changes in its administrative region, and if there are any, reloads all database service names, global database links, and aliases.
Prerequisites:	None
Password Required:	Yes

Syntax:	From the operating system prompt:
	`NAMESCTL RELOAD [`*servers*`] ...`
	From the NAMESCTL program:
	`RELOAD [`*servers*`] ...`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the current Names Server is reloaded.
Usage Notes:	All Names Servers load their data directly from the database specified by the names.admin_region configuration parameter in the Names Server configuration file. In an environment with multiple regions, RELOAD affects only the data for the current administrative region. All foreign data in the cache is unchanged.
Example:	`NAMESCTL>RELOAD` `Server reloaded.`

REORDER_NS

Purpose:	Creates the file which lists local Names Servers and their addresses.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system prompt:
	`NAMESCTL REORDER_NS [NS_ADDRESS]`
	
	From the NAMESCTL program:
	`REORDER_NS [NS_ADDRESS]`

Arguments:	An optional Names Server address will be used as the initial server to contact.
Usage Notes:	This command generates the file which defines Names Server names and addresses to enable clients to contact Names Servers for name lookup. The `REORDER_NS` command does the following; **1.** Finds a Names Server. It looks in the profile (`SQLNET.ORA`) for a Preferred Names Server parameter, or tries calling a well-known servers. It uses the address argument if it is present. **2.** Sends a query for all the Oracle Names Servers in the local region. **3.** Sends a 'ping' to each of these servers. **4.** Sorts the list of servers by increasing order of response time. **5.** Writes a Names Server List with the sorted list of names and addresses.
Example:	`NAMESCTL> REORDER_NS (address= (protocol=tcp)(host=nineva)(port=1383))`

REPEAT

Purpose:	Used to perform QUERY, REGISTER, TIMED_QUERY, or UNREGISTER multiple times to compute average return rates.
Prerequisites:	None
Password Required:	No
Syntax:	From the operating system prompt: `NAMESCTL REPEAT` *`number`* `QUERY` *`type`* From the NAMESCTL program: `REPEAT` *`number`* `QUERY` *`type`* where *number* is an integer and *type* is as shown in the QUERY command.
Arguments:	None
Usage Notes:	Repeat is useful for understanding the average response time over a number of requests. Do not specify too large a number here; while the number of iterations are occurring, the NAMESCTL program cannot do anything else.

Example:	`NAMESCTL> repeat 10 query manatee a.smd`
	`Number of requests: 10`
	`Average response time: 0.01 seconds`
	`Minimum response time: 0.01 seconds`
	`Maximum response time:0.04 seconds`
	`Total response time:0.14 seconds`
	`Response status:normal, successful completion`
	`Authoritative answer:yes`
	`Number of answers: 1`
	`TTL: 1 day`
	`Answers:`
	`data type is "a.smd"`
	`Syntax is ADDR:(DESCRIPTION=(ADDRESS=`
	`(PROTOCOL=TCP)(Host=salmon)`
	`(Port=1522))(CONNECT_DATA=(SID=otter)))`

RESET_STATS

Purpose:	Resets the Names Server statistics to the original values of the Names Server at startup.
Prerequisites:	None
Password Required:	Yes

Syntax:	From the operating system prompt:
	`NAMESCTL RESET_STATS [`*server*`]`
	From the NAMESCTL program:
	`RESET_STATS [`*server*`]`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the current Names Server's statistics are reset.
Usage Notes:	RESET_STATS has the same effect as waiting for the RESET_STATS_INTERVAL to conclude, except that it happens immediately.

Example:	`NAMESCTL> RESET_STATS`
	`Confirm [yes or no]: yes`
	`Server statistics reset.`

RESTART

Purpose:	Initiates a reset of a Names Server to its original state at startup.
Prerequisites:	None
Password Required:	Yes

Syntax:	From the operating system prompt:
	`NAMESCTL RESTART [`*server*`]`
	From the NAMESCTL program:
	`RESTART [`*server*`]`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the current Names Server is restarted.
Usage Notes:	RESTART is the same as STARTUP except that the Names Server is already running. Data is reloaded, statistics are reset, and all foreign data is flushed. Valid foreign cache data (that is, data with a TTL greater than 0) is retrieved from the checkpoint files. (The TTL value must be set to more than 0.)

Example:	`NAMESCTL> RESTART`
	`Confirm [yes or no]: yes`
	`Server restarted.`

SET CACHE_CHECKPOINT_INTERVAL

Purpose:	Sets how often all collected information about foreign regions is saved in the Names Server cache file.
Syntax:	From the operating system: `NAMESCTL SET CACHE_CHECKPOINT_INTERVAL` *`time`* From NAMESCTL program: `SET CACHE_CHECKPOINT_INTERVAL` *`time`*
Arguments:	Time is in *seconds* For example, to increase the CACHE_CHECKPOINT_INTERVAL to 36 hours, the following will work: `SET CACHE_CHECKPOINT_INTERVAL 129600`
Usage Notes	Minimum: 10 seconds Maximum: 259200 (3 days) Default: 0 (disabled)
Example:	`NAMESCTL> SET CACHE_CHECKPOINT_INTERVAL 10`

SET DEFAULT_DOMAIN

Purpose:	Sets or changes the default domain for the NAMESCTL client.
Prerequisites:	The NAMESCTL program must be loaded.
Password Required:	No
Syntax:	From the NAMESCTL program: `SET DEFAULT_DOMAIN` *domain_name*
Arguments:	one domain name
Usage Notes:	The existence of the `DEFAULT_DOMAIN` parameter allows names to be unqualified for names in that domain. For example, with `DEFAULT_DOMAIN` set to US.ACME the global name WIDE.US.ACME could be queried using: `NAMESCTL> QUERY WIDE` The initial value of DEFAULT_DOMAIN is set when the NAMESCTL program is started from the `NAMES.DEFAULT_DOMAIN` parameter in the client profile (`SQLNET.ORA`). When no arguments are specified, the default is read and assigned from the client profile (`SQLNET.ORA`). `SET DEFAULT_DOMAIN` could be used to simplify working on a set of names within a single domain, then a set in another.
Example:	`NAMESCTL> SET DEFAULT_DOMAIN US.ACME` `Default domain is now "US.ACME"`

SET FORWARDING_AVAILABLE

Purpose:	Turns on or off name request forwarding for a Names Server
Prerequisites:	Names Server must be running
PasswordRequired:	Yes

Syntax:	From the operating system:
	`NAMESCTL SET FORWARDING_AVAILABLE OFF`
	From the NAMESCTL program:
	`SET FORWARDING_AVAILABLE OFF`

Arguments:	Time arguments are: `[[ON|OFF] [YES|NO]]`
Restrictions:	Default Value: 0 (ON)
Usage Notes:	This setting is intended only for Names Servers that have no local clients and are exclusively handling requests from foreign Names Servers. This usually would only apply to Names Servers in the root region when the root is configured without clients or services. If such a server is a performance bottleneck in cross-region request processing then disabling forwarding in that Names Server will cut its workload in half. Rather than forward the request and return the answer the Names Server simply tells the requestor the address of the Names Server that can answer the request. Note that there is no overall reduction in work; the work is simply displaced from the non-forwarding Names Server to the requesting Names Server. **WARNING**: If SET FORWARDING_AVAILABLE is set to off, any clients who rely directly on that Names Server will be unable to resolve foreign names. Clients are not capable of redirecting their requests as Names Servers would. Their requests will fail at that point, even if other Names Servers are listed in the NAMES.PREFERRED_SERVERS configuration parameter.
Example:	`NAMESCTL> SET FORWARDING_AVAILABLE OFF` `Request processing is now disabled.`

SET LOG_FILE_NAME

Purpose:	Changes the log filename.
Prerequisites:	None
PasswordRequired:	Yes
Syntax:	From the operating system: `NAMESCTL SET LOG_FILE_NAME` *`filename`* From the NAMESCTL program: `SET LOG_FILE_NAME` *`filename`*
Arguments:	filename of the log file.
Usage Notes:	The LOG_FILE_NAME changes the destination of all logging messages.
Example:	`NAMESCTL> SET LOG_FILE_NAME namesvr1`

SET LOG_STATS_INTERVAL

Purpose:	Changes the frequency with which the statistics are logged to the log file.
Prerequisites:	None
PasswordRequired:	Yes
Syntax:	From the operating system: `NAMESCTL SET LOG_STATS_INTERVAL` *time* From the NAMESCTL program: `NAMESCTL>SET LOG_STATS_INTERVAL` *time*

Arguments:	Time is in seconds or[<*n*> DAY[S]] [<*hh*>:<*mi*>:<*ss*>]
	For example, to increase the LOG_STATS_INTERVAL to 36 hours, either of the following will work:
	`SET LOG_STATS_INTERVAL 129600`
	`SET LOG_STATS_INTERVAL 1 DAY 12:00:00`
	You can specify any valid combination, such as the number of days combined with number of hours, minutes, and seconds; or just the number in hours.

Restrictions:	Minimum Value: 10 seconds Maximum Value: no maximum Special Value: 0 (which means never reset) Default value: 0 (no logging)
Usage Notes:	The LOG_STATS_INTERVAL value is initially set based on the value configured in the Oracle Network Manager, or the value in NAMES.LOG_STATS_INTERVAL in the SQLNET.ORA file when the Names Server is loaded. By default, the value is 0 (no logging). This command is intended to override that value during server operation.
Example:	`NAMESCTL> SET LOG_STATS_INTERVAL 7200` `Statistic counter logging interval is now 2 hours`

SET NAMESCTL_TRACE_LEVEL

Purpose:	Sets the level at which the NAMESCTL program can be traced.
Prerequisites:	None
PasswordRequired:	Yes

Syntax:	From the operating system:
	`NAMESCTL SET NAMESCTL_TRACE_LEVEL [OFF|USER|ADMIN|SUPPORT]`
	From the NAMESCTL program:
	`NAMESCTL>SET NAMESCTL_TRACE_LEVEL [OFF|USER|ADMIN|SUPPORT]`

Arguments:	OFF, USER, ADMIN, SUPPORT
Usage Notes:	Tracing assists in diagnosing unexpected or unidentifiable failures in processing the NAMESCTL program. Tracing writes a series of events from normal NAMESCTL processing to an operating system file for review by the administrator. Tracing output is at three levels OFF (none), USER (basic information), or ADMIN. When no arguments are supplied, the setting is reset to the value in the client's SQLNET.ORA file. The default setting is OFF.
Example:	`NAMESCTL> SET NAMESCTL_TRACE_LEVEL ADMIN` `Controller's local trace level changed from 0 to 4`

SET PASSWORD	
Purpose:	Register the password for privileged Names Server operations such as RELOAD and STOP.
Prerequisites:	The NAMESCTL program must be loaded.
Password Required:	N / A
Syntax:	From the NAMESCTL program: `NAMESCTL>SET PASSWORD [`*`password`*`]`
Arguments:	Text string matching the value stored in the current Names Server parameter NAMES.PASSWORD.
Usage Notes:	SET PASSWORD does not change the Names Server's password. It simply sets a NAMESCTL variable that is sent over to the Names Server with any NAMESCTL command and is compared to the value configured on the Names Server. If they match, operations requiring passwords are allowed. Only "privileged" operations are affected, that is, operations that alter the functioning of the Names Server. Operations such as SHOW or STATUS are not considered privileged, and do not require a password. The password can either be passed as an argument of the SET PASSWORD command, or if no argument is given, it will be prompted for. Note that the input is not displayed on the screen as it is typed. When passed over the network the password is ALWAYS encrypted, regardless of how it is set.

Example:	`NAMESCTL> SET PASSWORD OPEN_SESAME`
	`NAMESCTL> SET PASSWORD`
	`Enter name server password:`

SET REQUESTS_ENABLED

Purpose:	Determine whether the current Names Server will respond to requests.
Prerequisites:	None
Password Required:	Yes

Syntax:	From the operating system:
	`NAMESCTL REQUESTS_ENABLED [ON|OFF]`
	From the NAMESCTL program:
	`NAMESCTL>SET REQUESTS_ENABLED [ON|OFF]`

Arguments:	ON or OFF
Usage Notes:	Setting this property to OFF will send refusals to all clients that approach with Names requests. This is primarily useful for diagnostics when a Names Server is functioning unexpectedly.

Example:	`NAMESCTL> SET REQUESTS_ENABLED OFF`
	`Confirm [yes or no]: yes`
	`General request processing is now disabled`

SET RESET_STATS_INTERVAL

Purpose:	Changes the time between the statistics being reset to zero or initial values in the current server.
Prerequisites:	None
Password Required:	Yes
Syntax:	From the operating system: `NAMESCTL SET RESET_STATS_INTERVAL` *`time`* From the NAMESCTL program: `NAMESCTL>SET RESET_STATS_INTERVAL` *`time`*

Arguments:	Time is in one of the following formats:
	`seconds`
	[*n DAY*[S]] [*hh:mi:ss*]
	For example, to increase the RESET_STATS_INTERVAL to 72 hours, either of the following will work:
	`SET RESET_STATS_INTERVAL 259200`
	`SET RESET_STATS_INTERVAL 3 DAYS`

Restrictions:	Minimum Value: 10 seconds Maximum Value: no maximum Default value: 0 (never reset)
Usage Notes:	The RESET_STATS_INTERVAL value is initially set based on the NAMES.RESET_STATS_INTERVAL parameter when the Names Server is loaded. This command is intended to override that value during Names Server operation.
Example:	`NAMESCTL> SET RESET_STATS_INTERVAL 1 DAY` `Statistic counter reset interval is now 24 hours`

SET SERVER

Purpose:	Change the current Names Server.
Prerequisites:	The NAMESCTL program must be loaded
Password Required:	No
Syntax:	From the NAMESCTL program: `NAMESCTL>SET SERVER [`*`server_name or server_address`*`]`
Arguments:	valid server name or valid server address
Usage Notes:	SET SERVER allows switching between multiple Names Servers while running the NAMESCTL utility. The qualifier can be a name where the name is defined in the memory of the current Names Server, or it can be the TNS address of any Names Server. The Names Server name specified is resolved through normal name lookup. Another Names Server can only be set if the current Names Server knows or can retrieve its address. If no current Names Server is set, you must type a TNS address to complete this command. IF there are no arguments, use NAMES.PREFERRED_SERVERS.
Example:	`NAMESCTL> SET SERVER SERVER1.US.ACME`

SET TRACE_FILE_NAME

Purpose:	Changes the trace destination filename
Prerequisites:	None
Password Required:	No
Syntax:	From the operating system: `NAMESCTL SET TRACE_FILE_NAME` *`filename`* From NAMESCTL program: `NAMESCTL>SET TRACE_FILE_NAME` *`filename`*
Arguments:	filename
Example:	`NAMESCTL> SET TRACE_FILE_NAME namesvr1`

SET TRACE_LEVEL

Purpose:	Changes the TRACE_LEVEL for tracing the current Names Server.
Prerequisites:	None
Password Required:	Yes

Syntax:	From the operating system:
	`NAMESCTL SET TRACE_LEVEL [OFF|USER|ADMIN|SUPPORT]`
	From the NAMESCTL program:
	`NAMESCTL>SET TRACE_LEVEL [OFF|USER|ADMIN|SUPPORT]`

Arguments:	OFF, USER, ADMIN, SUPPORT
Usage Notes:	Tracing assists in diagnosing unexpected or unidentifiable failures in processing the current Names Server. Tracing writes a series of events from normal Names Server processing to an operating system file for review by the administrator. Tracing output is at three levels OFF (none), USER (basic information), ADMIN (information required for administrator), or SUPPORT (information required for customer support). After the TRACE_LEVEL is set, tracing begins immediately. All operations are traced until it is reset to trace level OFF. Trace files can grow very large. Remember to turn trace level off after diagnosing the problem.
Example:	`NAMESCTL> SET TRACE_LEVEL ADMIN` `Trace level is now 6.`

SHOW CACHE_CHECKPOINT INTERVAL

Purpose:	Shows the frequency with which the Names Server's cache is written to the checkpoint file.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW CACHE_CHECKPOINT_INTERVAL`
	From the NAMESCTL program:
	`NAMESCTL>SHOW CACHE_CHECKPOINT_INTERVAL`

Arguments:	None
Usage Notes:	The CACHE_CHECKPOINT_INTERVAL is initially set with the value in `NAMES.CACHE_CHECKPOINT_INTERVAL` in the NAMES.ORA file. By default, the value is 0, which disables cache_checkpoint. Data written to the cache checkpoint file includes service names and addresses, and Names Server addresses which were learned by the Names Server as a result of forwarding a query to a foreign region on behalf of the client.
Example:	`NAMESCTL> SHOW CACHE_CHECKPOINT_INTERVAL` `Cache checkpoint interval is currently 8 minutes 20 seconds`

SHOW FORWARDING_AVAILABLE

Purpose:	Shows whether the Names Server is forwarding requests for foreign names or redirecting them.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW FORWARDING_AVAILABLE`
	From the NAMESCTL program:
	`NAMESCTL>SHOW FORWARDING_AVAILABLE`

Arguments:	Zero or more Names Servers separated by a space. If no names are given, then the setting is displayed for the current server.
Usage Notes:	By default, all Names Servers forward requests for foreign names. If forwarding is disabled, then requests for foreign names will be redirected to a Names Server in the region which is authoritative to the requested name. Disabling forwarding can reduce the load on a particular server, and will also make it impossible for direct clients of that server to resolve foreign names. Clients cannot be redirected, only other Names Servers. See also `SET FORWARDING_AVAILABLE`.
Example:	`NAMESCTL> SHOW FORWARDING_AVAILABLE` `Request forwarding is currently enabled`

SHOW DEFAULT_DOMAIN

Purpose:	Display the default domain for the NAMESCTL client.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW DEFAULT_DOMAIN`
	From the NAMESCTL program:
	`NAMESCTL> SHOW DEFAULT_DOMAIN`

Arguments:	None

Usage Notes:	The existence of the DEFAULT_DOMAIN parameter allows names to be unqualified for names in that domain. For example, with DEFAULT_DOMAIN set to ACME.WORLD, the global name WIDE.ACME.WORLD could be queried using:
	`NAMESCTL> QUERY WIDE`
	`Total response time:  0.20 seconds`
	`Response status: normal, successful completion`
	`Authoritative answer:  yes`
	`Number of answers:  0`
	`TTL:  1 day`
	The initial value of DEFAULT_DOMAIN is set when the NAMESCTL program is started from the NAMES.DEFAULT_DOMAIN parameter in the SQLNET.ORA file.
	SHOW DEFAULT_DOMAIN is used when the user is unsure of the current default domain, or wants to know the default for the current configuration.

Example:	`NAMESCTL> SHOW DEFAULT_DOMAIN` `Current default domain is "world"`

SHOW LOG_FILE_NAME

Purpose:	Shows the name of the file where the Names Server writes the logging information.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW LOG_FILE_NAME`
	From the NAMESCTL program:
	`SHOW LOG_FILE_NAME`

Arguments:	None
Usage Notes:	The LOG_FILE_NAME is initially set with the value in NAMES.LOG_FILE_NAME in the NAMES.ORA file.The default value is platform-specific, but is typically NAMES.LOG and is located in the network/log subdirectory during Oracle installation. This file must be writable to the Names Server.

Example:	`NAMESCTL> SHOW LOG_FILE_NAME`
	`Log file name is currently`
	`/private/ora23/network/names.log`

SHOW LOG_STATS_INTERVAL

Purpose:	Displays the frequency with which the statistics are logged to the log_file.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW LOG_STATS_INTERVAL`
	From NAMESCTL program:
	`SHOW LOG_STATS_INTERVAL`

Arguments:	Zero or more Names Servers separated by a space. If no names are given, then the setting is displayed for the current server.
Usage Notes:	The LOG_STATS_INTERVAL is initially set with the value in NAMES.LOG_STATS_INTERVAL in the NAMES.ORA file. By default, the value is 0, or no logging.
Example:	`NAMESCTL> SHOW LOG_STATS_INTERVAL` `Statistic counter logging is currently disabled`

SHOW NAMESCTL_TRACE_LEVEL

Purpose:	Displays control of the level at which the NAMESCTL program is being traced.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW NAMESCTL_TRACE_LEVEL`
	From the NAMESCTL program:
	`SHOW NAMESCTL_TRACE_LEVEL`

Arguments:	None
Usage Notes:	Tracing assists in diagnosing unexpected or unidentifiable failures in processing the NAMESCTL program. Tracing writes a series of events from normal NAMESCTL processing to an operating system file for review by the administrator. Tracing output is at three levels OFF (none), USER (basic information), or ADMIN (maximum amount of information). `SHOW NAMESCTL_TRACE_LEVEL` is the only guaranteed source of what the current tracing level is.
Example:	`NAMESCTL> SHOW NAMESCTL_TRACE_LEVEL` `Controller's trace level is currently 0`

SHOW REQUESTS_ENABLED

Purpose:	Shows whether or not the Names Server is responding to requests.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW REQUESTS_ENABLED`
	From the NAMESCTL program:
	`SHOW REQUESTS_ENABLED`

Arguments:	Zero or more Names Servers separated by a space. If no names are given, then the setting is displayed for the current server.
Usage Notes:	If REQUESTS_ENABLED is off, all requests to the Names Server will be refused. This parameter is intended for diagnostic purposes only.
Example:	`NAMESCTL> SHOW REQUESTS_ENABLED` `General request processing is currently enabled`

SHOW RESETS_STATS_INTERVAL

Purpose:	Shows the interval set on how often the statistics are dumped to the log file.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW RESET_STATS_INTERVAL`
	From the NAMESCTL program:
	`SHOW RESET_STATS_INTERVAL`

Usage Notes:	If RESET_STATS_INTERVAL is initially set with the value in NAMES.RESET_STATS_INTERVAL. By default the value is set to 0, or no reset. This results in the Names Server accumulating statistics the entire time it runs. For example, if statistics are reset every day, then the statistics will represent totals for the day rather than the entire time the server has been running.
Example:	`NAMESCTL> SHOW RESET_STATS_INTERVAL` `Statistic counter reset interval is currently 5 minutes`

SHOW SERVER

Purpose:	Displays the current Names Server
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW SERVER`
	From the NAMESCTL program:
	`SHOW SERVER`

Arguments:	None
Usage Notes:	SHOW SERVER displays the current Names Server that commands will operate on.

Example:	`NAMESCTL> SHOW SERVER`
	`currently managing name server`
	`"NameServer.us.oracle.com`
	`Version Banner is "Oracle Names for SunOS:`
	`Version 2.0.0.0"`

SHOW STATUS	
Purpose:	Display the general status information about the Names Server.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW STATUS`
	From the NAMESCTL program:
	`SHOW STATUS`

Arguments:	Zero or more Names Servers separated by a space. If no names are given, then the setting is displayed for the current server.
Usage Notes:	Shows the current state of a Names Server. Identical to the command STATUS.
Example:	`NAMESCTL> SHOW STATUS` `Version Banner is "Oracle Names for SunOS: Version 2.0.0.0" Server has been running for:1 day 2 hours 3 minutes 35.16 seconds....`

SHOW SYSTEM_QUERIES

Purpose:	Display the next occurrence of all system queries.
Prerequisites:	This is only relevant for distributed configurations. There are no system queries with only one administrative region.
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW SYSTEM_QUERIES`
	From the NAMESCTL program:
	`SHOW SYSTEM_QUERIES`

Arguments:	None
Usage Notes:	System queries are performed at intervals to keep information among Names Servers current. There is no specific action that can change the activities listed as system queries. Being able to show them gives the administrator an understanding of when a system change will occur, and may assist in a decision to RESTART, thus forcing system data to be reloaded sooner.

Example:	`NAMESCTL> SHOW SYSTEM_QUERIES`
	`System query index number:1`
	`Query ID:49824`
	`Query next issued in:2 hours 55 min 3.84 seconds`
	`Query state:2`
	`Name:""`
	`Desired data type:ns.smd`

SHOW TRACE_FILE_NAME

Purpose:	Displays the TRACE_FILE_NAME for the current Names Server.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW TRACE_FILE_NAME`
	From the NAMESCTL program:
	`SHOW TRACE_FILE_NAME`

Arguments:	None
Usage Notes:	The TRACE_FILE_NAME is initially set with the value in the NAMES.TRACE_FILE_NAME in the NAMES.ORA file.The default value is platform-specific, but is typically NAMES.TRC and is located in the network/trace subdirectory during the Oracle installation. This file must be a valid filename, and the file must be writable to the Names Server. This file is only used if tracing is enabled using the NAMES.TRACE_LEVEL.

Example:	`NAMESCTL> SHOW TRACE_FILE_NAME`
	`Trace file name is currently`
	`/private/ora23/network/names.trc`

SHOW TRACE_LEVEL

Purpose:	Displays the TRACE_LEVEL for tracing the current Names Server.
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW TRACE_LEVEL`
	From the NAMESCTL program:
	`SHOW TRACE_LEVEL`

Arguments:	None
Usage Notes:	Tracing assists in diagnosing unexpected or unidentifiable failures in processing the current Names Server. Tracing writes a series of events from normal Names Server processing to an operating system file for review by the administrator. Tracing output is at three levels OFF (none), USER (basic information), or ADMIN (maximum amount of information). SHOW TRACE_LEVEL is the only guaranteed source of what the current tracing level is. Even if the TRACE_LEVEL is configured in the Names Server configuration file, a previous call from the NAMESCTL program may have overridden it.
Example:	`NAMESCTL> SHOW TRACE_LEVEL` `Trace level is currently 0`

SHOW VERSION	
Purpose:	Displays the current version and name of the Names Server
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL SHOW VERSION`
	From the NAMESCTL program:
	`SHOW VERSION`

Arguments:	Zero or more Names Servers separated by a space. If no names are given, then the setting is displayed for the current server.
Usage Notes:	This banner identifies the server by name and version. This can be useful when clearing up minor difficulties. This command is enabled every time you connect NAMESCTL to a server.

Example:	`NAMESCTL> SHOW VERSION`
	`Currently managing Names Server`
	`"NameServer.world"`
	`Version banner is "Oracle Names for SunOS:`
	`Version 2.0.0.0.0"`

SHUTDOWN

Purpose:	Stops one or more Names Servers.
Prerequisites:	Names Server must be started.
Password Required:	Yes

Syntax:	From the operating system:
	`NAMESCTL SHUTDOWN [`*server*`]`
	From the NAMESCTL program:
	`SHUTDOWN [`*server*`]`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the current Names Server is shut down.
Usage Notes:	SHUTDOWN stops the current Names Server and unloads the program from memory. A Names Server should only be shut down for operational reasons like upgrades or machine maintenance. The preferred way to stop and start a Names Server is using the RESTART command because you can perform it from anywhere in the network. If SHUTDOWN and START are processed individually, they must occur on the Names Server machine. SHUTDOWN is identical to STOP.

Example:	`NAMESCTL> SHUTDOWN`
	`Confirm [yes or no] yes`
	`Server shut down.`

START

Purpose:	Loads the Names service program and starts loading system and local administrative region data.
Prerequisites:	Names Server must be stopped.
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL START`
	From the NAMESCTL program:
	`NAMESCTL> START names.parameter = value`

Arguments:	None
Usage Notes:	START is the command used to initially load a Names Server into memory. At startup, the Names Server reads its configuration files to set up its operating parameters, then loads all data for the administrative region. Security on Names Server startup is supplied through the operating system Names is installed on. Because Names must be started from a local session, network security is not an issue. See the Oracle installation manual for your platform for more details. START is identical to STARTUP.

Example:	`Starting "/oracle/bin/names"...server successfully started`
	`Currently managing name server "NSERVER.world" ...`
	`Server name: NSERVER.world`
	`Server has been running for: 0.86 seconds`
	`Request processing enabled: yes`
	`Request forwarding enabled: no`
	`Requests received: 0`
	`Requests forwarded: 0`
	`Data items cached: 0...`

START_CLIENT_CACHE

Purpose:	Starts the client cache daemon process.
Prerequisites:	The client cache daemon process must be stopped. A Names Server List must exist before you run the client cache daemon process.
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL START_CLIENT_CACHE`
	From the NAMESCTL program:
	`NAMESCTL> START_CLIENT_CACHE`

Arguments:	None
Usage Notes:	Once started, the client cache daemon process will store all information received from a Names Server.

STARTUP	
Purpose:	This command is identical to START
Prerequisites:	Names Server must be stopped.
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL STARTUP names.parameter = value`
	From the NAMESCTL program:
	`NAMESCTL> START names.parameter = value`

Arguments:	None
Usage Notes:	STARTUP is identical to START.
Example:	See example for START.

STATUS

Purpose:	Displays statistics for one or more Names Servers as well as many of its internal settings.
Prerequisites:	Names Server must be started.
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL STATUS [`*server*`]`
	From NAMESCTL program:
	`STATUS [`*server*`]`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, status is given only for the current Names Server.
Usage Notes:	STATUS shows the activity of the Names Server over time and its state at a point in time.

Example:	`NAMESCTL> STATUS`
	`Version banner is "Oracle Names for SunOS: 2.0.0.0.0"`
	`Server name:NSERVER.world`
	`Server has been running for:1 day 20 hours`
	`.....`

STOP	
Purpose:	Stops one or more Names Servers
Prerequisites:	Names Server must be started.
Password Required:	Yes

Syntax:	From the operating system:
	`NAMESCTL STOP [`*server*`]`
	From the NAMESCTL program:
	`STOP [`*server*`]`

Arguments:	Zero or more Names Server names separated by a space. When no arguments are supplied, only the current Names Server is stopped.
Usage Notes:	STOP stops the current Names Server and unloads the program from memory. A Names Server should only be shut down for operational reasons like upgrades or machine maintenance. The preferred way to stop and start a Names Server is using the RESTART command because you can issue it from anywhere in the network. If STOP and START are processed individually, they must occur on the Names Server machine. STOP is identical to SHUTDOWN.

Example:	`NAMESCTL> STOP`
	`Confirm [yes or no]: yes`
	`Server shut down`

TIMED_QUERY

Purpose:	Show all registered data in the Names Server cache.

Syntax:	From the operating system:
	`NAMESCTL TIMED_QUERY`
	From the NAMESCTL program:
	`TIMED_QUERY [`*`time`*`]`

Usage Notes:	TIMED_QUERY returns all objects that have been registered automatically by the listener or manually through NAMESCTL. The *time* argument returns all objects registered after a given time. To use the *time* argument, the first TIMED_QUERY dumps out all information available since start-up. At the end of the first dump is a "last timestamp" number which gives a bookmark as to where the last dump of information ended. To see all logged data since that point, provide the "last timestamp" number in the *time* argument.

UNREGISTER

Purpose:	To remove a network object from a Names Server
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system
	`NAMESCTL UNREGISTER` *`OBJECT_NAME`*
	`(-d` *`address_data`*`) [-h` *`hostname`*`]`
	`[-l` *`listener_name`*`]`
	`From the NAMESCTL program:`
	`UNREGISTER` *`OBJECT_NAME`* `(-d` *`address_data`*`) [-h` *`hostname`*`] [-l` *`listener_name`*`]`

Arguments:	Mandatory object name and the address, listener, or hostname that it was registered with.
Usage Notes:	Provides a manual mechanism for unregistering a service. The definition for that object is removed from the Names Servers in the region. If the object was registered with an address, listener name, or a hostname, the address, listener name, or hostname must be provided on the command line in order to unregister the object.

Example:	`NAMESCTL> unregister parts -t oracle_database -d`
	`(DESCRIPTION= (ADDRESS=`
	`(PROTOCOL=TCP)(HOST=nineva)(PORT=1387))`
	`(CONNECT_DATA=(SID=db3)))`

VERSION

Purpose:	Displays the current version and name of the Names Server
Prerequisites:	None
Password Required:	No

Syntax:	From the operating system:
	`NAMESCTL VERSION`
	From the NAMESCTL program:
	`VERSION`

Arguments:	Zero or more Names Servers separated by a space. If no names are given, then the setting is displayed for the current server
Usage Notes:	This banner identifies the server by name and version. This can be useful when clearing up minor difficulties. This command is enabled every time you connect NAMESCTL to a server.

Example:	`NAMESCTL> VERSION`
	`Currently managing Names Server`
	`"NameServer.world"`
	`Version banner is "Oracle Names for SunOS:`
	`Version 2.0.0.0.0"`

A.3 Connection Manager Control Utility (CMCTL)

The Connection Manager Control Utility (CMCTL) is a tool that you run from the operating system prompt to start and control Oracle Connection Manager. The general form of the Connection Manager Control Utility is:

```
CMCTL command [process_type]
```

where the *process_type* is the type of process that the command is being executed on. The choices are:

- cman (both main and administration processes)
- adm (only the administration process)
- cm (main process only).

For example, to start both the administration and main processes, you would execute the following:

```
CMCTL start cman
```

In this syntax: `CMCTL` specifies the name of the tool that controls the Oracle Connection Manager. In some operating systems, this fixed parameter is case sensitive. If the operating system is case sensitive, enter CMCTL in lowercase.

You can also issue Oracle Connection Manager Control commands at the program prompt. When you enter CMCTL on the command line, the program is opened. You can then enter the desired commands from the program prompt.

A.3.1 CMCTL Commands

The following commands are available through the Oracle Connection Manager Control Utility (CMCTL).

EXIT	
Purpose:	To exit out of the CMCTL utility program.
Prerequisites:	None
Example:	`CMCTL> exit`

START	
Purpose:	To start Oracle Connection Manager.
Prerequisites:	Oracle Connection Manager must not be running.
Arguments:	cman - start both the administration and main processes
	cm - start only the main process
	adm - start only the administration process
Example:	`CMCTL> start` *process_type*

STATS	
Purpose:	To display basic statistical information including total relays, active relays, most relays out of total, total refused.
Prerequisites:	None
Arguments:	cman - display main process statistics
	cm - display main process statistics
Example:	`CMCTL> stats` *process_type*

STATUS	
Purpose:	To display basic information: version, start time, uptime.
Prerequisites:	None
Arguments:	cman - display both administration and main process information cm - display only main process information adm - display only administration process information
Example:	`CMCTL> status` *process_type*

STOP	
Purpose:	To stop Oracle Connection Manager.
Prerequisites:	Oracle Connection Manager must be running.
Arguments:	cman - stop both the administration and main processes cm - stop only the main process adm - stop only the administration process
Usage Notes:	Verify that all current connections have closed before stopping Oracle Connection Manager. If you issue a stop command while connections remain active, Oracle Connection Manager will remain open. Oracle Connection Manager will not stop until all active connections close. During this time, new users will not be able to connect through Oracle Connection Manager.
Example:	`CMCTL> stop` *process_type*

B

Configuration Parameters

A complete listing of all Net8 configuration parameters is provided for your reference. The following sections appear in this Appendix:

- Section B.1, “Syntax Rules for Configuration Files”
- Section B.2, “Profile Parameters (SQLNET.ORA)”
- Section B.3, “Local Naming Parameters (TNSNAMES.ORA)”
- Section B.4, “Listener Parameters (LISTENER.ORA)”
- Section B.5, “Oracle Names Parameters (NAMES.ORA)”
- Section B.6, “Oracle Connection Manager Parameters (CMAN.ORA)”
- Section B.7, “Protocol-specific Parameters (PROTOCOL.ORA)”

B.1 Syntax Rules for Configuration Files

The configuration files in a Net8 network consist of parameters which include keyword-value pairs. Keyword-value pairs are surrounded by parentheses:

```
parameter = (keyword=value)
```

Some keywords have other keyword-value pairs as their values:

```
(keyword=
    (keyword=value)
    (keyword=value)
)
```

For example, the address portion of a local naming configuration file (TNSNAMES.ORA) might include the following lines:

```
(ADDRESS=
    (PROTOCOL=tcp)
    (HOST=max)
    (PORT=1521)
)
```

Set up configuration files so that indentation reflects what keyword is the "parent" or "owner" of other keyword-value pairs. This format is not required, but it does make the files much easier to read and understand.

Even if you do not choose to indent your files in this way, you must indent a wrapped line by at least one space, or it will be misread as a new parameter. The following layout is acceptable:

```
(ADDRESS=(PROTOCOL=tcp)
  (HOST=max.world)(PORT=1521))
```

The following layout is *not* acceptable:

```
(ADDRESS=(PROTOCOL=tcp)
(HOST=max.world)(PORT=1521))
```

B.1.1 Further Syntax Rules for Configuration Files

The following rules apply to the syntax of configuration files:

- Any keyword in a configuration file that begins a parameter that includes one or more keyword-value pairs must be in the far left column of a line. If it is

indented by one or more spaces, it is interpreted as a continuation of the previous line.

- All characters must belong to the *network character set* (see the next section).
- Keywords are not case sensitive. Values may be case sensitive, depending on the operating system and protocol.
- Spaces around the "=" sign are optional in keyword-value pairs.
- There is a hierarchy of keywords in that some keywords are always followed by others. At any level of the hierarchy, keywords can be listed in any order. For example, the following entries are equally valid:

```
(ADDRESS =
   (PROTOCOL = TCP)
   (HOST = MARTHA.WORLD)
   (PORT = 1521)
)

(ADDRESS =
   (PROTOCOL = TCP)
   (PORT = 1521)
   (HOST = MARTHA.WORLD)
)
```

- Keywords cannot contain spaces. Values must not contain spaces unless enclosed within double quotes (").
- The maximum length of a connect descriptor is 4 kilobytes.
- Comments can be included using the pound sign # at the beginning of a line. Anything following the sign to the end of the line is considered a comment.
- If the keyword-value pair consists of a single word or a concatenation of words on either side of the equal sign, no parentheses are needed.

B.1.2 Network Character Set

The network character set for keyword values consists of the following characters. Connect descriptors must be made up of single-byte characters.

A-Z, a-z

0-9

( ) < > / \

, . : ; ' " = - _

$ + * # & ! % ? @

Within this character set, the following symbols are reserved:

( ) = \ ” ’ #

Reserved symbols are used as delimiters, not as part of a keyword or a value unless the keyword or value is quoted. Either single or double quotes can be used to enclose a value containing reserved symbols. To include a quote within a value that is surrounded by quotes, use different quote types. The backslash (\) is used as an escape character.

A specific example of the use of reserved symbols is a numeric DECnet object within an address. An OBJECT can be a name such as ABC or #123. These would be entered in the form:

(OBJECT=ABC)

or

(OBJECT=\#123)

Because the “#” sign is a reserved symbol, the character must be preceded by a backslash.

The following characters may be used within a connect descriptor, but not in a keyword or value:

<space> <tab> <CR> <newline>

B.1.3 Service Name Character Set

The listener name, service name, and Oracle Connection Manager names are limited to the following character set:

[a...z] [A...Z] [0...9] _

The first character must be an alphabetical character. The number of characters allowed is platform specific. In general, up to 64 characters is acceptable. A database service name must match the global database name defined by the database administrator, which consists of a database name (originally limited to eight characters), and the database domain. Service names and global database names are not case sensitive.

B.2 Profile Parameters (SQLNET.ORA)

The following parameters are available in a profile. Profiles are stored in a configuration file called SQLNET.ORA.

AUTOMATIC_IPC	
Purpose:	Forces the session to use or not use IPC addresses on your node.
Default Value:	OFF
Available Values:	■ ON - If an IPC address can be used, do so. ■ OFF - Do not use any IPC addresses to connect to
Example:	`automatic_ipc=off`

BEQUEATH_DETACH	
Purpose:	Turn on / off signal handling in Net8 on UNIX systems.
Default Value:	NO which leaves signal handling on.
Available Values:	■ YES - Turns off signal handling ■ NO - Leaves signal handling on
Example:	`bequeath_detach=yes`

DAEMON.TRACE_DIRECTORY	
Purpose:	Controls the destination directory of the Oracle Enterprise Manager daemon trace file.
Default Value:	$ORACLE_HOME / network / trace
Example:	`daemon.trace_directory=/oracle/traces`

DAEMON.TRACE_LEVEL	
Purpose:	Turns tracing on / off to a certain specified level for the Oracle Enterprise Manager daemon.
Default Value:	OFF
Available Values	■ OFF - No trace output ■ USER - User trace information ■ ADMIN - Administration trace information ■ SUPPORT - WorldWide Customer Support trace information
Example:	`daemon.trace_level=user`

DAEMON.TRACE_MASK	
Purpose:	Specifies that only the Oracle Enterprise Manager daemon trace entries are logged into the trace file.
Default Value:	$ORACLE_HOME / network / trace
Example:	`daemon.trace_mask=(106)`

DISABLE_OOB	
Purpose:	Disables out of band breaks.
Default Value:	OFF
Usage Notes:	Disable out of band breaks if the underlying transport protocol does not support the feature.
Example:	`disable_oob=off`

LOG_DIRECTORY_CLIENT	
Purpose:	Controls the directory for where the log file is written.
Default Value:	Current directory where executable is started from.
Example:	`log_directory_client=/oracle/network/log`

LOG_DIRECTORY_SERVER	
Purpose:	Controls the directory for where the log file is written.
Default Value:	Current directory where executable is started from.
Example:	`log_directory_server=/oracle/network/log`

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

LOG_FILE_SERVER	
Purpose:	Controls the log output filename for an Oracle server.
Default Value:	SQLNET.LOG
Example:	`log_file_server=svr`

NAMES.DCE.PREFIX	
Purpose:	Specifies the DCE cell name (prefix) to use for name lookup.
Default Value:	/.:/subsys/oracle/names
Example:	`names.dce.prefix=/.:/subsys/oracle/names`

NAMES.DEFAULT_DOMAIN	
Purpose:	Indicates the domain from which the client most often requests names. When this parameter is set, the default domain name will be automatically appended to any unqualified service name. Any name which contains an unescaped dot ('.') will not have the default domain appended. Simple names may be qualified with a trailing dot (for example, 'rootserver.').
Default Value:	NULL
Example:	`names.default_domain = <valid domain name>`

NAMES.DIRECTORY_PATH

Purpose:	Indicates the order of the Names services, such as TNSNAMES or Oracle Names, that will be used for client name requests.
Default Value:	TNSNAMES, ONAMES, HOSTNAME
Values	■ TNSNAMES ■ ONAMES ■ HOSTNAME ■ DCE ■ NIS ■ NOVELL
Example:	`names.directory_path = (tnsnames, onames)`

NAMES.INITIAL_RETRY_TIMEOUT

Purpose:	Determines how long a client will wait for a response from a Names Server before reiterating the request to the next server in the preferred servers list.
Default Value:	15 (Operating System Dependent)
Minimum Value	1
Maximum Value	600
Example:	`names.initial_retry_timeout=20`

NAMES.MAX_OPEN_CONNECTIONS	
Purpose:	Determines how many connections an Oracle Names client may have open at one time.
Default Value:	10
Minimum Value:	3
Maximum Value:	64
Example:	`names.max_open_connections=3`

NAMES.MESSAGE_POOL_START_SIZE	
Purpose:	Determines the initial number of messages allocated in the client's message pool which are used for forwarded message requests.
Default Value:	10
Minimum Value:	3
Maximum Value:	256
Example:	`names.message_pool_start_size=10`

NAMES.NDS.NAME_CONTEXT	
Purpose:	Specifies the default NDS name context in which to look for the name to be resolved
Default Value:	operating system specific
Example:	`names.nds.name_context=<`*`nds name`*`>`

NAMES.NIS.META_MAP

Purpose:	Specifies the file to be used to map NIS attributes to an NIS mapname.
Default Value:	sqlnet.maps
Example:	`names.nis.meta_map=sqlnet.maps`

NAMES.PREFERRED_SERVERS

Purpose:	Indicates the name, addresses, and order of Names Servers that will be used for a client's name requests.
Default Value:	None

Example:	`names.preferred_servers= (address list=`
	`(address=(protocol=ipc)(key=n23))`
	`(address=(protocol=tcp)(host-nineva)(key=1383))`
	`(address=(protocol=tcp)(host=cicada)(key=1575)))`

NAMES.REQUEST_RETRIES

Purpose:	Specifies the number of times the client should try each server in the list of preferred servers before allowing the operation to fail.
Default Value:	1
Minimum Value:	1
Maximum Value:	5
Example:	`names.request_retries=5`

NAMESCTL.INTERNAL_ENCRYPT_PASSWORD	
Purpose:	If set to TRUE, NAMESCTL will not encrypt the password when it is sent to the Names Server. This enables unencrypted passwords to be set in names.ora: names.server_password.
Default Value:	FALSE
Values	TRUE or FALE
Example:	`namesctl.internal_encrypt_password = true`

NAMESCTL.NO_INITIAL_SERVER	
Purpose:	If set to TRUE, NAMESCTL suppresses any error messages when unable to connect to a default Names Server.
Default Value:	FALSE
Values	TRUE or FALSE
Example:	`namesctl.no_initial_server = true`

NAMESCTL.INTERNAL_USE	
Purpose	If set to TRUE, NAMESCTL enables a set of internal undocumented commands. All internal commands are preceded by an underscore in order to distinguish them as internal.

NAMESCTL.NOCONFIRM	
Purpose:	Indicates whether sensitive commands (STOP, RELOAD, RSTART) should be prompted with a confirmation when running the NAMESCTL utility.
Default Value:	OFF
Values	OFF or ON
Example:	`namesctl.noconfirm = on`

NAMESCTL.SERVER_PASSWORD	
Purpose:	Indicates the value that matches configured password in the Names Server. This eliminates the need to type the password each time you use the NAMESCTL utility.
Example:	`namesctl.server_password=secret`

NAMESCTL.TRACE_LEVEL	
Purpose:	Indicates the level at which the NAMESCTL program should be traced.
Default Value:	OFF
Values:	OFF, USER, ADMIN, SUPPORT
Example:	`namesctl.trace_level=admin`

NAMESCTL.TRACE_FILE	
Purpose:	Indicates the file in which the NAMESCTL trace output is placed.
Default Value:	namesctl_*PID*.trc
Example:	`namesctl.trace_file=nmsctl`

NAMESCTL.TRACE_DIRECTORY	
Purpose:	Indicates the directory where trace output from the NAMESCTL utility is placed.
Default Value:	$ORACLE_HOME/network/trace
Example:	`namesctl.trace_directory=/oracle/trace`

NAMESCTL.TRACE_UNIQUE	
Purpose:	Indicates whether a process identifier is appended to the name of each trace file generated, so that several can co-exist.
Default Value:	ON
Values:	OFF or ON
Example:	`namesctl.trace_unique = on`

OSS.SOURCE.MY_WALLET

Purpose:	Defines the method for retrieving and storing the credentials for this Net8 client.
Example:	`oss.source.my_wallet = (source= (method=oracle) (method_data=/oss/wallet))`

OSS.SOURCE.LOCATION

Purpose:	Defines the method for retrieving encrypted private keys.
Example:	`oss.source.location= (source= (method=oracle) (method_data= (sqlnet_address=oss)))`

SQLNET.AUTHENTICATION_SERVICES	
Purpose:	Enables one or more authentication services. If authentication has been installed, it is recommended that this parameter be set to either NONE or to one of the authentication adapters.
Available Values:	NONE - the database will use the username and password to log in ALL - Enables all authentication adapters to be used. BEQ - always used in conjunction with one or more of the following items: KERBEROS5 - use the Kerberos adaptor SECURID - use SecurID CYBERSAFE - use Cybersafe IDENTIX - use Identix DCEGSSAPI - use DCE GSSAPI
Example:	`sqlnet.authentication_services =(beq, kerberos5, cybersafe)`

SQLNET.AUTHENTICATION_KERBEROS5_SERVICE	
Purpose:	Defines the name of the service used to obtain a Kerberos service ticket
Example:	`sqlnet.authentication_kerberos5_service= oracle`

SQLNET.AUTHENTICATION_GSSAPI_SERVICE	
Purpose:	Defines the CyberSAFE service principal

SQLNET.CLIENT_REGISTRATION

Purpose:	Sets a unique identifier for this client machine. This identifier will be passed to the listener with any connection request and will be included in the Audit Trail. The identifier can be any alphanumeric string up to 128 characters long.
Example:	`sqlnet.client_registration` *`unique_id`*

SQLNET.CRYPTO_CHECKSUM_CLIENT

Purpose:	Specifies the desired checksum behavior when this client is connecting to a server
Default Value:	ACCEPTED
Available Values:	■ ACCEPTED ■ REJECTED ■ REQUESTED ■ REQUIRED
Example:	`sqlnet.crypto_checksum_client=accepted`

SQLNET.CRYPTO_CHECKSUM_SERVER

Purpose:	Specifies the desired checksum behavior when a client is connecting to this server
Default Value:	ACCEPTED
Available Values:	■ ACCEPTED ■ REJECTED ■ REQUESTED ■ REQUIRED
Example:	`sqlnet.crypto_checksum_server=accepted`

SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT	
Purpose:	Specifies a list of crypto-checksum algorithms this client is allowed to use
Default Value:	MD5
Available Values:	MD5 - RSA Data Security's MD5 algorithm
Example:	`sqlnet.crypto_checksum_types_client=(md5)`

SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER	
Purpose:	Specifies a list of crypto-checksum algorithms this server is allowed to use
Default Value:	MD5
Available Values:	MD5 - RSA Data Security's MD5 algorithm
Example:	`sqlnet.crypto_checksum_types_server=(md5)`

SQLNET.CRYPTO_SEED	
Purpose:	Specifies the characters used when generating cryptographic keys. The more random the characters are, the stronger the keys are. This is required whenever encryption or checksumming is turned on.
Example:	`sqlnet.crypto_seed = "4fhfguweotcadsfdsafjkdsfqp5f201p45mxskdl"`

SQLNET.ENCRYPTION_CLIENT

Purpose:	Specifies the desired behavior when this client is connecting to a server
Default Value:	ACCEPTED
Available Values:	■ ACCEPTED ■ REJECTED ■ REQUESTED ■ REQUIRED
Example:	`sqlnet.encryption_client=accepted`

SQLNET.ENCRYPTION_SERVER

Purpose:	Specifies the desired behavior when a client is connecting to this server
Default Value:	ACCEPTED
Available Values:	■ ACCEPTED ■ REJECTED ■ REQUESTED ■ REQUIRED
Example:	`sqlnet.encryption_server=accepted`

SQLNET.ENCRYPTION_TYPES_CLIENT	
Purpose:	Specifies a list of encryption algorithms this client is allowed to use
Default Value:	All algorithms are used if none are specified.
Available Values:	One or more of the following: RC4_40 - RSA RC4 (40 bit key size) Domestic & International RC4_56 - RSA RC4 (56 bit key size) Domestic only RC4_128 - RSA RC4 (128 bit key size) Domestic only DES - Standard DES (56 bit key size) Domestic only DES40 - (40 bit key size) Domestic & International
Example:	`sqlnet.encryption_types_client=(rc4_40)`

SQLNET.ENCRYPTION_TYPES_SERVER	
Purpose:	Specifies a list of encryption algorithms this server is allowed to use when acting as a server
Default Value:	All algorithms are used if none are specified.
Available Values:	RC4_40 - RSA RC4 (40 bit key size) Domestic & International RC4_56 - RSA RC4 (56 bit key size) Domestic only RC4_128 - RSA RC4 (128 bit key size) Domestic only DES - Standard DES (56 bit key size) Domestic only DES40 - (40 bit key size) Domestic & International
Example:	`sqlnet.encryption_types_server=(rc4_40, des, ...)`

SQLNET.EXPIRE_TIME	
Purpose:	Determines time interval to send a probe to verify the session is alive
Default Value:	NONE
Minimum Value:	0 minutes
Recommended Value:	10 minutes
Example:	`sqlnet.expire_time=10`

SQLNET.IDENTIX_FINGERPRINT_DATABASE	
Purpose:	Specifies the service name or alias for the authentication fingerprint database
Example:	`sqlnet.identix_fingerprint_database = service_name`

SQLNET.IDENTIX_FINGERPRINT_DATABASE_USER	
Purpose:	Specifies the well known user name for the fingerprint database
Example:	`sqlnet.identix_fingerprint_database_user = username`

SQLNET.IDENTIX_FINGERPRINT_DATABASE_PASSWORD	
Purpose:	Specifies the well known password for the fingerprint database
Example:	`sqlnet.identix_fingerprint_database_password = password`

SQLNET.IDENTIX_FINGERPRINT_METHOD	
Purpose:	Specifies the method name for the fingerprint database. The method name must be ORACLE
Example:	`sqlnet.identix_fingerprint_method = oracle`

SQLNET.KERBEROS5_CC_NAME	
Purpose:	Specifies the complete pathname to the Kerberos credentials cache file.
Example:	`sqlnet.kerberos5_cc_name= /usr/tmp/krbcache`

SQLNET.KERBEROS5_CLOCKSKEW	
Purpose:	Specifies how many seconds can pass before a Kerberos credential is considered out of date.
Default Value	300
Example:	`sqlnet.kerberos5_clockskew = 1200`

SQLNET.KERBEROS5_CONF	
Purpose:	Specifies the complete pathname to the Kerberos configuration file, which contains the realm for the default KDC and maps realms to KDC hosts.
Default	/krb5/krb.conf
Example:	`sqlnet.kerberos5_conf = /krb5/krb.conf`

SQLNET.KERBEROS5_KEYTAB	
Purpose:	Specifies the complete pathname to the Kerberos principal/secret key mapping file, which is used to extract keys and decrypt incoming authentication information.
Default	/etc/v5srvtab
Example:	`sqlnet.kerberos5_keytab = /etc/v5srvtab`

SQLNET.KERBEROS5_REALMS	
Purpose:	Specifies the complete pathname to the Kerberos realm translation file, which provides a mapping from a host name or domain name to a realm.
Default	/krb5/krb.realms
Example:	`sqlnet.kerberos5_realms= /krb5/krb.realms`

TNSPING.TRACE_DIRECTORY	
Purpose:	Controls the destination directory of the trace file.
Default Value:	$ORACLE_HOME/network/trace
Example:	`tnsping.trace_directory=/oracle/traces`

TNSPING.TRACE_LEVEL	
Purpose:	Turns tracing on / off to a certain specified level.
Default Value:	OFF
Available Values	■ OFF - No trace output ■ USER - User trace information ■ ADMIN - Administration trace information ■ SUPPORT - WorldWide Customer Support trace information
Example:	`tnsping.trace_level=sdmin`

TRACE_DIRECTORY_CLIENT	
Purpose:	Controls the destination directory of the trace file.
Default Value:	$ORACLE_HOME / network / trace
Example:	`trace_directory_client=/oracle/traces`

TRACE_DIRECTORY_SERVER	
Purpose:	Controls the destination directory of the trace file.
Default Value:	$ORACLE_HOME / network / trace
Example:	`trace_directory_server=/oracle/traces`

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

TRACE_FILE_SERVER	
Purpose:	Controls the name of the server trace file.
Default Value:	SVR_*PID*.TRC
Example:	`trace_file_server=svr`

TRACE_LEVEL_CLIENT	
Purpose:	Turns tracing on / off to a certain specified level.
Default Value:	OFF
Available Values	■ OFF - No trace output ■ USER - User trace information ■ ADMIN - Administration trace information ■ SUPPORT - WorldWide Customer Support trace information
Example:	`trace_level_client=user`

TRACE_LEVEL_SERVER	
Purpose:	Turns tracing on / off to a certain specified level.
Default Value:	OFF
Available Values	■ OFF - No trace output ■ USER - User trace information ■ ADMIN - Administration trace information ■ SUPPORT - WorldWide Customer Support trace information
Example:	`trace_level_server=admin`

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

USE_CMAN	
Purpose:	Forces all sessions to go through Oracle Connection Manager to get to the server.
Default Value:	FALSE
Values:	TRUE or FALSE
Example:	`use_cman=true`

USE_DEDICATED_SERVER	
Purpose:	Forces the listener to spawn a dedicated server process for sessions from this client.
Default Value:	OFF
Values:	■ ON - spawn dedicated server processes ■ OFF - hand off to existing server processes
Example:	`use_dedicated_server=on`

B.3 Local Naming Parameters (TNSNAMES.ORA)

The following name-value pairs are available in a local names configuration file. Local naming configurations are stored in a file called TNSNAMES.ORA.

DESCRIPTION	
Purpose:	Beginning of a definition of a database listening address.

Example:	`service name= (description =`
	`  (address = ...)`
	`  (connect_data=(sid=db1))`

SOURCE_ROUTE	
Purpose:	Creates a source route of addresses through all Connection Managers to the destination database.
Values:	YES or NO

Example:	`(description =`
	`  (address_list =`
	`  (address = ...)`
	`  (address = ...)`
	`  )`
	`  (connect_data=(sid=db1))`
	`  (source_route = yes))`

B.4 Listener Parameters (LISTENER.ORA)

The following parameters are available in the network listener configuration film (LISTENER.ORA).

CONNECT_TIMEOUT_*listener_name*	
Purpose:	Sets the number of seconds that the listener waits to get a valid database query after the session has started.
Default	10
Comment:	If value is set to 0, it will wait forever.

LISTENER_*address*	
Purpose:	Defines the listening addresses for the listener.

Default Value:	`(address_list=`
	`(address=(protocol=tcp)(host=<host>)(port=1521))`
	`(address=(protocol=ipc)(key=pnpkey)))`

Example:	`listener_name = (address_list=`
	`(address=(address)(address=(address))`
	`(address=(address)(address=(address))`
	`)`

LOG_DIRECTORY_*listener_name*	
Purpose:	Controls the directory for where the log file is written.
Default Value:	Current directory where executable is started from.
Example:	`log_directory_listener=/oracle/traces`

LOG_FILE_*listener_name*	
Purpose:	Specifies the filename where the log information is written
Default Value:	listener_name.log
Example:	`log_file_listener=lsnr`

LOGGING_*listener_name*	
Purpose:	Logging is always on unless you provide this parameter and turn logging off.
Default Value:	ON
Available Values	ON or OFF
Example:	`logging_listener=off`

PASSWORDS_*listener_name*	
Purpose:	Specifies a password to perform certain DBA tasks against the listener using LSNRCTL.
Default Value:	NULL
Example:	`passwords_listener=(super32, sly51)`

SAVE_CONFIG_ON_STOP_*listener_name*	
Purpose:	Any changes made by the LSNRCTL SET command is made permanent if parameter is TRUE.
Default Value:	OFF
Values:	ON or OFF
Example:	`save_config_on_stop_listener=true`

SERVICE_LIST_*listener_name*	
Purpose:	Defines the service served by the listener. This is the same as the SID_LIST, made more generic for non-database servers.
Default Value:	NULL
Usage Notes:	The one difference between SERVICE_LIST and SID_LIST is that it cannot be used for prespawned services.

Example:	`service_list_`*`listener_name`* `=`
	`  (service_list= (service=`
	`  (global_dbname=`*`global_database_name`*`)`
	`  (service_name=`*`service_name`*`)`
	`  (`*`operating_system_specific=db_location`*`) [(service=...)`
	`)`

SID_LIST_*listener_name*

Purpose: Defines the SID of the databases served by the listener.

Example:

```
sid_list_listener_name =
  (sid_list= (sid_desc=
    (global_dbname=global_database_name)
    (sid_name=sid)
    (operating_system_specific=db_location)
    [(prespawn_max=99) (prespawn_list=
         (prespawn_desc= (protocol=tcp)
            (pool_size=10)(timeout=2))
         (prespawn_desc=...))]
  [(sid_desc=...)]
)
```

STARTUP_WAIT_TIME_*listener_name*

Purpose: Sets the number of seconds that the listener sleeps before responding to the first LSNRCTL STATUS command. This assures that a listener with a slow protocol will have time to start up before responding to a status request.

Default Value: 0

Example:

```
startup_wait_time_listener=2
```

TRACE_DIRECTORY_*listener_name*

Purpose: Controls the destination directory of the trace file.

Default Value: $ORACLE_HOME/network/trace

Example:

```
trace_directory_listener=/oracle/traces
```

TRACE_FILE_*listener_name*	
Purpose:	Controls the name of the listener trace file.
Default Value:	LISTENER_NAME.TRC
Example:	`trace_file_listener=lsnr`

TRACE_LEVEL_*listener_name*	
Purpose:	Turns tracing on / off to a certain specified level.
Default Value:	OFF
Available Values	■ OFF - No trace output ■ USER - User trace information ■ ADMIN - Administration trace information ■ SUPPORT - WorldWide Customer Support trace information
Example:	`trace_level_listener=admin`

USE_PLUG_AND_PLAY_*listener_name*	
Purpose:	Instructs the listener to register with a well-known Names Server. Will continue to look for a well-known Names Server until one is found.
Default Value:	OFF
Available Values	ON or OFF
Usage Notes	If no well-known Names Server exists in the environment, this parameter should be set to OFF.
Example:	`use_plug_and_play=on`

B.5 Oracle Names Parameters (NAMES.ORA)

The following parameters are available in an Oracle Names configuration file (NAMES.ORA).

NAMES.ADDRESSES	
Purpose:	Describes the address on which the Names Server listens. Any valid TNS address is allowed.
Default Value:	`names.addresses =` `(address=(protocol=tcp)(host=oranamesrvr0)(port=1575)`

NAMES.ADMIN_REGION	
Purpose:	Describes the data source for an administrative region. This parameter defines a database as a repository for information, instead of relying on replication of data between Names Server caches.
Default Value:	NULL

Example:	`names.admin_region=`
	`(region=(name=local_region.world)`
	`(type=rosdb)`
	`(userid=names)`
	`(password=names)`
	`(description=`
	`(address_list=`
	`(address=(protocol=tcp)`
	`(host=nineva)(port=1575)`
	`)`
	`)`
	`(connect_data=`
	`(sid=em)`
	`)`
	`)`
	`(version=34619392)`
	`(refresh=14400)`
	`(retry=600)`
	`(expire=259200)`

NAMES.AUTHORITY_REQUIRED

Purpose:	Determines whether system queries require Authoritative answers.
Default Value:	FALSE
Example:	`names.authority_required=true`

NAMES.AUTO_REFRESH_EXPIRE

Purpose:	Specifies the amount of time in seconds the Names Server caches other region's database server addresses which have been obtained through "NAMES.HINTS". At the end of this interval, the Names Server issues a query to the other region database servers to refresh the address.
Default Value:	600 seconds
Acceptable Values:	60-1209600 seconds
Example:	`names.auto_refresh_expire=1200000`

NAMES.AUTO_REFRESH_RETRY

Purpose:	Specifies the interval in seconds that the Names Server will retry when the auto refresh query fails.
Default Value:	180
Minimum Value:	60
Maximum Value	3600
Example:	`names.auto_refresh_retry=180`

NAMES.CACHE_CHECKPOINT_FILE	
Purpose:	Specifies the name of the operating system file to which the Names Server writes its checkpoint file. See the Oracle operating system-specific manual for your platform for filename restrictions.
Default Value:	ckpcch.ora
Example:	`names.cache_checkpoint_file = filename`

NAMES.CACHE_CHECKPOINT_INTERVAL	
Purpose:	Indicates the interval at which a Names Server writes a checkpoint of its stored data to the checkpoint file. Each Names Server can periodically write its cached data to a file to protect against startup failures.
Default Value:	0 (disabled)
Minimum Value:	10
Maximum Value:	259200 (3 days)
Example:	`names.cache_checkpoint_interval = seconds`

NAMES.CONFIG_CHECKPOINT_FILE	
Purpose:	Name of the file used to checkpoint configuration settings.
Default Value:	$ORACLE_HOME/NETWORK/NAMES/CKPCFG.ORA
Example:	`names.config_checkpoint_file=cache.ckp`

NAMES.DEFAULT_FORWARDERS

Purpose:	Address list of other Names Servers which are used to forward queries.
Example:	`names.default_forwarders= (forwarder_list= (forwarder=(name=rootserv1.world)(address=(protocol=tcp) (port=4200)(host=roothost))))`

NAMES.DEFAULT_FORWARDERS_ONLY

Purpose:	When TRUE the Names Server forwards queries only to those Names Servers listed as default forwarders.
Default Value:	FALSE
Usage Notes:	If set to FALSE, Names Servers listed as default forwarders will be called before Names Servers specified in the cache.

NAMES.DOMAIN_HINTS

Purpose:	Lists the names, addresses and domains of all servers in one or more foreign regions. Enables the Names Server to know about other regions' servers. This includes at least the "root" region for all servers who are not in the root region. Other regions may be provided as optimization requires.
Example:	`names.domain_hints= (hint_desc=(hint_list= (hint=(name=rootserv1.world) (address=(protocol=tcp)(host=nineva)(port=4200) ) ) )`

NAMES.DOMAINS	
Purpose:	List of domains in the server's local region, as well as the default time to live (TTL) for data in those domains.

Example:	`names.domains=`
	`(domain_list=`
	`(domain=(name=)(min_ttl=86400))`
	`(domain=(name=world)(min_ttl=8640))`
	`)`

NAMES.DOMAINS_DNS_ALIGNED	
Purpose:	The Names Server will use the local default DNS domain as its default domain, if there are no domains set in names.domains, and if this parameter is not set to FALSE.
Default Value:	TRUE
Example:	`names.domains_dns_aligned=false`

NAMES.FORWARDING_AVAILABLE	
Purpose:	The server forwards operations to foreign regions if set to TRUE.
Default Value	TRUE
Example:	`names.forwarding_available=false`

NAMES.FORWARDING_DESIRED	
Purpose:	If set to true, then this server will forward queries.
Default Value:	TRUE
Example:	`names.forwarding_desired=true`

NAMES.LOG_DIRECTORY

Purpose:	Indicates the name of the directory where the log file for Names Server operational events are written. For the default name of this file on your system, refer to the Oracle operating system-specific manual for your platform.
Default Value:	platform specific
Example:	`names.log_directory = complete_directory_name`

NAMES.LOG_FILE

Purpose:	Indicates the name of the output file to which Names Server operational events are written. The file name extension is always.log. Do not enter an extension for this parameter.
Default Value:	names
Example:	`names.log_file = filename`

NAMES.LOG_STATS_INTERVAL

Purpose:	Specifies the number of seconds between statistical entries in log file.
Default Value:	0 (0 = OFF)
Minimum Value:	10 (least possible ON value)
Maximum Value:	none
Example:	`names.log_stats_interval = seconds`

NAMES.LOG_UNIQUE	
Purpose:	If set to true, then the log filename will be unique and not overwrite existing log files.
Default Value:	FALSE
Example:	`names.log_unique=true`

NAMES.MAX_OPEN_CONNECTIONS	
Purpose:	Specifies the number of connections that the Names Server can have open at any given time. The value is generated as the value 10 or the sum of one connection for listening, five for clients, plus one for each foreign domain defined in the local administrative region, whichever is greater. The calculated value is acceptable for most installations.
Default Value:	Calculated based on entered data
Minimum Value:	2
Maximum Value:	64
Example:	`names.max_open_connections = number`

NAMES.MAX_REFORWARDS	
Purpose:	The maximum number of times the server attempts to forward an operation.
Default Value:	2
Minimum Value:	1
Maximum Value:	15
Example:	`names.max_reforwards=2`

NAMES.MESSAGE_POOL_START_SIZE

Purpose:	Determines the initial number of messages allocated in the server's message pool which are used for incoming or outgoing forwarded messages.
Default Value:	10
Minimum Value:	3
Maximum Value:	256
Example:	`names.message_pool_start_size=10`

NAMES.NO_MODIFY_REQUESTS

Purpose:	If set to true, the server refuses any operations which modify the data in its region.
Default Value:	FALSE
Example:	`names.no_modify_requests=true`

NAMES.NO_REGION_DATABASE

Purpose:	If set to true, the server won't look for a region database.
Default Value:	FALSE
Example:	`names.no_region_database=true`

NAMES.PASSWORD	
Purpose:	The Names Server may require a user password (if set) in the namesctl session in order to do 'sensitive' operations, such as STOP, RESTART, RELOAD.
Default Value:	None
Example:	`names.password=625926683431aa55`

NAMES.REGION_CHECKPOINT_FILE	
Purpose:	Specifies the name of the file used to checkpoint region data (for example, domain addresses, database addresses of Names Servers in the local region)
Default Value:	$ORACLE/HOME/network/names/ckpreg.ora
Example:	`names.region_checkpoint_file=oracle/home/network/names/ckpreg.ora`

NAMES.RESET_STATS_INTERVAL	
Purpose:	Specifies the number of seconds during which the statistics collected by the Names Servers should accumulate. At the frequency specified, they are reset to zero. The default value of 0 means never reset statistics.
Default Value:	0 (never reset)
Minimum Value:	10
Maximum Value:	none
Example:	`names.reset_stats_interval =` *seconds*

NAMES.SERVER_NAME	
Purpose:	Each Names Server is uniquely identified by a name. All configuration references to a particular Names Server use this name.
Default Value:	ONAMES_*hostname*
Example:	`names.server_name = valid_string`

NAMES.TRACE_DIRECTORY	
Purpose:	Indicates the name of the directory to which trace files from a Names Server trace session are written.
Default Value:	platform specific
Example:	`names.trace_directory = complete_directory_name`

NAMES.TRACE_FILE	
Purpose:	Indicates the name of the output file from a Names Server trace session. The filename extension is always.trc.
Default Value:	names
Example:	`names.trace_file = filename`

NAMES.TRACE_FUNC	
Purpose:	Enables internal mechanism to control tracing by function name.
Default Value:	FALSE
Example:	`names.trace_func = false`

NAMES.TRACE_LEVEL	
Purpose:	Indicates the level at which the Names Server is to be traced.
Default Value:	OFF
Available Values	■ OFF - No trace output ■ USER - User trace information ■ ADMIN - Administration trace information ■ SUPPORT - WorldWide Customer Support trace information
Example:	`names.trace_level = admin`

NAMES.TRACE_UNIQUE	
Purpose:	indicates whether each trace file has a unique name, allowing multiple trace files to coexist. If the value is set to ON, a process identifier is appended to the name of each trace file generated.
Default Value:	ON
Example:	`names.trace_unique = on`

B.6 Oracle Connection Manager Parameters (CMAN.ORA)

The following parameters are available in an Oracle Connection Manager configuration file (CMAN.ORA).

CMAN

Purpose: Specifies listening addresses for Oracle Connection Manager.

Default Value: `cman = (address=(protocol=tcp)(host=anyhost)(port=1600))`

Example:

```
cman=
   (address_list=
       (address= ...)
     [(address = ...)]
    )
```

CMAN_PROFILE

Purpose: Sets parameters related to the running of Oracle Connection Manager.

Default Value:

- MAXIMUM_RELAYS = 8
- LOG_LEVEL = 0
- TRACING = NO
- TRACE_DIRECTORY = <pathname>
- RELAY_STATISTICS = NO
- SHOW_TNS_INFO = NO
- USE_ASYNC_CALL = YES
- AUTHENTICATION_LEVEL = 0
- MAXIMUM_CONNECT_DATA = 1024
- ANSWER_TIMEOUT = 0

CMAN_PROFILE

Available Values:

- MAXIMUM_RELAYS = [0-10240]
- LOG_LEVEL = [0-4]
- TRACING = [YES, NO]
- TRACE_DIRECTORY = <pathname>
- RELAY_STATISTICS = [YES,NO]
- SHOW_TNS_INFO = [YES, NO]
- USE_ASYNC_CALL = [YES, NO]
- AUTHENTICATION_LEVEL = [0, 1]
- MAXIMUM_CONNECT_DATA = [257-4096]
- ANSWER_TIMEOUT = [0 to n]

Usage Notes:

- MAXIMUM_RELAYS = determines the maximum number of concurrent connections allowed.
- LOG_LEVEL = determines the level of logging performed by the Connection Manager.
- TRACING = enables tracing on the Connection Manager.
- TRACE_DIRECTORY = indicates the location where trace information specific to the Connection Manager is written.
- RELAY_STATISTICS = instructs the Connection Manager to maintain statistics pertaining to relay I/O activities such as number of IN bytes, number of OUT bytes, number of IN packets, number of out packets.
- SHOW_TNS_INFO = instructs the Connection Manager to include TNS events in the log file.
- USE_ASYNC_CALL = instructs the Conenction Manager to use all asynchronous functions while in the answering, accepting, or calling phase of establishing a connection.

  NOTE: CMAN supports out-of-band breaks, it will forward it on to the server.
- AUTHENTICATION_LEVEL instructs the Connection Manager to reject connect requests that are not using Secure Network Services. Secure Network Services is part of the Advanced Networking Option.

CMAN_PROFILE

Example:

```
cman_profile =
   (parameter_list=
      (maximum_relays=512)
      (log_level=1)
      (tracing=yes)
      (relay_statistics=yes)
      (show_tns_info=yes) (use_async_call=yes)
      (authentication_level=0)
   )
```

CMAN_RULES

Purpose: Sets the rules for the network access control portion of Oracle Connection Manager.

Values:

- SRC - Source host name or IP address (in dot notation) of session request
- DST - Destination server host name or IP address (in dot notation)
- SRV - Database server SID
- ACT - Accept or Reject incoming requests with the previous characteristics.

Usage Notes: The wildcard for host name is the single character ‘x’. In the case of an IP address (d.d.d.d), you may wild card the individual d’s with an ‘x’.

Example:

```
cman_rules=
  (rules_list=
      (rule=(src=hostname)(dst=hostname)(srv=sid)
      (act=accept))
      [(rule= ...)]
   )
```

B.7 Protocol-specific Parameters (PROTOCOL.ORA)

The following parameters in the PROTOCOL.ORA configuration file are applicable to Net8.

protocol.EXCLUDED_NODES	
Purpose:	Defines which nodes do not have validnode checking.
Example:	`tcp.excluded_nodes= (hostname or tcp_address, ...)`

protocol.INVITED_NODES	
Purpose:	Defines which nodes have validnode checking.
Example:	`tcp.invited_nodes= (hostname or tcp_address, ...)`

protocol.VALIDNODE_CHECKING	
Purpose:	Restricts session access of clients to destinations with enabling host privilege.
Default Value:	NO
Values:	■ YES ■ NO
Example:	`tcp.validnode_checking = yes`

C

Sample Configuration Files

This appendix provides sample configuration files used in Net8. These files include:

- Section C.1, “Profile (SQLNET.ORA)”
- Section C.2, “Local Naming Configuration File (TNSNAMES.ORA)”
- Section C.3, “Listener Configuration File (LISTENER.ORA)”
- Section C.4, “Names Server Configuration File (NAMES.ORA)”
- Section C.5, “Oracle Connection Manager Configuration File (CMAN.ORA)”

This appendix is intended for use as a reference only.

C.1 Profile (SQLNET.ORA)

A profile (SQLNET.ORA) contains the parameters that specify preferences for how a client or server uses Net8 features. For more information on each individual parameter, refer to "Profile Parameters (SQLNET.ORA)", in Appendix B, "Configuration Parameters".

```
names.default_domain = world
names.initial_retry_timeout = 30
names.max_open_connections = 3
names.message_pool_start_size = 10
names.preferred_servers = (address_list =
 (address=(protocol=ipc)(key=n23))
  (address=(protocol=tcp)(host=nineva)(port=1383))
  (address=(protocol=tcp)(host=cicada)(port=1575))
 )
names.request_retries = 2
names.directory_path = (tnsnames, onames, hostname)
namesctl.trace_directory = /oracle/network/trace
namesctl.trace_file = namesctl.trc
namesctl.trace_level = admin
namesctl.trace_unique = true
namesctl.no_initial_server = false
namesctl.internal_use = true
namesctl.noconfirm = true
namesctl.server_password = mangler
namesctl.internal_encrypt_password = false
names.dce.prefix = /.:/subsys/oracle/names
names.nds.name_context = personnel.acme
names.nis.meta_map=sqlnet.maps
sqlnet.authentication_services=(none)
sqlnet.authentication_services=(beq, oss)
sqlnet.kerberos5_cc_name=/tmp/mycc
sqlnet.kerberos5_clockskew=600
sqlnet.kerberos5_conf=/tmp/mykrb.conf
sqlnet.kerberos5_realms=/tmp/mykrb.realms
sqlnet.kerberos5_keytab=/tmp/myv5srvtab
sqlnet.authentication_kerberos5_service=acme
sqlnet.authentication_gssapi_service=acme/asriniva.us.oracle.com@us.oracle.com
sqlnet.identix_fingerprint_method=oracle
sqlnet.identix_fingerprint_database=ofm
sqlnet.identix_fingerprint_database_user=<username>
sqlnet.identix_fingerprint_database_password=<password>
sqlnet.authentication_gssapi_service=acme/scott.us.oracle.com@us.oracle.com
```

```
oss.source.my_wallet
   =(source
       =(method=file)
        (method_data=/dve/asriniva/oss/wallet)
        )
oss.source.encrypted_private_key
   =(source
       =(method=oracle)
        (method_data=
           (username=andre_security_service)
           (password=andre_security_service)
           (sqlnet_address=andreoss)
        )
    )
oss.source.certificates
   =(source
       =(method=oracle)
        (method_data=
           (username=scott_security_service)
           (password=ascott_security_service)
           (sqlnet_address=andreoss)
        )
    )
oss.source.attributes
   =(source
       =(method=oracle)
        (method_data=
           (username=scott_oracle_security_service)
           (password=scott_oracle_security_service)
           (sqlnet_address=andreoss)
        )
    )
sqlnet.crypto_checksum_client = required
sqlnet.encryption_client = required
sqlnet.crypto_checksum_types_client = required
sqlnet.crypto_checksum_types_server = required
sqlnet.encryption_types_client = required
sqlnet.encryption_types_server = required
sqlnet.crypto_seed ="4fhfguweotcadsfdsafjkdsfqp5f201p45mxskdlfdasf"
sqlnet.crypto_checksum_server = required
sqlnet.encryption_server = required
trace_level_client = admin
trace_directory_client = /oracle/network/trace
trace_file_client =  /oracle/network/trace/cli.trc
trace_unique_client = on
```

```
log_directory_client = /oracle/network/log
log_file_client = /oracle/network/log/sqlnet.log
log_directory_server = /oracle/network/trace
trace_directory_server = /oracle/network/trace
trace_file_server = /orace/network/trace/svr_<pid>.trc
trace_level_server = admin
use_dedicated_server = on
use_cman = true
tnsping.trace_directory = /oracle/network/trace
tnsping.trace_level = admin
sqlnet.expire_time = 10
sqlnet.client_registration = <unique_id>
bequeath_detach = yes
automatic_ipc = off
disable_oob = on
```

C.2 Local Naming Configuration File (TNSNAMES.ORA)

The local naming configuration file (TNSNAMES.ORA) contains the names and addresses of services on the network. The service names of databases are mapped to connect descriptors that describe their location on the network. The local naming configuration file (TNSNAMES.ORA) is used by clients and distributed database servers to identify destinations and servers. For more information on each individual parameter, refer to "Local Naming Parameters (TNSNAMES.ORA)", in Appendix B, "Configuration Parameters".

```
tcpnew1 = (description=
              (address_list=
                  (address=(protocol=tcp)(port=1610)(host=spcstn))
                  (address=(protocol=tcp)(port=1580)(host=spcstn)))
              (connect_data=(sid=cman))
              (source_route=yes)
          )

spx2tcp  = (description=
              (address_list=
                  (address=(protocol=spx)(service=orasrvc1))
                  (address=(protocol=tcp)(port=1580)(host=spcstn)))
              (connect_data=(sid=cman))
              (source_route=yes)
          )
```

C.3 Listener Configuration File (LISTENER.ORA)

The listener configuration file (LISTENER.ORA) contains the parameters that specify preferences for how a network listener behaves. For more information on each individual parameter, refer to "Listener Parameters (LISTENER.ORA)", in Appendix B, "Configuration Parameters".

```
listener=(address_list=
    (address= # default tcp listening address
        (protocol=tcp)
        (port=1521)
        (host=mudshark)
    )
    (address= # non-default ipc listening address
        (protocol=ipc)
        (key=salesdb)
    )
)
sid_list_listener=(sid_list=
    (sid_desc=
        (sid_name=sales)
        (global_dbname=salesdb.mycompany)
        (oracle_home=/private1/app/oracle/product/8.0.3)
        (prespawn_max=20)
        (prespawn_list=
            (prespawn_desc=(protocol=tcp)(pool_size=2)(timeout=5))
            (prespawn_desc=(protocol=ipc)(pool_size=3)(timeout=2))
        )
    )
)
trace_level_listener=admin
trace_directory_listener=/private1/app/oracle/product/8.0.2/network/trace
trace_file_listener=listener
logging_listener=on
log_directory_listener=/private1/app/oracle/product/8.0.2/network/log
log_file_listener=listener
save_config_on_stop_listener=true
startup_wait_time_listener=0
```

C.4 Names Server Configuration File (NAMES.ORA)

The Names Server configuration file (NAMES.ORA) contains the parameters that specify preferences for each Names Server. For more information on each individual parameter, refer to "Oracle Names Parameters (NAMES.ORA)", in Appendix B, "Configuration Parameters".

```
names.addresses = (address=(protocol=tcp)(host=oranamesrvr0)(port=1575))
names.server_name = ons1.world
names.domains = (domain_list=
                  (domain=
                     (name=world)
                     (min_ttl=86400)
                  )
                  (domain=
                     (name=lostworld)
                     (min_ttl=86400)
                  )
               )
names.admin_region= (region=
    (name= local_region.world)
    (type= rosdb)
    (userid = names)
    (password = names)
    (description =
            (address = (protocol = tcp)(host = nineva)(port = 1387))
            (connect_data = (sid = em))
    )
    (docname = sbox)
    (version = 34619392)  2.1.4
    (refresh = 14400)
    (retry = 600)
    (expire = 259200)
)
names.authority_required = false
names.auto_refresh_expire = 259200
names.auto_refresh_retry = 180
names.cache_checkpoint_file = cache.ckp
names.cache_checkpoint_interval = 7200
names.config_checkpoint_file = cache.ckp
names.default_forwarders=
 (forwarder_list=
    (forwarder=
       (name= rootserv1.world)
       (address=(protocol=tcp)(port=42100)(host=roothost))))
```

```
names.default_forwarders_only = true
names.domain_hints =
 (hint_desc=
    (hint_list=
        (hint=(name=rootserv1.world)
              (address=(protocol=tcp)(host=nineva)(port=42100)))))
names.domains_dns_aligned = na
names.forwarding_available = true
names.forwarding_desired = true
names.log_directory = /oracle/network/log
names.log_file = names.log
names.log_stats_interval = 3600
names.log_unique = false
names.max_open_connections = 10
names.max_reforwards = 2
names.message_pool_start_size = 24
names.no_modify_requests = false
names.no_region_database = false
names.password = 625926683431aa55
names.reset_stats_interval = 3600
names.region_checkpoint_file = reg.ckp
names.trace_directory = /oracle/network/trace
names.trace_file = names.trc
names.trace_level = admin
names.trace_unique = true
```

C.5 Oracle Connection Manager Configuration File (CMAN.ORA)

The Connection Manager configuration file (CMAN.ORA) contains the parameters that specify preferences for using Oracle Connection Manager. For more information on each individual parameter, refer to "Oracle Connection Manager Parameters (CMAN.ORA)", in Appendix B, "Configuration Parameters".

```
cman = (address_list=
           (address = (protocol=tcp)(host=anyhost)(port=1610))
           (address = (protocol=tcp)(host=anyhost)(port=1620))
)
cman_profile = (parameter_list=
        (maximum_relays=512)
        (log_level=1)
        (tracing=yes)
        (trace_directory=/oracle/network/trace)
        (relay_statistics=yes)
        (show_tns_info=yes)
        (use_async_call=yes)
        (authentication_level=0)
)
# the following specifies a rule for single access control #
cman_rules = (rule_list=
        (rule=(src=spcstn)(dst=x)(srv=x)(act=accept))
)
```

D

Native Naming Adapters

This appendix provides details about the configuration required by Native Naming Adapters. If you are using an external naming service in your network and want the Oracle clients and servers in the network to use the same service, follow the instructions for the service you use. The following Native Naming Adapters are discussed:

- Section D.1, "NIS"
- Section D.2, "NDS"

> **Note:** For information about the CDS naming adapter, which is part of the DCE Integration component of Oracle Advanced Networking Option, refer to the *Oracle Advanced Networking Option Administrator's Guide.*

D.1 NIS

Organizations and corporations already using Network Information Service (NIS) as part of their systems infrastructure have the option to store Oracle service names and addresses in NIS, using the NIS Native Naming Adapter.

D.1.1 System Requirements

The NIS Naming Adapter requires SQL*Net 2.2 or greater.

D.1.2 How the NIS Naming Adapter Interacts with SQL*Net and Oracle

When a user gives a command such as

```
sqlplus scott/tiger@payroll
```

(where "payroll" is an Oracle service name) the NIS Naming Adapter on the node running the client program (or server acting as a client program) contacts an NIS server located somewhere in the network, and passes the service name to the NIS server. The NIS server resolves the service name into a SQL*Net address and returns this address to the client program (or server acting as a client program). The client program then uses this address to connect to the Oracle database.

Note: The service name should be the same as the global database name defined in the server's `INIT.ORA` file.

D.1.3 Oracle Database Service Names are Stored in a Separate NIS Map

A machine that acts as an NIS server runs a program called ypserv, which handles name requests. ypserv stores different types of data in special files called "maps". For example, passwords are stored in a map called passwd.byname. Oracle database service names are stored in a map called tnsnames.

Note: The `tnsnames` map is called a.smd in releases previous to SQL*Net 2.2.2.

When a user issues a command like the one in the previous section, the NIS Naming Adapter uses an RPC call to contact the `ypserv` program and passes the Oracle service name "payroll" and the name of the map—`tnsnames`. The `ypserv` program looks in the `tnsnames` map for the name "payroll" and its corresponding value, which is the address for the service name. The address is returned to the client, and the client program (or server acting as a client program) uses this address to contact the database server.

D.1.4 Configuring NIS Servers to Support the NIS Adapter

Before configuring servers to support the NIS Naming Adapter, make sure that NIS is configured and running on the NIS servers that need to resolve Oracle database service names. Consult your NIS documentation for specifics.

D.1.4.1 Add the tnsnames Map to the Existing Set of NIS Maps

To add the `tnsnames` map to the existing set of NIS maps:

1. Use the Oracle Net8 Assistant to create a TNSNAMES.ORA file.

   Note: Keep a copy of the TNSNAMES.ORA file, preferably in your $TNS_ADMIN or $ORACLE_HOME/network/admin directory. You may need to use this file again later to load Oracle service names into the NIS map.

2. Convert the contents of the TNSNAMES.ORA file to `tnsnames` using the `tns2nis` program.

   Note: The `tns2nis` program is supplied with the NIS adapter on the Oracle Installer tape or disk.

   For example, run `tns2nis` on the command line with one argument:

   ```
   tns2nis tnsnames.ora
   ```

   tns2nis reads the NATIVE.ORA file from the current directory. (If TNSNAMES.ORA is not located in the current directory, you can use a full pathname to specify its location—for example, /etc/native.ora or $ORACLE_HOME/network/admin/tnsnames.ora).

   tnsnames is then written into the current working directory.

3. Copy `tnsnames` to the NIS server, if it is not already there.

4. Install the `tnsnames` map using `makedbm`, which is an NIS program. Refer to your NIS documentation for more information.

   Note: This step should be performed by the person in charge of NIS administration.

   `makedbm` converts `tnsnames` to two files that the NIS server can read. The location of these files is platform-specific. Refer to your platform-specific documentation for details.

   For example, as "root", a command of the form:

   `# makedbm tnsnames /var/yp/'domainname'/tnsnames`
   generates and installs the `tnsnames` map in the appropriate directory on the SunOS.

D.1.4.2 Verifying that the tnsnames Map Has Been Properly Installed

You can test the NIS server to see if the map has been installed properly by typing a command with the format:

`ypmatch` *`global_database_name`* `tnsnames`
For example, you might enter:

```
ypmatch payroll.world tnsnames
```
This returns the length of the address (in characters) followed by the address; for example:

```
99 (DESCRIPTION = (ADDRESS = (PROTOCOL = TCP)
  (HOST = garlic)(PORT = 1999)))
  (CONNECT_DATA=(SID=DIRPROD)))
```

D.2 NDS

The NDS Naming Adapter allows you to use native NDS naming conventions to connect to an Oracle database on a Novell NDS-enabled network. After the NDS Naming Adapter has been installed on clients and servers, users can type

`username/password@service_name`
The NDS Naming Adapter provides network users with the following benefits:

- Allows clients to use simple NDS names (partial or full) when connecting to a database.
- Simplifies the maintenance of the TNS addresses; one change will affect all clients using the NDS Naming Adapter.

- Reduces network traffic by eliminating the need for the listener to advertise.

D.2.1 How the NDS Adapter Interacts with SQL*Net and Oracle

This section provides a brief discussion of how the NDS Naming Adapter interacts with SQL*Net and an Oracle Server.

D.2.1.1 What the Client Does

The NDS Naming Adapter resides on the client workstation and translates the NDS object name into a TNS address. The client code gets attributes from the NDS tree for the NDS object whose name matches the Oracle service name. This name can be a full name or a partial name. If it is a partial name, it will be qualified with respect to the *current name context*. Refer to "Optional Configuration File Parameter for the Client" in this chapter for information on fully-qualified and partially-qualified NDS names.

D.2.1.2 What the Server Does

There are three aspects to the server-side NDS Naming Adapter: schema extension, SAP disabling, and listener address storage.

D.2.1.2.1 Schema Extension During the Oracle installation process on a NetWare 4 server the NDS schema is extended to include an object class called "ORACLE:DBInstance". For the NDS Naming Adapter to function, this class will need an attribute called "ORACLE:TNSAddress". If the class does not exist, it is created and will include the TNSAddress attribute. If the class exists but the TNSAddress attribute does not, the class will be modified. The NLM that performs this during installation is called ORASCHEM.NLM.

D.2.1.2.2 SAP (Service Address Protocol) Disabling The Oracle SPX Protocol Adapter for NetWare looks for a value in CONFIG.ORA called "ORACLE_SAP". The value of this parameter is ON or OFF. If this parameter is not specified, the default is ON. This has performance implications for SPX networks. When ORACLE_SAP is ON, the SQL*Net listener advertises its address using SAP (Service Advertising Protocol). When ORACLE_SAP is set to OFF, the SQL*Net listener relies on NDS to deliver information to its clients.

If your network consists entirely of NDS enabled clients (that is, clients using NetWare 4 and above), you will get better network performance if you edit the CONFIG.ORA file to set ORACLE_SAP to OFF.

D.2.1.2.3 How Oracle Service Names and Addresses are Stored in NDS

When the network listener is started, it stores its address in the NDS database by locating the Oracle database instance that resides on its server.

> **Note:** An NDS object for the Oracle database must have already been created.

At that point, the address(es) is (are) accessible to the client from the NDS database

D.2.2 System Requirements

The NDS Naming Adapter requires SQL*Net 2.2 or later and Oracle 7.2 or later. It can be used with any client running Novell libraries, but requires NetWare 4.1 or later on the server.

D.2.3 Optional Configuration Parameters for Clients and Servers

This section describes optional parameters for the NDS Naming Adapter.

D.2.3.1 Optional Configuration Parameter for the Client

There is one optional parameter for the client. It specifies the default name context in which to look for the name to be resolved. The parameter is:

```
NATIVE_NAMES.NDS.NAME_CONTEXT
```

> **Note:** You must add this parameter manually to the SQLNET.ORA file. It cannot be created using Oracle Net8 Assistant.

For example, if the name of the database object is “Payroll.Finance.Oracle” and the SQLNET.ORA parameter is

```
NATIVE_NAMES.NDS.NAME_CONTEXT=Finance.Oracle
```

then the name “Payroll” will be qualified to “.Payroll.Finance.Oracle”. This is an example of a typeless name.

> **Note:** The leading dot designates this as a full NDS name. If you want to override the name context parameter in SQLNET.ORA, then you can specify the full NDS name in the connect string by using a leading dot.

Additionally, names can be specified as typed names. To specify a typed name, enter a parameter and value in SQLNET.ORA like the following:

```
NATIVE_NAMES.NDS.NAME_CONTEXT=OU=Finance.O=Oracle
```

This line will be parsed to produce the typed name `CN=Payroll.OU=Finance.O=Oracle.`

This parameter works similarly to the NET.CFG parameter ”name context”. The name context in SQLNET.ORA will override the entry in NET.CFG. If the SQLNET.ORA parameter is not specified, the NET.CFG parameter will be used. If no name context is specified in either file, it defaults to [root]. See the Novell client documentation for more information on the NET.CFG parameters.

Note: The default name context specified in SQLNET.ORA cannot contain a leading dot. This will result in an NDS error code of -309 (ERR_EXPECTED_IDENTIFIER):

```
The parameter being parsed is not typed.
```

D.2.3.2 Optional Configuration Parameter for the Server Configuration

The network listener running on a NetWare server looks at the following optional parameter in the CONFIG.ORA file, which is located in the ORACLE_HOME/NLM directory:

```
ORACLE_SAP=[OFF|ON]
```

ORACLE_SAP can be set to either ON or OFF. When ORACLE_SAP is ON, the SQL*Net listener advertises its address using SAP (Service Advertising Protocol). When ORACLE_SAP is set to OFF, the SQL*Net listener relies on NDS to deliver information to its clients. If not specified in CONFIG.ORA, ORACLE_SAP defaults to “ON”.

D.2.3.2.1 Reduce Network Traffic by Setting ORACLE_SAP to OFF To reduce network traffic on a network where all clients use NDS, use

```
ORACLE_SAP=OFF
```

D.2.4 Known Limitations

Following are some known limitations when using the NDS Naming Adapter:

- The TNS address stored in the NDS database cannot be more than 2048 characters in length.

- You cannot use more than one listener per database instance. If you do, the last listener to start will overwrite any other TNS address stored in the database object.
- If SID support is enabled on the server, you should not use a NULL SID for any of the database instances. If a NULL SID is used for one of the instances, you cannot connect to it using SQL*Net version 2 or Net8.
- If SID support is not enabled, the last SID specified in the listener's SID_LIST will be the one used. In this case, the SID is transparent to the user and database. The user does not see it and the database ignores it.

Note: SID support is controlled by the following parameter in the CONFIG.ORA file:

```
NW_ENABLE_SID_SUPPORT=[TRUE|FALSE]
```

Glossary

A

access control

See network access control

address

A unique network location used to identify a network object, such as a database server, client, or Names Server. Addresses have a specific format and must be unique. See also well known address.

administrative region

An organizational entity for administering Net8 network components. Each administrative region includes:

- one or more domains
- one or more Oracle Names Servers
- one or more databases, listeners and clients

alias

An alternative name for an existing network object. Once an alias is created, it is resolved to the same name as the initial network object.

API

See Net8 Open.

ASCII character set

Stands for 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. Contrast with EBCDIC character set.

C

cache

Memory that stores recently-accessed data to so that subsequent requests to access the same data can be processed quickly.

CDS

Cell Directory Service

central administration

A Net8 network where network management consists of one administrative region for the entire network. With central administration, all Names Servers know about one another and about all the services in the network. Contrast with delegated administration.

client

A user, software application, or computer that requests the services, data, or processing of another application or computer. In a two-task environment, 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 profile

The properties of a client, often shared by many clients. May include the protocols used and preferred Names Servers.

client-server architecture

Software architecture based on a separation of processing between two CPUs, one acting as the client in the transaction, requesting and receiving services, and the other as the server that provides services in a transaction.

concentration

Consolidating multiple connection requests from clients to establish a single connection to a server in order to conserve server resources.

configuration files

Files that are used to identify and characterize the components of a network. Configuration is largely a process of naming network components and identifying relationships among those components.

CONNECT DATA

A portion of the connect descriptor, introduced by the keyword CONNECT DATA, that specifies the application to which the connection is to be made. The CONNECT DATA section of the connect descriptor includes the system identifier (SID).

connect descriptor

A specially formatted description of the destination for a network connection. Connect descriptors are constructed using a set of keywords and values. They are mapped to service names to provide more convenient reference.

connection

An interaction between two processes on a network. Connections are originated by an initiator (client), who requests a connection with a destination (server).

connection request

A notification sent by an initiator and received by a listener that indicates that the initiator wants to start a connection.

D

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 username that has been given DBA privileges and can perform database administration functions. Usually the two meanings coincide. Many sites have multiple DBAs.

database link

A network object stored in the local database or in the network definition that identifies a remote database, a communication path to that database, and optionally, a username and password. Once defined, the database link is used to access the remote database. Also called DB link.

A public or private database link from one database to another is created on the local database by a DBA or user.

A global database link is created automatically from each database to every other database in a network with Oracle Names. Global database links are stored in the network definition.

See also global database link, private database link, and public database link.

database service name

See *service name.*

decentralized administration

See delegated administration.

dedicated server

A server that requires its own dedicated server process for each user process. Contrast with multi-threaded server.

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.

delegated administration

A Net8 network where network management is delegated to one or more administrative regions below the root administrative region. Also referred to as distributed or decentralized administration. Contrast with central administration.

delegated administrative region

A region hierarchically below the root administrative region. Any region other than the root administrative region.

destination

The client that is the endpoint of a connection. The initiator of the connection requires some data or service of the destination.

distributed administration

See delegated administration.

distributed processing

Division of front-end and back-end processing to different computers. Net8 supports distributed processing by transparently connecting applications to remote databases.

domain

A grouping of network objects, such as databases, that simplifies the naming of network services. Within a domain, all the names must be unique.

domestic domains

The set of domains that are managed within a given administrative region. Domains are only domestic relative to a region; they are never domestic in any absolute sense. Also referred to as local domains.

E

error message

A message from a computer program informing you of a potential problem or condition preventing program or command execution.

F

firewall support

See network access control.

flat naming model

An Oracle Names infrastructure in which there is only one domain. All names must be unique within that domain.

foreign domains

The set of domains not managed within a given administrative region. Domains are only foreign relative 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 Names Server caching performance.

G

global database link

A database link that links each database in a network to all other databases. This enables any user of any database in the network to specify a global object name in a SQL statement or object definition. (The global object name for the Debunk must be the same as the database service name.) See also database link, private database link, and public database link.

global database name

A unique name that identifies a database in a network. It consists of a database name and its network domain name. For example, HR.US.ORACLE.COM is comprised of a database name component HR and a network domain component US.ORACLE.COM.

H

hierarchical naming model

An Oracle Names infrastructure in which names are divided into multiple hierarchically-related domains. You can use the hierarchical naming model with either central or delegated administration.

I

initiator

The client that starts a connection by sending a connection request. The initiator of the connection requires some data or service of the destination.

IPC

Interprocess Communication

K

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.

L

listener process

The server process that listens for and accepts incoming connection requests from client applications. Oracle listener processes start up Oracle database processes to handle subsequent communications with the client.

listener

An agent on a server that listens for connection requests for one or more databases on one or more protocols.

LISTENER.ORA file

A configuration file that describes one or more listeners on a server.

Listener Control Utility (LSNRCTL)

A utility included with Net8 to control various functions, such as to start, stop, and get the status of the listener.

location transparency

A distributed database characteristic that allows 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 for execution 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. See also tracing.

M

multi-threaded server

A 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. Contrast with dedicated server.

multiplexing

Combining multiple sessions for transmission over a single transport connection in order to conserve the operating system's resources.

N

Names Server

A server that uses Oracle Names to store a service's network address along with its simple name so that client applications can request connections with simple names, rather than lengthy addresses.

naming model

The set and structure of domains within which names can be allocated.

In a flat naming model, there is a single domain.

In a hierarchical naming model, the highest level is the root domain, and all other domains are hierarchically related.

Native Naming Adapters

An Oracle product that, when installed with Net8, enables you to connect to Oracle services while continuing to use existing industry-standard name services, such as Network Information Services (NIS), Distributed Computing Environment Cell Directory Service (DCE CDS), Banyan StreetTalk, and Novell's NetWare Directory Services (NDS).

Net8

Oracle's remote data access software that enables both client-server and server-server communications across any network. Net8 supports distributed processing and distributed database capability. Net8 runs over and interconnects many communications protocols. Net8 is backward compatible with SQL*Net version 2.

Net8 Open

The application program interface (API) to Net8 that enables programmers to develop both database and non-database applications that make use of the Net8 network already deployed in their environment. Net8 Open provides applications a single common interface to all industry standard network protocols.

netBIOS

Stands for Network Basic Input/Output System, a process-to-process communications protocol designed to provide virtual links between machines on a network.

netWare

A network operating system produced by Novell.

network

A group of two or more computers linked together through hardware and software to allow the sharing of data and/or peripherals.

network access control

A feature of Oracle Connection Manager that sets rules for denying or allowing certain clients to access designated servers. Also known as firewall support.

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 listener

See listener.

network object

Any service that can be directly addressed on a network; for example, a listener or a Names Server.

network objects

The types of network names stored in an Oracle Names Server. These include database service names, global database links, and aliases.

network protocol

A set of rules that defines how data is transported across networks. There are several industry standard transport protocols, such as TCP/IP and SPX.

network service

In an Oracle application network, a service performs tasks for its service consumers; for example, a Names Server provides name resolution services for clients.

NI

Network Interface

NL

Network Library

NN

Network Naming (Oracle Names)

node

A computer or terminal that is part of a network.

NPI

Network Program Interface

NR

Network Routing

NS

Network Session

NT

Network Transport

O

OPI

Open Program Interface

ORACLE_HOME

An alternate name for the top directory in the Oracle directory hierarchy on some directory-based operating systems.

Oracle Names

A directory service made up of a system of Oracle Names Servers (called Names Servers) that provide name-to-address resolution for each Net8 service on the network.

Oracle Names infrastructure

A set of initial decisions and policies that govern how names are allocated, and how Oracle Names operates. The infrastructure defines how users and administrators interact with the Oracle Names system.

Oracle Names Server

See Names Server.

Oracle Protocol Adapters

A set of products which map Net8 functionality to industry-standard protocols used in client-server connections. These adapters are installed separately from Net8.

P

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, etc. Packet information can be useful in troubleshooting.

parameter

Information passed to a program, command, or function, such as a file specification, a keyword, or a constant value.

password

A string (word or phrase) used for data security and known only to its owner. Passwords are entered in conjunction with an operating system login ID, Oracle username, or account name, in order to connect to an operating system or software application (such as the Oracle database). Whereas the username or ID is public, the secret password ensures that only the owner of the username can use that name, or access that data.

preferred Names Server

The Names Server(s) preferred by a client for names resolution; usually the Names Server that is physically closest to the client, or available over the least expensive network link.

prestarted or prespawned dedicated server process

Prespawned dedicated server processes are prestarted by the Net8 listener before any incoming connection request. They improve the time it takes to establish a connection on servers where the multi-threaded server is not used or not supported on a given machine. They also use allocated memory and system resources better by recycling server processes for use by other connections with shutting down and recreating a server. The ability to prespawn dedicated servers requires Oracle7 Server release 7.1 and SQL*Net release 2.1 or later.

private database link

A database link created by one user for his or her exclusive use. See also public database link, database link, and global database link.

protocol

See network protocol.

protocol adapter

See Oracle Protocol Adapters.

public database link

A database link created by a DBA on a local database which is accessible to all users on that database. See also public database link, database link, and global database link.

R

RDBMS

Relational Database Management System

root administrative region

The highest level administrative region in a distributed installation. The root administrative region contains the root domain.

root domain

The highest level domain in a hierarchical naming model.

RPC

Remote Procedure Call

S

service name

A name for a connect descriptor that is easy to use and remember. End users need only know the appropriate service name to make a connection. Each connect descriptor is assigned a service name in the network definition.

service replication

A process that fully replicates a directory system on the network. New services need to register with only one Names Server. The service replication process automatically distributes the new registration to all other active Names Servers on the network.

SID

See system identifier.

SPX

Sequenced Packet Exchange, a network protocol known for high performance and acceptance among many major network management systems, in particular, Novell Advanced NetWare.

SQL*Net

Net8's precursor. An Oracle product that works with the Oracle Server and enables two or more computers that run the Oracle RDBMS or Oracle tools such as SQL*Forms to exchange data through a network. SQL*Net supports distributed processing and distributed database capability. SQL*Net runs over and interconnects many communications protocols.

system identifier (SID)

A unique name for a database instance. To switch between databases, users must specify the desired SID. The SID is included in the CONNECT DATA parts of the connect descriptors in a TNSNAMES.ORA file, and in the definition of the network listener in the LISTENER.ORA file. If you choose the default configuration, the SID defaults to "*".

system or topology data

Data used by the Names Server to control regular functioning or communicate with other Names Servers. Includes Interchanges, root region's Names Servers, and any delegated regions' Names Servers.

T

TNS

See Transparent Network Substrate.

TNSNAMES.ORA file

A file that contains connect descriptors mapped to service names. The file may be maintained centrally or locally, for use by all or individual clients.

tracing

A facility that writes detailed information about an operation to an output file. The trace facility produces a detailed sequence of statements that describe the events of an operation as they are executed. Administrators use the trace facility for diagnosing an abnormal condition; it is not normally turned on. See also logging.

Transparent Network Substrate (TNS)

A foundation technology, built into Net8, Oracle Connection Manager, and Oracle Names, that works with any standard network transport protocol.

U

username

The name by which a user is known to the Oracle Server and to other users. Every username is associated with a password, and both must be entered to connect to an Oracle database.

UPI

User Program Interface

W

well-known address

Addresses for one or more Names Servers are hardcoded into both the Names Server and its clients. Names Servers then become available at these well known addresses, so that clients do not need to be told, by way of configuration files, where to find the server.

Index

Symbols

A

B

C

D

E

F

G

H

I

J

K

L

M

N

O

P

Q

R

S

T

U

V

W

X

Y

Z