AT&T 585-350-913
Issue 1
Comcode 407298637

SQL*Net® TCP/IP User's Guide

VERSION 1.2

[Revised 1994]

ORACLE®

The Relational Database Management System

SQL*Net TCP/IP User's Guide
Version 1.2

Part No. A19164-1 [Revised 1994]

Contributing Authors: J'aime Amadeo, Jocelyn Audette, Andrew Davis, Jason Durbin, Lynn Hollinger, Patrick Holtz, Jane Sokolow, Thomas Sandy

PREFACE

Purpose

The *SQL*Net TCP/IP User's Guide* explains how to use SQL*Net on a TCP/IP network to exchange information among ORACLE databases and applications.

This user's guide contains the following information:

- an overview of TCP/IP and SQL*Net
- how to configure SQL*Net for your system
- how to use SQL*Net to connect with remote databases

Audience

The information in this manual is intended primarily for system and database administrators (DBAs). However, the *SQL*Net TCP/IP User's Guide is* for anyone who wants to communicate over a TCP/IP network with Oracle products.

How This Manual Is Organized

The *SQL*Net TCP/IP User's Guide* is divided into two parts and is followed by an error code appendix, a glossary, and an index.

Part I: Understanding SQL*Net TCP/IP

This section contains two chapters:

Chapter 1 introduces SQL*Net and the TCP/IP protocol, explains how ORACLE uses the communications protocol, and provides general information about configuring SQL*Net on a TCP/IP network.

Chapter 2 explains constructing a SQL*Net TCP/IP Version 1 connect string, and provides some examples of ORACLE commands that use SQL*Net to connect with remote databases.

Part II: Configuring SQL* Net for your System

This section contains five chapters. Each provides information on how to setup and configure SQL*Net for an operating system that can use SQL*Net TCP/IP Version 1 to communicate across a network. The UNIX, OpenVMS, and DG AOS/VS chapters contain some additional information about monitoring network activity. The UNIX chapter also describes using SQL*Net TCP/IP TLI over UNIX System V networks implementing the TCP/IP protocol.

The appendix contains error codes for SQL*Net TCP/IP and TCP/IP TLI. The glossary contains definitions of SQL*Net terms.

Related Publications

Along with this guide, you may want to refer to other documents published by Oracle Corporation

- *ORACLE for DEC VAX OpenVMS Installation*
  Part No. 1001-V5.1, 1001-V6.0
- *ORACLE for Alpha AXP OpenVMS Installation Guide*
  Part No.
- *ORACLE for Unix V Installation and* User's *Guide*
  Part No. 5083-V5.1.22
- *Ungermann Bass TCP-PC User's Guide*
  Available from Ungermann Bass
- *SQL*Net for Data General AOS/VS Installation and User's Guide*
  Part No. 6918-10-0992
- *Installing and Administering Data General for TCP/IP*
  Available from Data General
- *ORACLE for Data General AOS/VS Installation and User's Guide*
  Part No. 1005-V5.1
- *SQL*Net for MVS Installation and User's Guide*
  Part No. 5136-17-0692
- *SQL*Net TCP/IP Manager for MVS System Release Bulletin*
  Part No. 5145-V1.1.1
- *ORACLE7 for VM SQL*Net Configuration and User's Guide*
  part No. A10305-1
- *Oracle for Macintosh: Networking*
  Part No. 51180-V1.1
- *Oracle for Macintosh: Networking Guide*
  Part No. 51180
- *Setting Up SQL*Net TCP/IP for Macintosh Servers*
  Part No. 90752-15-0942
- *Setting Up SQL*Net TCP/IP for Macintosh Clients*
  Part No. A13700

- *ORACLE TCP/IP Adapter for Windows NT Installation and User's Guide* Part No. A16999-1
- *Setting Up SQL*Net TCP/IP for Windows NT Clients* Part No. A16999-1
- *Setting Up SQL*Net TCP/IP for Windows* Part No. A10458-3
- *Setting Up SQL*Net Named Pipes for Windows* (Oracle Part No. A13525-3)
- *Setting Up SQL*Net TCP/IP for Netware* Part No. A11624-1
- *Setting Up SQL*Net TCP/IP for DOS* (Oracle Part No. A11046-1)
- *SQL*Net TCP/IP for MS-DOS Getting Started* Part No. 1022-V5.1 or 1022-V6.0)
- *ORACLE RDBMS for MS-DOS Getting Started* Part No. 5053-V5.lB
- *ORACLE RDBMS for MS-DOS User's Guide* Part No. 5048-V5.lB
- *Setting Up SQL*Net TCP/IP for OS/2 Clients* (Oracle Part No. A16079-1)
- *SQL*Net TCP/IP for OS/2 Release Notes* (Oracle Part No. 90038-11-1192)

Notational Conventions

This manual describes using SQL*Net TCP/IP with several types of operating systems. The same notational conventions can be applied to all operating systems except UNIX. Because UNIX is case sensitive, all filenames and commands mentioned in the text appear in boldface. For the other chapters, the following conventions are used:

`Monospace Text`	Enter text or command exactly as shown.
`Italics`	Italics in a mono spaced font represents a parameter. Substitute an appropriate value.
[ ]	Brackets indicate that the enclosed item is optional. Do not enter the brackets.
{ }	Curly braces indicate that one of the enclosed items must be entered. If the curly braces are surrounded by brackets, then all of the enclosed items are optional. Do not enter the curly braces.
|	A vertical bar is used to separate options within brackets and braces. You must enter one of the options separated by the vertical bar. Do not enter the vertical bars.

Punctuation	Punctuation other than brackets, braces, and vertical bars must be entered as shown.
UPPERCASE	Uppercase words within the text indicate command names and filenames.

Your Comments Are Welcome

We value and appreciate your comments as an ORACLE user. As we write, revise, and evaluate our work, your opinions are the most important input we receive. At the back of this manual is a Reader's Comment Form; we encourage you to use this form to tell us both what you like and dislike about this (or other) ORACLE manuals. If the form is missing, or you would like to contact us, please use the following address and phone number:

SQL*Net Product Manager
Oracle Corporation
500 Oracle Parkway
Box 659410
Redwood Shores, California 94065

CONTENTS

PART

I

UNDERSTANDING SQL*NET TCP/IP VERSION 1

This part introduces ORACLE and SQL*Net TCP/IP Version 1, and explains how you can use them to connect to remote databases and gain access to the information stored in those databases.

CHAPTER

1

OVERVIEW OF SQL*NET AND THE TCP/IP PROTOCOL

This chapter provides an overview of SQL*Net and the TCP/IP protocol by briefly discussing the following topics:

- SQL*Net
- TCP/IP protocol
- SQL*Net TCP/IP Version 1 and ORACLE

For information about initiating and maintaining connections with ORACLE databases located on remote hosts, refer to Chapter 2, “Using SQL*Net TCP/IP.” To obtain specific information about configuring and using SQL*Net TCP/IP with your operating system, refer to the appropriate chapter in Part II.

What is SQL*Net?

SQL*Net is the Oracle communications component that allows you to share information stored in different ORACLE databases. With SQL*Net, you can run an ORACLE tool or another application on one machine and be able to find, manipulate, and store data in an ORACLE database located on another machine.

SQL*Net also allows applications to connect to multiple ORACLE databases across a network, using a variety of *communication protocols* and *application program interfaces* (APIs) to establish a *distributed processing* and *distributed database* environment.

A communication protocol is a set of implemented standards or rules that govern data transmission across a network.

An API is a set of subroutines that provide an interface for application processes to the network environment. TCP/IP is one of several network protocols supported by SQL*Net.

Distributed Processing

Dividing processing between a front end machine running an application and a back end machine that is utilized by the application is known as distributed processing. SQL*Net enables an ORACLE tool or some other application to connect with a remote machine containing an ORACLE RDBMS.

One advantage of distributed processing is the optimization of your computing resources. For example, running an ORACLE RDBMS on a large computer with enormous and inexpensive memory and using a PC for an application that needs access to the database maximizes the capabilities of both machines.

Distributed Database

Several databases linked through a network that appear to a user as a single logical database is known as a distributed database. SQL*Net provides you with access to distributed databases. An ORACLE tool running on a client machine or an ORACLE RDBMS running on a host machine can share and obtain information retrieved from other remote ORACLE databases. Regardless of the number of database information sources, the user is only aware of one logical database.

The major advantage of a distributed database is that users and applications do not have to know where the needed data tables are actually located. If a client requests access to a database that is not local, the SQL*Net driver takes control and makes the proper connection to satisfy the request. Users no longer need to “switch” between computers or transfer files across the network to work with the remote database.

Clients and Servers

The role of SQL*Net in the client/server model is to connect the client application with the database server. Each role can be defined as follows:

Client	ORACLE tool or application that connects to the database server(s) on host machines.
Server	A multiuser ORACLE RDBMS running on the host machine that serves applications on remote computers.

Machines can function as either servers or clients with two exceptions: an IBM PC running MS-DOS and a Macintosh running its hierarchical filing system. The single-user architectures of these operating systems prevent them from being anything but clients.

An ORACLE RDBMS can make a client connection request through SQL*Net to a remote host running ORACLE and obtain access to a remote database. Whenever two machines communicate through SQL*Net, one machine functions as a client and the other machine functions as the host running the database server.

A communication protocol is necessary for one node to transfer information to another. The same protocol must be available on both the client and server machines.

SQL*Net Use

SQL*Net is designed to work in either of two configurations:

- without a local ORACLE RDBMS
- with a local ORACLE RDBMS

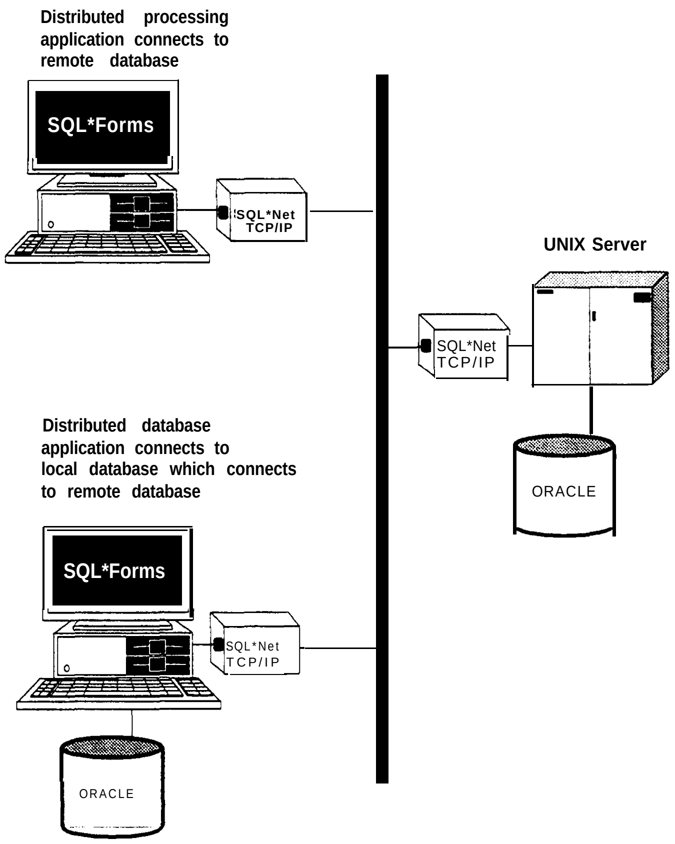
SQL*Net without a local ORACLE RDBMS can be used for distributed processing and databases. Client applications make requests and SQL*Net directs them to the information whether it involves a single step to one host or several steps to multiple hosts and databases.

SQL*Net with a local ORACLE RDBMS enables an application to use either the local database or make a connection to a remote host. The client request could involve a distributed database if both the local and a remote database are used.

With SQL*Net and ORACLE, users can communicate and share database information stored in different locations, on different machines, and with different operating systems.

Figure 1-1 shows the difference between distributed processing and a distributed database.

Figure 1-1 Database Configurations


Distributed processing application connects to remote database
SQL*Forms
SQL*Net TCP/IP
UNIX Server
SQL*Net TCP/IP
ORACLE
Distributed database application connects to local database which connects to remote database
SQL*Forms
SQL*Net TCP/IP
ORACLE

What is TCP/IP?

TCP/IP, like other communication protocols, is a set of rules used to exchange information between computers. TCP/IP is a family of related protocols that derives its name from two main components: the Transmission Control Protocol (TCP) and the Internet Protocol (IP). The IP component dispatches information around the network, and the TCP component assures information accuracy.

TCP/IP was developed at the University of California at Berkeley for ARPANET, the Advanced Projects Agency Network of the U.S. Department of Defense. As a publicly developed, non-proprietary protocol, the TCP/IP standard has been embraced by a number of hardware and software vendors and is commonly used for linking equipment supplied by different vendors.

TCP/IP is a process-to-process (sometimes called a task-to-task or host-to-host) protocol. When one system requests a connection with another, the receiving machine's listener process acts upon the request by starling a server for the specific type of request. The server (host) process is independent of the communication task and remains after the communication is complete.

Benefits

The TCP/IP protocol offers many advantages:

- It offers high speed data transmission over an Ethernet network.
- The specifications are complete and available.
- It contains provisions for most network events, including data errors, security, and line failure.
- The call level interface is consistent across hardware systems, making it one of the most portable protocols available.
- It is currently used in many heterogeneous computing environments to enable diverse CPUs to communicate.
- It has migration paths to meet the networking standards of the International Standards Organization (ISO).

Common Uses and Applications

TCP/IP is a flexible, robust protocol with a wide range of applications. For example:

- TCP/IP has been adopted by many independent vendors and implemented on many different computer systems (from PCs to supercomputers). Therefore, organizations using TCP/IP networks can connect dissimilar machines with various operating systems.
- TCP/IP is fast and efficient on a local-area network and is very effective for departments and organizations that require close communication, including:
  - business offices
  - CAD/CAM installations
  - hospitals
  - laboratories
  - universities and colleges

- TCP/IP was developed for UNIX systems and is widely used in UNIX installations such as universities and data processing departments.
- TCP/IP was also developed for Arpanet and is widely used for defense applications. TCP/IP is consistent with the Department of Defense standards.

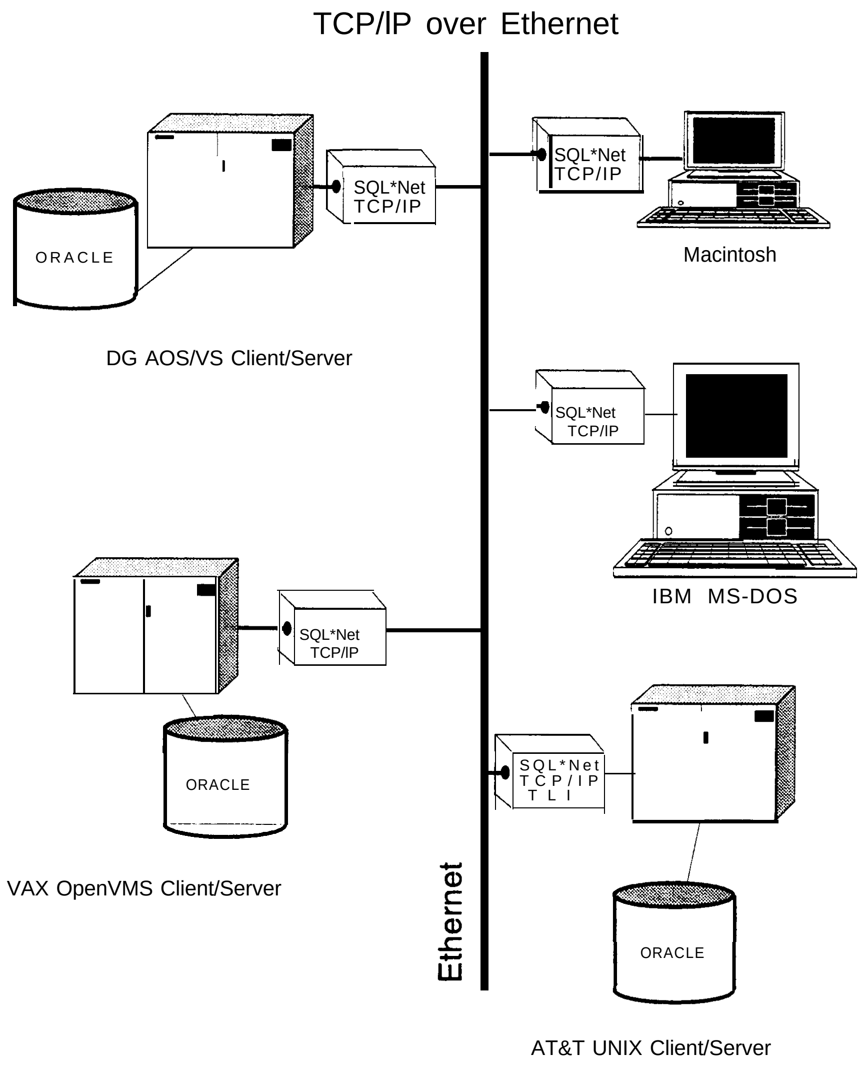
SQL*Net TCP/IP Version 1 Support

SQL*Net and the TCP/IP protocol provide distributed processing and database capabilities for the following configurations and operating systems:

- MS-DOS, Windows, and OS/2 systems
- BSD UNIX systems
- UNIX Systems V supporting TCP/IP TLI
- VAX and Alpha Open VMS systems
- Xenix LANserver systems
- DG AOS/VS systems
- IBM systems running VM
- IBM systems running MVS
- Macintosh systems running HFS

These configurations run on an Ethernet, among other media, with an appropriate controller installed on each node in the network. For details about the software and hardware requirements, consult the ORACLE documentation that covers your hardware and software and the documentation provided by their manufacturer. Figure 1-2 shows a TCP/IP Ethernet network.

Figure 1-2
TCP/IP Ethernet Network



CHAPTER

2

USING SQL*NET TCP/IP VERSION 1

After SQL*Net TCP/IP Version 1 has been installed and configured for your system, you can use it to connect to any ORACLE applications or databases on the network. This chapter contains:

- explanations of SQL*Net TCP/IP Version 1 command line parameters
- several SQL*Net TCP/IP Version 1 command line examples

SQL* Net Login Parameters

You can use SQL*Net to connect to a remote database when you run any ORACLE application that accepts a username and password in the command line. For example, SQL*Plus, SQL*Forms, and SQL*Calc all accept a username and password on the command line. The general syntax for running an application program from the command line is:

```
application  username/password@connect  string
```

where

application	Is the command that invokes the Oracle application program.
username	Specifies the username to be used to connect to the remote database.
password	Specifies the password associated with the username.
connect string	Specifies the network prefix, the target host for the remote database, and the system ID (SID) of the remote database that is the target of your connection request.

Note: If you are using an ORACLE application that requires its own parameters, place those parameters after the SQL*Net-specific parameters. For example:

```
SQLFORMS SCOTT/TIGER@D: HR -C BIOS
```

Connect String Syntax

The syntax for the connect string for the TCP/IP protocol is:

```
net prefix: hostname: system ID, buffer size
```

You can omit any optional parameter, but you must include comma separators for any parameter you skip.

Note: Some operating systems require different parameters to be included when using a connect string. Special parameters to use in connect strings are described in chapters 3 through 8 explaining SQL*Net TCP/IP configuration for the supported operating systems.

The sign @ is the required delimiter that begins the connect string.

net prefix	For TCP/IP systems, this prefix is always **T.** If you are using SQL*Net TCP/IP TLI, use the **TT** net prefix. If you want to connect to a remote database, this parameter must be specified on the command line, or in a system default or alias.

hostname	The name or alias of the remote host that is the target of your connection request. This name must be defined in the HOSTS file on your client system. The default node is identified by the alias “SQLNET”.
system ID	For ORACLE V5.0, a single character that represents the system ID (SID) of the database you want to use on the remote server. If you are using ORACLE RDBMS V6.0, the SID can be more than one character. This character must be defined in the database listing file on the remote host. This parameter is case sensitive.
buffer size	The size (in bytes) of the buffer used by SQL*Net to pass data between ORACLE and TCP/IP. This value defines the maximum size of a SQL*Net packet. You can set the value between 4 and 4096; larger values result in faster transfer of large data arrays but consume more memory. The default is 4096 except on MS-DOS where the default is 1024. On a system with limited memory, you may want to reduce this value to conserve space.

Aliases and Defaults

SQL*Net TCP/IP allows you define aliases and defaults to replace certain parameters in the connect string. For example, by assigning the alias “SQLNET’ to a system, you can omit a hostname in the connect string. SQL*Net automatically connects with the host identified by SQLNET.

Note: If SQL*Net Version 2 is installed on your system, aliases are done using the Version 2 naming mechanism. See your SQL*Net Version 2 documentation for more information.

You may be able to define some or all of the TCP/IP parameters as system defaults. On an MS-DOS system, you can use the LOCAL variable in the CONFIG.ORA file to define a default protocol, host, and database; this allows you to omit connect string parameters when you want to connect to the default system.

Similarly, you may be able to use an alias in place of some or all of the TCP/IP parameters on the command line. UNIX systems permit you to use a file called **sqlnet** to define aliases or abbreviations for parameter combinations.

For more information about setting up system defaults and aliases, see the section on your SQL*Net system in Part II.

Connecting with SQL*Net TCP/IP

This section provides examples of how to use SQL*Net parameters on the command line to access a remote database.

Example 1 This example shows how to use TCP/IP to connect to the username SCOTT/TIGER on the remote host "BOSTON." The database in this example has an ORACLE SID of "A'.

```
SQLPLUS SCOTT /TIGER@T : BOSTON : A
```

For VAX/VMS clients, you must enclose the SID in quotes:

```
SQLPLUS SCOTT /TIGER@T : BOSTON: "A"
```

Example 2 This example shows how the hostname and/or the SID can be omitted from the command line. (Refer to the chapter explaining the operating system you are using to see if this option applies to you.) The command line below specifies a SQL*Net TCP/IP connection to SCOTT/TIGER on the default host "SQLNET". (You must have SQLNET setup as an alias for the default host in the HOSTS file.) In this example, the database identified on the host with the null ORACLE SID is used as the default.

```
SQLPLUS SCOTT/TIGER@T :
```

Example 3 This example shows how to specify a buffer size for passing data between ORACLE and TCP/IP. This command will connect to "BOSTON" using a buffer size of 2000 bytes:

```
SQLPLUS SCOTT /TIGER@T : BOSTON : A, 2000
```

Example 4 This example shows how you can omit a username and password from the command line if an **ops$ login** has been assigned to the username and password you logged in under. (Refer to the chapter explaining your operating system and the *ORACLE RDBMS Database Administrator's Guide* to see if this option applies to you.)

NOTE: If you are using ORACLE7, the ops$login mechanism has been replaced with the REMOTE_OS_AUTHENT parameter. See the ORACLE7 server manual for more information.

If the username "scott" and the password "tiger" are assigned an ops$ login on the host machine, any clients logged in to the client operating system as "scott/tiger" can connect to a database with the following command:

```
SQLPLUS /
```

The following command will connect to “BOSTON” using a buffer size of 3000 bytes:

```
SQLPLUS /@T: BOSTON:A, 3000
```

Connecting to Another System

The SQL CONNECT command is normally used to connect to another ORACLE username on the current database. You can also use CONNECT with SQL*Net parameters to connect to a different database. The syntax is:

```
SQL>  CONNECT username/password@ connect string;
```

This example shows how to connect to another remote database from within SQL*Plus:

```
SQL>  CONNECT SCOTT /TIGER@T : DALLAS : B
```

The CONNECT command commits all of your pending work in the current database and log offs the current username. With SQL*Plus, you can never log into more than one database at once.

Copying Data Between Systems

At times, you may want to move data from one node to another. For example, you may want to download sales data from a particular region to your PC for further analysis.

The SQL*Plus COPY command lets you copy data directly from one database to another. The syntax is:

```
COPY FROM username/password@ connect string
TO username/password@ connect string
{APPEND | CREATE | INSERT | REPLACE}
tablename [ (columnist) ]
USING subquery;
```

where the *connect string* contains a valid set of SQL*Net parameters.

Note: When performing a COPY command, SQL*Plus makes a connection to both the source and destination databases if they are different than the current logged in database. This can result in up to three connections at once being open, which may not be supported on some protocols and platforms (for example, SQL*Net Asynchronous).

Creating Database Links

The SQL command CREATE DATABASE LINK grants you access to multiple remote databases while you are still connected to a local database. All database links are saved in the data dictionary of the database where the link was created. Some data dictionary views can be queried to see names of existing database links and synonyms.

The following data dictionary views can be queried to identify database links:

USER_DBLINKS	Public and private links to external databases
DBA_DBLINKS	All links to external databases (DBA use only)

To define a database link to a remote database, use the following syntax:

```
CREATE [PUBLIC] DATABASE LINK  linkname
CONNECT TO username IDENTIFIED BY password
USING  'connect string';
```

Only DBAs can use the PUBLIC option. This option makes the database link access available to any user with CONNECT privileges on the local database.

If the local RDBMS is V6.0 or higher, the CONNECT TO clause is optional. The absence of the CONNECT TO clause indicates that your current ORACLE username and password should be used to make the connection with the remote database.

The following example shows how to create a private database link (accessible only to the creator) named SALESDATA that refers to the user SCOTT with the password TIGER on the specified database:

```
SQL> CREATE DATABASE LINK SALESDATA
CONNECT TO SCOTT IDENTIFIED BY TIGER
USING 'T: BOSTON:A' ;
```

After the link has been established, you can connect to a remote database while you are still logged in locally. You can use the *database link in* SQL commands to query information in the remote database. As many as four different links can be used in one SQL statement. However, the DBA can adjust this limit.

For example, you can query a table named EMP that is owned by SCOTT on 'T:BOSTON:A' as in the following statement:

```
SQL>  SELECT * FROM EMP@SALESDATA;
```

For more information about using SQL*Plus, refer to the *SQL*Plus Reference Guide.*

Creating Synonyms

Database links provide access to multiple remote databases from a local logon, but their use increases the complexity of the network. Synonyms are aliases for the database links you created and a table located on the host containing the database. Using synonyms eliminates the need to specify both a table name and the database link on the command line. All synonyms are saved in the data dictionary in the database where you created them. The following data dictionary views can be queried to identify synonyms:

PRIVATESYN	Shows only private synonyms created by the user
PUBLICSYN	Shows only public synonyms
SYNONYMS	Shows both public and private synonyms

To create a synonym, type:

```
SQL> CREATE SYNONYM name  FOR tablename@databaselink
```

For example, to create the synonym BEMP for the EMP table on the remote database server BOSTON that is identified by the database link SALESDATA, type:

```
SQL> CREATE SYNONYM BEMP FOR EMP@SALESDATA;
```

If you created this synonym, you could request a connection to the EMP table by typing:

```
SQL> SELECT * FROM BEMP;
```

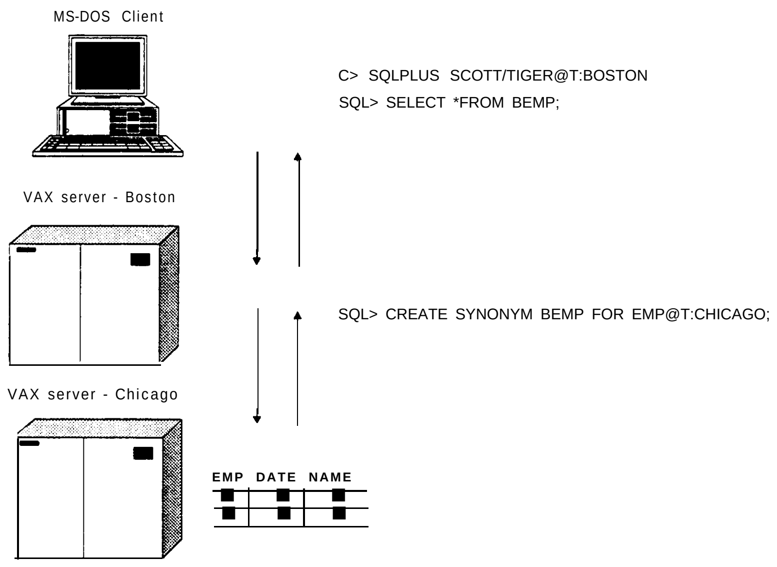
Synonym Chaining

Sometimes it maybe necessary to move a database table from one host to another remote host. For example, you might move the EMP table from the BOSTON host to a host named CHICAGO because the department using the table moved from Boston to Chicago. Synonym “chaining” allows a client to access the EMP table even though it is unaware of the new table location. The client is then working in a “location transparency” environment.

In Figure 2-1, the EMP table now resides on the most remote database server, identified by the node CHICAGO. You can define a synonym on the node BOSTON that refers to the EMP table at CHICAGO with this command:

```
SQL> CREATE SYNONYM BEMP FOR EMP@T : CHICAGO
```

Figure 2-1
Chaining Synonyms



The client can then connect to the EMP table on the CHICAGO machine using the BEMP synonym and issue SQLPLUS commands without needing to know the exact location of the table.

Both remote queries and remote updates are possible.

For example, a client can issue this command:

```
SELECT * FROM BEMP
```

or this command:

```
UPDATE BEMP set x = y
```

PART

II

CONFIGURING SQL*NET FOR YOUR SYSTEM

This part presents instructions on how to configure SQL*Net TCP/IP Version 1 on your system. These tasks are normally completed by the database administrator (DBA). However, if you are using SQL*Net on a personal computer, you may be acting as the DBA and should complete the procedures in the appropriate chapter in this part.

This part is organized into separate chapters for each type of operating system supported by SQL*Net TCP/IP:

- UNIX (Chapter 3)
- VAX/VMS (Chapter 4)
- Data General AOS/VS (Chapter 5)
- IBM VM (Chapter 6)
- IBM MVS (Chapter 7)

Turn directly to the chapter for your SQL*Net TCP/IP system.

If you are using SQL*Net TCP/IP with a platform not listed above, see the installation manual for your platform. For example, if you are using SQL*Net TCP/IP with a Macintosh, refer to the following manuals:

- *Oracle for Macintosh: Networking Guide* (Oracle Part No. 51180)
- *Setting Up SQL*Net TCP/IP for Macintosh Servers* (Oracle Part No. 90752-15-0942)
- *Setting Up SQL*Net TCP/IP for Macintosh Clients* (Oracle Part No. A13700)

If you are using SQL*Net TCP/IP on a PC, refer to the following documents, depending on your environment:

- *ORACLE TCP/IP Adapter for Windows NT Installation and User's Guide (Oracle Part No. A16999-1)*
- *Setting Up SQL*Net TCP/IP for Windows NT Clients* (Oracle Part No. A16999-1)
- *Setting Up SQL*Net TCP/IP for Windows* (Oracle Part No. A10458-3)
- *Setting Up SQL*Net Named Pipes for Windows* (Oracle Part No. A13525-3)
- Setting *Up SQL*Net TCP/IP for OS/2 Clients* (Oracle Part No. A16079-1)
- *SQL*Net TCP/IP for OS/2 Release Notes* (Oracle Part No. 90038-11-1192)
- *Setting Up SQL*Net TCP/IP for Netware* (Oracle Part No. A11624-1)
- *Setting Up SQL*Net TCP/IP for DOS* (Oracle Part No. A11046-1)

CHAPTER

3 SQL*NET TCP/IP FOR UNIX

This chapter explains how to configure SQL*Net TCP/IP for the various UNIX operating systems. It is assumed that SQL*Net TCP/IP has been installed on your system. Please refer to the *ORACLE Installation and User's Guide* written for your system to see details about installing SQL*Net TCP/IP.

This chapter also explains how to use the SQL*Net TCP/IP TLI product. SQL*Net TCP/IP TLI interoperates completely with SQL*Net TCP/IP (which is sockets-based.)

Note: Because UNIX is case-sensitive, commands and filenames are printed in **boldface** type. Therefore, anytime you enter syntax found in this chapter's examples, you should enter the syntax exactly as it appears in the examples.

Configuration Tasks for UNIX

The machines that are involved in your network must be configured to be able to communicate as a client or a server. You need to complete the following tasks to configure the clients and server machines on the network.

- test the TCP/IP program
- identify the TCP/IP port number
- identify the available hosts
- set aliases and defaults
- identify the available databases
- start the server process

Note: A DBA can securely manipulate remote databases if four conditions are met

- The user ID of the DBA must be the same on all machines administered by the DBA.
- The DBA user ID is part of the UNIX system that has DBA privileges for the database instances that are the responsibility of the DBA.
- To separate database and machine administrations, **root** and **daemon** should never be a member of the DBA group.
- **orasrv** (the SQL*Net TCP/IP server process) must be made **"setuid root"** as follows:

```
chown root orasrv
chmod 4555 orasrv
```

Note: Automatic logins are supported over SQL*Net TCP/IP, but not over SQL*Net TCP/IP TLI. Also, note that out-of-band breaks do not function properly when using TCP/TLI.

$ORACLE_HOME is a shell variable representing the ORACLE home directory on your system. This variable is given a value in each user's startup file **(.login** for C Shell, **.profile** for Bourne Shell or ksh). In the examples given in this chapter, unless otherwise noted, you can substitute the ORACLE home directory for **$ORACLE_HOME.**

Testing the TCP/IP System

To make sure that your communications hardware and software are working correctly, connect to your local machine by using the TCP/IP **telnet** procedure. To use **telnet,** enter:

```
$ telnet hostname
```

where *hostname* is the host system to which you are trying to connect.

For example, if you entered "boston” as a hostname, you should receive a prompt for logging onto **boston.**

Identifying the Port Number and Service (Clients and Servers)

The SQL*Net TCP/IP protocol uses a port number to indicate where clients and servers connect. A server process named **orasrv** handles connection requests.

Oracle Corporation recommends using the port number 1525. Whatever number you choose, all clients and servers on the network must use the same number. You only have to identify the port number once, when you perform the initial system configuration tasks. To make the port number known to the TCP/IP driver, it must be identified in the **/etc/services** file. (If the **/etc/services** file does not already exist, use a text editor to create it on *both* the client and server machines.)

If your **/etc/services** file does not include the following information, use a text editor to add the following line to identify 1525 as the port number:

```
orasrv  1525/tcp
```

where

orasrv	Is the name of the service used by SQL*Net.
1525/tcp	Is the port number and the protocol used.

Note: In a production ORACLE environment, the **/etc/services** file may be write protected. In that case, coordinate any modifications to the file with your system administrator.

Identifying the Available Hosts (Clients Only)

If you are configuring a machine as a client, you must identify the hosts to which you want to connect. Complete this task during the initial system configuration and each time you want to identify a new host.

The **/etc/hosts** file is often used to identify the available hosts. The entries in this file map the hostname for each available host on the network to its Internet address.

The mapping for each host listed in the **/etc/hosts** file is specified on a single line in the following format:

```
internet_address  hostname  [ alias alias . . . ]
```

Host names can also be listed in NIS or DNS-contact your system administrator if names need to be added using these mechanisms.

where

internet_address	Is the Internet address of the host computer (a four-byte value specified in decimal, octal, or hexadecimal).
hostname	Is the name of the host associated with the *internet_address.*
alias	Is an alternate name for the host. Brackets indicate that this parameter is optional. You can have as many aliases as you want for any one host. An alias does not have to be unique across the network.

For example, a host named “bostonsales” might be referenced like this in the **/etc/hosts** file:

```
89.0.1.20    bostonsales   boston
```

where

89.0.1.20	Is the Internet address of the remote host.
bostonsales	Isa name for the remote host.
boston	Is an alias setup for the bostonsales host.

If your **/etc/hosts** file is incomplete, edit the file to include additional host names and Internet addresses.

If a second alias, “northeast,” was assigned to the “bostonsales” host, the **/etc/hosts** file would look like this:

```
89.0.1.20 bostonsales boston
89.0.1.20 bostonsales northeast
```

Setting Aliases and Defaults (Clients Only)

To use the TCP/IP protocol more efficiently on client systems, Oracle Corporation recommends:

- creating aliases for commonly used connect strings in the **/etc/sqlnet** file
- setting the default alias **sqlnet** in the **/etc/hosts** file for a host that will be the default for all SQL*Net connections

Note: If SQL*Net Version 2 is installed on your system, naming mechanisms are different and different rules about aliases apply. See your SQL*Net Version 2 documentation for more information.

Creating Aliases

This section explains how to use the **/etc/sqlnet** file to set configuration defaults or aliases.

Aliases are simply shorter, more convenient names for connect strings. You can define connect string aliases either in the public database link file **/etc/sqlnet,** or in your private database link file **$HOME/.sqlnet.** The public database link file is accessible by all users on your machine. The private database link file is in your home directory for private use.

The two-task interface scans your **$HOME/.sqlnet** file before it scans the public **/etc/sqlnet** file and returns the first matching instance. Thus, if you define an alias in your **$HOMEi.sqlnet** file that has the same name as another alias in **/etc/sqlnet,** the two-task interface will always use your private alias definition.

You can include an alias in the **/etc/sqlnet** file for a commonly used connect string. The syntax for including such an alias is:

```
alias    net prefix: hostname:system id
```

where

alias	Isa single word substitution for the full connect string.
net prefix	Is the network prefix for TCP/IP (always T) or TCP/IP TLI (always TT).
hostname	Is the default host name.
system id	Is the system ID or SID for the database. The SID is case sensitive, but can be more than one character if you are using ORACLE V6.0 or later.

For example, if you enter in your **/etc/sqlnet** file:

```
boston      T: bostonsales:P
```

where

boston	Is the alias for the entire connect string.
T	Is the driver prefix for TCP/IP.
bostonsales	Is the host name.
P	Is the SID for the bostonsales database.

Then you can enter

```
$ sqlplus scott/tiger@boston
```

instead of

```
$ sqlplus scott/tiger@T:bostonsales:P
```

and SQL*Net TCP/IP will connect using the connect string identified by the alias "boston” in the **/etc/sqlnet** file.

Specifying the Default Host with the Alias **sqlnet**

If a machine is acting as a client, you may want to specify a default host to connect to when using SQL*Net TCP/IP. To do this, modify the **/etc/hosts** file on the client by adding the alias **sqlnet** for the default host. If you fail to identify a host in the connect string, the application you are using is automatically routed to the host specified by the **sqlnet** alias.

For example, a host named "bostonsales" might be referenced like this in the **/etc/hosts** file:

```
89.0.1.20    bostonsales   sqlnet
```

where

89.0.1.20	Is the Internet address of the remote host.
bostonsales	Isa name for the remote host.
sqlnet	Is an alias for the bostonsales host. The **sqlnet** alias identifies bostonsales (or any host name it is associated with) as the default host. That is, if no host name is specified at logon time, TCP/IP will search for the host name corresponding to the **sqlnet** alias.

If you do not specify a full connect string, SQL*Net TCP/IP searches for the host name corresponding to the **sqlnet** alias. For example, if you specify

```
$ .sqlplus scott/tiger@T:
```

the default database on the default remote machine will be used.

Note: You do not need to use the same default host on all systems on the network. Each client can have its own default SQL*Net TCP/IP host.

Using TWO_TASK to Set Defaults

You can use the TWO.TASK environment variable to avoid specifying full connect strings. If you do not specify a driver or an alias on the command line, SQL*Net TCP/IP reads the TWO_TASK environment variable to determine the default alias or the default driver and its parameters.

For example, instead of entering

```
SQL> sqlplus scott/tiger@local
```

where "local" is the alias for a local database, you could setup "local" as a default using one of the following methods.

In the Bourne shell or ksh, enter:

```
$ TWO_TASK="local" ;  export TWO_TASK
```

In the C shell, enter:

```
$ setenv  TWO_TASK  “local”
```

Then, to connect to the local database with SQL*Plus and enter:

```
$ sqlplus   scott/tiger
```

Identifying the Available Databases (Servers Only)

On server systems, you must identify the local databases that are available for clients to use. The **/etc/oratab** file is the default file to use for this function.

Note: On some UNIX systems, the oratab file resides in **/opt/var/oratab.**

The oratab file lists the ORACLE SID to ORACLE home directory mappings of those databases that are available on the server system. When SQL*Net TCP/IP is installed on the server, the automatic installation script updates the file. However, whenever new databases become available, you must manually add an entry to this file.

Note: You do not need to list all databases in **/etc/oratab.** Only include those databases that are required for connections by remote clients.

The syntax for identifying an available database is:

```
ORACLE_sid:ORACLE_home:  [Y |N]
```

where

ORACLE_sid	Is the ORACLE SID. It can be longer than one character if you are using ORACLE V6.0.
ORACLE_home	Is the full path name of the root directory of your ORACLE system directory.
Y	Indicates that you want this database to become available when the system starts.
N	Indicates that you do not want this database to become available when the system starts.

Each database must be listed on a separate line and begin in the first column. Here is an example of three valid entries in the **/etc/oratab** file:

```
A:usr/users/oracle:Y
SHIPPING:/usr/users/oradev/ship:N
FINANCE:/usr/users/finance:N
```

The following entry illustrates the use of **orasrv** to start a server other than ORACLE by adding the name of the server process in the fourth field, as in:

```
MSERVER:/oracle/6.0:N:mserver
```

where mserver might be the Oracle*Mail server. If this field is omitted, the ORACLE shadow process will run as usual.

For more information about logging onto the database, refer to Chapter 2, "Using SQL*Net TCP/IP."

Starting the Server Process (Servers Only)

Before SQL*Net clients can connect to the server, the DBA (or anyone with the necessary authority) must start the server process on the host system. A server process listens for client connection requests and directs them to the correct database. SQL*Net TCP/IP uses a server process named **orasrv.**

You can start **orasrv** by using one of the following options:

- use an entry in the **/etc/rc** file to automatically start **orasrv** when the UNIX system is booted
- start **orasrv** manually on the command line
- use the **tcputl** utility that controls and monitors **orasrv**

Note: When any SQL*Net TCP/IP server process is running, you can use the UNIX **netstat** command to determine what server process is running. To use **netstat,** enter the following command:

```
$ netstat -a
```

If the server is using the service name **orasrv,** entering the command produces the following display:

```
Proto Recv-Q Send-Q  Local Address Foreign Address (state)
tcp        0      0   *.orasrv       *.*       LISTEN
```

If the service name **tcptlisrv** is in use (the service process for SQL*Net TCP/IP TLI), then the output looks like this:

```
Proto Recv-Q Send-Q  Local Address Foreign Address (state)
tcp        0      0   *.tcptlisrv  *.*       LISTEN
```

Starting **orasrv** Automatically

To start **orasrv** when UNIX is booted, include the following lines in the **/etc/rc** file (assuming the ORACLE home directory is **/usr/users/oracle):**

```
ORACLE_HOME=/usr/users/oracle
SU -oracle -c $ORACLE_HOME/bin/orasrv
```

Starting **orasrv** Manually

To start the server process without rebooting UNIX, use the **tcpctl** shell script as described in the following section.

Using SQL*Net TCP/IP

The following is the syntax of the connect string for the TCP/IP driver:

```
T: [hostname, servicename: sid,
buffer_ size] : [conn_retries, keepalive, VSN1, break_mode]
```

where

T:	Specifies the TCP/IP driver for communication.

Note: The remaining parameters are optional, positional, and must be separated by commas. You can leave out any parameter, but you must leave in the comma separators for any parameter you skip.

hostname	Is the host containing the database that is the target of your connection request; it must be defined in the **/etc/hosts** file. The host must be properly configured to run SQL*Net TCP/IP. **Note: If** you specify the *hostname,* you must also specify the *sid.*
servicename	Is the name of the server in the **/etc/services** file that you want to connect to the TCP driver. There are two options: **tcptlisrv** and **orasrv.** The **orasrv** server connects to a system using the standard TCP/IP (sockets-based) driver. The **tcptlisrv** server connects to a host system that supports the TCP/IP TLI driver. This parameter is needed if you are connecting to an MVS system running IBM TCP/IP or to VM where the service is actually equivalent to the SID. The only other use is on a UNIX platform where you want to have more than one orasrv running at the same time to reduce the number of concurrent connection requests that could possibly be made to the server, or when you want to connect to a server which uses the TCP/TLI driver.
sid	Is the system ID (SID) for the database you want to access. **Note:** If you specify the *sid* you must also specify the *hostname.*
buffer_size	Is the size of the buffer allocated to hold data being passed. Its default value is generally set to 4K on most platforms. If you use array fetches, then a larger *buffer size* will generally make SQL*Loader and Import/Export run faster. On machines with limited memory, you may want to reduce the buffer size to

	conserve memory. Any new value you enter should fall between 4096 and the size of the message header, which is currently 4 bytes.
conn_retries	Specifies the number of times the TCP/IP driver will attempt to establish a connection before giving up.
keepalive	Tells the host to look for PCs or other remote machines that get disconnected from the network after the connection has been established and close those connections. YES is the default.
VSN1	Explicitly sets the SQL*Net version number to 1 which it is by default.
break_mode	Specifies whether you want to send in-band or out-of-band breaks.

Using tcpctl to Monitor orasrv Activities

The **tcpctl** shell script calls **tcputl,** a utility that controls and monitors the activities of **orasrv.** You can use **tcpctl** to obtain information about the status of the **orasrv** listening process.

Syntax of tcpctl

The tcpctl command is used in the following format:

```
tcpctl [port <value>] [start <options> I stat I versopm I
stop I log I debug I timeout]
```

where

port	Specifies the port number to which you will connect. When omitted, **tcpctl** connects to the default **orasrv** port (typically 1525).
start	Starts the **orasrv** process.
stat	Displays the status of **orasrv.** You can use this command to check if **orasrv** on the remote node is running.
version	Displays the version number of **orasrv.**
stop	Stops the **orasrv** process.
log	Turns logging mode on/off. The default is off.
debug	Turns debugging mode on/off. The default is off.
timeout	Set or view the handshake timeout.

If you invoke **tcpctl** without specifying any actions, a help screen showing its syntax is displayed.

The **stat** and **version** options can be used across a network. For example:

```
$ tcpctl stat @hostname
```
or

```
$ tcpctl version @hostname
```

For network security, only the user who owns the **orasrv** process can toggle logging and debugging modes, or stop **orasrv.**

If you want to start **orasrv** using the **orasrv** flags described in the previous section, you can invoke **orasrv** directly instead of using **tcpctl start.**

When using **tcpctl** to start **orasrv,** you can specify the following additional **orasrv** options:

```
[mapfile] [I |O] [logon |logoff]
[debugon |debugoff |debug= <level>] [dbaon |dbaoff]
[opson |opsoff] [opsrooton |opsrootoff] [-0<logfile>]
[port=<port number or service name] [listen=<queue size>]
[timeout=<seconds>] [forkon |forkoff] [detachon |detacnoff]
```

Starting orasrv

To start **orasrv,** type:

```
$ tcpctl start
```

Messages similar to the following display:

```
tcpctl : log file is /usr/oracle/tcp/log/orasrv.log
tcpctl : SID mapping file is /etc/oratab
tcpctl : server will be run under oracle
tcpctl : logging mode is on

orasrv: Version 1.2.7.8.1 - Production on Tue May 10
09:37:10 1994

Copyright (c) Oracle Corporation 1979, 1994.  All rights
reserved.

Starting server on port 1525.
tcpctl : server has been started
```

The message "Server will be run under oracle’’ means that the **orasrv** process will be owned by the user "oracle”.

Getting orasrv status

To see the status of the **orasrv** process, type:

```
$ tcpctl stat @hostname
```

Messages similar to the following display:

```
tcputl: Status summary follows
Server is running:
Started               : 14-APR-94 14:48:13
Total connections     : 117
Total rejections      : 5
Active subprocesses   : 3
ORACLE SIDs           : S,Y,G,A,M, (null),SALES, FINANCE
Default SID           : (null)
Logging mode is ENABLED.
DBA logins are DISABLED.
OP$ logins are DISABLED.
OP$ROOT logins are DISABLED.
Orasrv is detached from the terminal.
Break mode = OUT OF BAND.
Debug level = 1
No timeout (on orasrv handshaking) .
Length of listen queue = 10
Orasrv logfile = /tmp/orasrv.log
Orasrv mapfile = /etc/oratab
```

where

Started	Is the date and time when **orasrv** was brought up.
Last connection	Is the date and time of the last request for a SQL*Net TCP/IP connection.
Total connections	Is the total number of successful SQL*Net TCP/IP connections since **orasrv** was started.
Total rejections	Is the total number of SQL*Net TCP/IP connections rejected either because of human error (e.g., typing the wrong SIDs), or because of system error (e.g., network failure).
Active subprocesses	Is the number of active ORACLE shadow processes utilizing SQL*Net TCP/IP on this node.
ORACLE SIDs	Is a list of SIDs in the **/etc/oratab** file of this node. Note that the databases corresponding to these system identifiers may or may not be running.

Default SID	Is the default system identifier to be used if no SID is stated in the connect string. This is always null on UNIX systems.

Stopping orasrv

To bring down the **orasrv** process, type:

```
$ tcpctl stop
```

Messages similar to the following display:

```
tcputl: checking user permission...
tcputl: server has been stopped
```

Only the user who started up **orasrv** has the privilege to shut it down.

Sample orasrv.log file

The **orasrv** process logs all types of errors. If you cannot establish a SQL*Net TCP/IP connection, examine the **orasrv.log** file for the error diagnostic message.

In the sample **orasrv.log** file shown below, **orasrv** was started on 11/23/88 by user "oracle". The modification date of **/etc/oratab** was 11/19/88. Whenever **/etc/oratab** is modified, **orasrv** will reload it and log its modification date.

```
SQL*Net TCP/IP Network Server
Started at 23-NOV-88 13:17:05 by oracle
LOGGING MODE IS ENABLED

Reloading /etc/oratab...
Last modified: Thu Nov 19 13:35:34 1988
Connection request from wrvms at 24-NOV-88 06:08:17
STATUS request from ultravax at 2-DEC-88 12:18:52
Connection request from huffy at 11-DEC-88 10:30:11
ERROR: SID lookup failure
Connection request from justine at 11-DEC-88 14:14:36
Connection request from indigo at 15-DEC-88 10:52:58
VERSION request from oracle at 15-DEC-88 16:59:52
STOP request from ultravax at 13-JAN-89 16:03:02

SQL*Net TCP/IP Network Server
Shutdown at 13-JAN-89 16:03:04
```

An error occurred when this **orasrv** process was running. The error was caused by specifying an incorrect SID in the connection request from node ''buffy'', causing a "SID lookup failure". Error messages always describe the last connection request.

Testing SQL*Net TCP/IP

You can verify that TCP/IP is configured correctly by using the **checkTCP** utility. This utility runs a series of tests automatically. To run **checkTCP,** refer to the next section. You can also test your system manually. To run the tests manually, skip the next two sections and refer to "Testing TCP/IP Manually."

Testing TCP/IP Automatically

Perform the following steps to check your configuration.

Step 1 Move to the proper directory by typing:

```
% cd $ORACLE_HOME/tcp/install
```

Step 2 To automatically verify that TCP/IP is configured correctly, use the **checkTCP** utility. The syntax for using **checkTCP** is as follows:

```
% checkTCP {[-a] |[-1] [-2] [-3] [-4] [-5] [-6]}
```

where

checkTCP	Is the required keyword used for starting the system test.
-a	Runs all the system tests. "a" is the default.
-1	Verifies that **orasrv** is installed.
-2	Checks the ORACLE network services.
-3	Checks for the presence of **/etc/hosts.**
-4	Lists the available databases in the **/etc/oratab** file.
-5	Checks whether **orasrv** will be started when the system boots up,
-6	Verifies that the TCP/IP driver is linked to the database kernel.

The following example shows how to run all the system tests using **checkTCP:**

```
% checkTCP -a
```

When you run **checkTCP,** you should receive a response similar to the following output. If a test does not complete successfully, **checkTCP** displays a message that defines the cause of the problem and suggests a course of action.

Test 1 Verify **orasrv** installation. The following message indicates that **orasrv** is installed correctly:

```
orasrv is installed in /oracle5/bin
```

Test 2

Check the ORACLE network services. The following message lists the connecting port number found in the **/etc/services** file:

```
orasrv is defined in /etc/services
orasrv will listen on port number 1525
```

Test 3

Check the host list. The following message verifies that the **/etc/hosts** file exists:

```
/etc/hosts exists
```

Test 4

List the available databases. The following messages verify that the **/etc/oratab** file exists and includes the available databases and corresponding SIDs:

```
/etc/oratab exists
Database SID P has ORACLE_HOME  /oracle6
```

Test 5

Check whether **orasrv** will be started when the system boots. The following messages indicate where **orasrv** is located, and whether or not it will be started when the system boots:

```
orasrv is in /etc/rc
and will be started on boot
```

Test 6

Verify the TCP/IP link to the ORACLE kernel. The following message tells where the TCP/IP driver is linked:

```
TCP/IP driver linked into
/oracle6/bin/oracle
```

Testing TCP/IP Manually

Perform the following steps to test your configuration.

Step 1 Check the ORACLE TCP/IP server by typing:

```
$ 1s $ORACLE_HOME/bin/orasrv
```

Step 2 Before you can connect to another computer, it must be running. To determine whether the machine you want to access is running, type:

```
$ ping hostname
```

Step 3 Check the network services by listing the contents of the **/etc/services** file by typing:

```
$ cat /etc/services
```

Your screen should display a response that includes the line:

```
$ orasrv 1525/tcp
```

Step 4 Make sure that every remote computer you want to connect to is listed in the **/etc/hosts** file. To check the **/etc/hosts** file, type:

```
$ cat  /etc/hosts
```

This should return a response such as:

```
89.0.1.105   salesvms   sales
89.0.1.160   financevms   finance
```

The Internet address is listed at the beginning of each line, followed by one or more names for the computer at that address.

Step 5 If you are working on the host system, you can check the **/etc/oratab** table to see the databases which remote users can connect to. To check the **/etc/oratab** file, type:

```
$ cat /etc/oratab
```

This will display one line for each database that is accessible to remote users. The following example of an **/etc/oratab** file contains the SID, the name of the ORACLE home directory for that database, and the startup instruction.

```
D:/oracle6:Y
finance:/usr/finance:N
DEV:/usr/user/dev/ship:N
```

Step 6 To verify that the **orasrv** process is working correctly, you can check the file of commands executed at system boot time to see if the server will start automatically when the system is booted. To check the **orasrv** process, type:

```
$ grep orasrv /etc/rc
```

This command should return the following message:

```
SU -oracle -c $ORACLE_HOME/ bin/orasrv
```

To verify that the server is running, enter one of the following commands:

On UNIX System V systems, type

```
$ ps -ef |grep orasrv
```

On Berkeley BSD systems, type

```
$ ps -ax |grep orasrv
```

This Command should display two lines from the process status list: one for the **orasrv** process and one for the **grep orasrv** process.

Using SQL*Net TCP/IP TLI

This section explains SQL*Net and the TCP/IP TLI product. The TCP/IP TLI product provides task-to-task communication across networks consisting of certain variants of UNIX. It can also connect to the SQL*Net TCP/IP driver on other machines.

Both the SQL*Net TCP/IP and SQL*Net TCP/IP TLI drivers use the TCP/IP protocols. However, TCP/IP TLI uses the TLI programmatic interface (instead of Berkeley sockets) for compatibility purposes. Also, SQL*Net TCP/IP TLI listeners typically use different ports than SQL*Net TCP/IP, so clients must be careful to connect to the correct port number.

Refer to the *ORACLE Installation and User's Guide* written for your system for installation and configuration details particular to your system.

Following is the syntax of the connect string for the TCP/IP TLI driver:

```
TT: [hostname, servicename:sid, buffer size]
```

where

TT:	Specifies the TCPTLI driver for two-task communication. You can use the complete driver name (TCPTLI_TWO_TASK) or just its prefix (TT:)

Note: The remaining parameters are optional, positional, and must be separated by commas. You can leave out any parameter, but you must leave in the comma separators for any parameter you skip.

hostname	Is the host containing the database that is the target of your connection request; it must be defined in the **/etc/hosts** file. The host must be properly configured to run SQL*Net TCP/IP TLI. **Note:** If you specify the *hostname* you must also specify the *sid.*
servicename	Is the name of the server in the **/etc/services** file that you want to connect to the TLI driver. There are two options: **tcptlisrv** and **orasrv.** The **tcptlisrv** server connects to a host system that supports the TCP/IP TLI driver; it is the default. The **orasrv** server connects to a system using the standard TCP/IP (sockets-based) driver.
sid	Is the system ID (SID) for the database you want to access. For the TCP/IP TLI driver, the default SID

	is NULL ("*"). If you do not define an SID, the default database is used. **Note:** If you specify the *sid* you must also specify the *hostname.*
buffer size	Is the size of the buffer allocated to hold data being passed. Tuning this parameter could improve SQL*Net TCP/IP TLI performance. Its default value is hard-coded in the driver. If you use array fetches, then a larger *buffer size* will generally make ODL, SQL*Loader, and Import/Export run faster. On machines with limited memory, you may want to reduce the buffer size to conserve memory. Any new value you enter should fall between 4096 and the size of the message header, which is currently 4 bytes.

You have two options to invoke the TCP/IP TLI driver:

```
$ sqlplus scott/tiger@TT:
```

which invokes the TCP/IP TLI driver on the host whose server name is defined in the **/etc/hosts** file, and

```
SQL> connect scott/tiger@TT: dread,orasrv
```

which invokes the TCP/IP TLI driver on the "dread" host, that uses a server named **orasrv.**

SQL*Net TCP/IP TLI Configuration Tasks

The SQL*Net TCP/IP TLI configuration tasks are the same as those for socket-based TCP/IP. They are listed below. Tasks marked with a asterisk *, are performed exactly as the configuration tasks for SQL*Net TCP/IP. Refer to the SQL*Net TCP/IP sections for information on performing these tasks.

- test the TCP/IP TLI program *
- identify the TCP/IP TLI port number and service
- identify the available hosts *
- set aliases and defaults *
- identify the available databases *
- start the server process

Identifying the Port Number and Service (Clients and Servers)

The SQL*Net TCP/IP protocol uses a port number to indicate where clients and servers connect. To avoid conflict with SQL*Net TCP/IP, Oracle Corporation recommends using the number 1527 for SQL*Net TCP/IP TLI. Whatever number you choose, all SQL*Net TCP/IP TLI clients and servers must use the same number. The **/etc/services** file for all clients and servers should contain the following line:

```
tcptlisrv  1527/tcp
```

If you want to use SQL*Net TCP/IP TLI with SQL*Net TCP/IP, the following information should be included in your **/etc/services** file:

```
orasrv 1525/tcp
```

Identifying the Current Service Name

When the SQL*Net TCP/IP TLI server process is running, you can use the **netstat** command to inquire about the service name currently in use by the server. If the server is using the service name **orasrv,** entering the command

```
netstat -a
```

would produce the following display:

```
Proto Recv-Q Send-Q  Local Address  Foreign Address (state)
tcp        0      0  *.orasrv         *.*      LISTEN
```

If the default service name **tcptlisrv** is in use, then the output is similar to the following:

```
Proto Recv-Q Send-Q  Local Address Foreign Address (state)
tcp        0      0  *.tcptlisrv      *.*      LISTEN
```

Note: When the SQL*Net TCP/IP TLI server program is listening at **orasrv,** all SQL*Net TCP/IP TLI clients must specify the service name **orasrv** either in the connect string or in the environment.

It is possible to run two TCP/IP TLI server processes, one listening at **orasrv** for TCP/IP clients, and another listening at **tcptlisrv** for TCP/IP TLI clients, but this is **not** recommended.

Starting the Server Process (Servers Only)

SQL*Net TCP/IP TLI uses **tcptlisrv** to accept connection requests from SQL*Net clients. You can start **tcptlisrv** by using one of the following options:

- use an entry in the **/etc/rc** file to automatically start **tcptlisrv** when the UNIX system is booted
- start **tcptlisrv** manually on the command line
- use the **tcptlictl** utility that controls and monitors **tcptlisrv**

Starting **tcptlisrv** Automatically

To start **tcptlisrv** when UNIX is booted, include the following lines in the **/etc/rc** file (assuming the ORACLE home directory is **/usr/users/oracle):**

```
ORACLE_HOME=/usr/users/oracle.
SU -oracle –c $ORACLE_HOME/bin/tcptlisrv
```

Starting **tcptlisrv** Manually

To start the **tcptlisrv** server process manually, use the following command:

```
$ tcptlictl start
```

The following **tcptlisrv** options can be specified:

```
[mapfile]  [I |O]  [logon | logoff]  [debugon | debugoff]
-Ologfile  -Sservice name
```

where

mapfile	Specifies the file containing ORACLE SID to ORACLE home mapping. The default is **/etc/oratab.**
I	Specifies in-band breaks. This is the default.
O	Specifies out-of-band breaks.
logon	Turns logging activities on. This is the default.
logoff	Turns logging activities off.
debugon	Turns debugging activities on.
debugoff	Turns debugging activities off. This is the default.
-O	Specifies the log file.
logfile	Names the log file to be used. The default is **$ORACLE_HOME/tcptli/log/tcptlisrv.log.** If the log file already exists, new log messages will be appended to that file. If the specified log file cannot be created, all log messages will be written to **/tmp/tcptlisrv.log.** We recommend using the default value whenever possible.
-S	Specifies the service name.
service name	Names the service to be used (either **tcptlisrv** or **orasrv.** It should be the corresponding entry in the **/etc/services** file. The default is **tcptlisrv.**

Using tcptlictl to Monitor tcptlisrv Activities

The **tcptlictl** shell script calls **tcptliutl,** a utility that controls and monitors **tcptlisrv.**

Syntax of tcptlictl

If you invoke **tcptlictl** without specifying any actions, a help screen similar to the following displays:

```
tcptlictl  [start | stat |version |stop | log |debug]
```

where

start	Starts the **tcptlisrv** process.
stat	Displays the status of **tcptlisrv.**
version	Displays the version number of **tcptlisrv.**
stop	Stops the **tcptlisrv** process.
log	Turns logging mode on/off. The default is off,
debug	Turns debugging mode on/off. The default is off.

The **stat** and **version** options can be used across a network. For example:

```
$ tcptlictl stat @hostname
```

or

```
$ tcptlictl version @hostname
```

For network security, only the user who owns the **tcptlisrv** process can toggle logging and debugging mode, or stop **tcptlisrv.**

If you want to start **tcptlisrv** using the **tcptlisrv** flags described in the previous section, you can invoke **tcptlisrv** directly instead of using **tcptlictl start.**

Starting tcptlisrv

To start **tcptlisrv,** type:

```
$ tcptlictl start
```

The following messages display:

```
tcptlictl : log file is
/usr/oracle/tcptli/log/tcptlisrv.log
tcptlictl: SID mapping file is /etc/oratab
tcptlictl : server will be run under oracle
tcptlictl: logging mode is on
```

```
tcptlisrv: Version 1.0.2 - Production on Thu Mar 3
00:04:20 1988
Copyright (c) 1986, Oracle Corporation, California, USA.
All rights reserved.
tcptlictl: server has been started
```

The message "Server will be run under oracle'' means that the **tcptlisrv** process will be owned by the user ''oracle".

Getting tcptlisrv Status

To see the status of the **tcptlisrv** process, type:

```
$ tcptlictl stat @hostname
```

The following messages display:

```
tcptliutl: Status summary follows
Server is running:
Started               : 5-FEB-88 14:48:13
Last connection       : 2-MAR-88 16:41:44
Total connections     : 117
Total rejections      : 5
Active subprocesses   : 3
ORACLE SIDs           : s,Y,G,A,M, (null),K,Q,SALES,FINANCE
Default SID           : (null)
Logging mode is DISABLED
```

where

Started	Is the date and time when **tcptlisrv** was started.
Last connection	Is the date and time of the last request for a SQL*Net TCP/IP TLI connection.
Total connections	Is the total number of successful SQL*Net TCP/IP TLI connections since **tcptlisrv** was started.
Total rejections	Is the total number of SQL*Net TCP/IP TLI connections rejected either because of human error (e.g., typing the wrong SID), or because of system error (e.g.,network failure).
Active subprocesses	Is the number of active ORACLE shadow processes utilizing SQL*Net TCP/IP TLI on this node.
ORACLE SIDs	Is a list of SIDs in the **/etc/oratab** file of this node. Note that the databases corresponding to these SIDs may or may not be currently running.

Default SID	Is the default SID if an SID is not included in the connect string. The default SID is always null on UNIX systems.

You can also use **tcptlictl stat** to check whether **tcptlisrv** on the remote node is running **tcptliutl.**

Stopping tcptlisrv

To bring down the **tcptlisrv** process, type:

```
$ tcptlictl stop
```

The following messages display:

```
tcptliutl:  checking user permission...
tcptliutl:  server has been stopped
```

Only the user who started **tcptlisrv** has the privilege to shut it down.

tcptlisrv.log file

The **tcptlisrv** process logs all types of errors. If you cannot establish a SQL*Net TCP/IP TLI connection, examine the log file for the error diagnostic message.

In the following sample **tcptlisrv.log** file, **tcptlisrv** was started on 11/23/88 by user oracle. The modification date of **/etc/oratab** was 11/ 19/88. Whenever **/etc/oratab** is modified, **tcptlisrv** reloads it and logs its modification date.

```
SQL*Net TCP/IP Network Server
Started at 23-NOV-88 13:17:05 by oracle
LOGGING MODE IS ENABLED

Reloading /etc/oratab...
Last modified: Thu Nov 19 13:35:34 1988
Connection request from wrvms at 24-NOV-88 06:08:17
STATUS request from ultravax at 2-DEC-88 12:18:52
Connection request from huffy at 11-DEC-88 10:30:11
ERROR:  SID lookup failure
Connection request from justine at 11-DEC-88 14:14:36
Connection request from indigo at 15-DEC-88 10:52:58
VERSION request from oracle at 15-DEC-88 16:59:52
STOP request from ultravax at 13-JAN-89 16:03:02

SQL*Net TCP/IP Network Server
Shutdown at 13-JAN-89 16:03:04
```

An error occurred when this **tcptlisrv** process was running. The error was caused by specifying an incorrect SID in the connection request from node "buffy'',causing an ''SID lookup failure". Error messages always describe the last connection request.

CHAPTER

4

SQL*NET TCP/IP FOR VAX OPEN VMS AND ALPHA OPEN VMS SYSTEMS

This chapter explains how to configure SQL*Net TCP/IP for the VAX OpenVMS and Alpha OpenVMS operating systems. This chapter assumes that SQL*Net TCP/IP has been installed on your system. Refer to the *ORACLE for VAX OpenVMS Installation Guide* for details on installing SQL*Net TCP/IP on VAX OpenVMS systems. Refer to the *ORACLE for Alpha AXP OpenVMS Installation Guide* for details on installing SQL*Net TCP/IP on Alpha AXP OpenVMS systems.

The TCP/IP protocol is a software layer that cooperates with Ethernet to provide task-to-task communication between any two systems. SQL*Net can use this protocol for communication between ORACLE processes on OpenVMS systems and other systems running SQL*Net TCP/IP. An OpenVMS system can act as both a client or a server.

SQL*Net TCP/IP supports the Wollongong Pathway, Multinet, Fusion, and DEC TCP (formerly UCX) implementations of TCP/IP on OpenVMS systems. On Alpha OpenVMS systems, Multinet, Wollongong Pathway, and DEC TCP are supported. If you have

DECnet available on your machine, you can support the TCP/IP protocol using the same Ethernet controller that DECnet uses.

SQL*Net also supports the TCPWARE TCP/IP implementation on VMS. The configuration for this is the same as that for the Wollongong Pathway TCP/IP (for VAX only.)

For more details about system requirements, refer to the *ORACLE for DEC VAX OpenVMS Installation Guide* or the *ORACLE for Alpha AXP OpenVMS Installation Guide.*

Configuration Tasks for VAX OpenVMS

To define either a client or a server on the network, you must tell SQL*Net TCP/IP about the system configuration by completing the configuration tasks in this section.

TCP/IP System Test

To ensure that the communications hardware and software are working correctly, connect to your local machine using the TCP/IP TELNET procedure. To use TELNET, type:

```
$ TELNET hostnarne
```

where *hostname* is the host (defined in the HOSTS file) to which you want to connect. The name and location of the HOSTS file depends on your TCP/IP vendor:

- for DEC TCP it is in SYS$COMMON:[SYSEXE]UCX$HOSTS.DAT
- for Wollongong it is in TWG$COMMON:[NETDIST.ETC]HOSTS
- for Multinet it is in MULTINET_ROOT:[MULTINET]HOSTS.LOCAL

For example, if you specified the hostname of an OpenVMS host, you should receive the familiar “Username:” prompt for logging onto that system. If you receive this prompt, the hardware and software are working and you can use CTRL-Z to cancel the logon procedure and return to the DCL prompt. If you cannot perform this simple test, you will need to resolve problems with your vendor’s TCP/IP product before trying to configure SQL*Net.

Identifying the Port Number (Clients and Servers)

The SQL*Net TCP/IP protocol uses a port number which you must specify to indicate where clients and servers connect. Oracle Corporation recommends using the number 1525. Whatever number you choose, all clients and servers on the network must use the same number.

Note: For **DEC TCP** implementations only, you do not need to specify the port number. The SQL*Net TCP/IP protocol for the DEC TCP implementation will always use the next freely available port number on the system. Therefore, it is not possible to predetermine the port number.

To make the port number known to the TCP/IP driver, you must add a line to the services file (if the services file does not already exist, you must create it as described below). The services file should be located in one of the following directories, depending on your TCP/IP implementation.

Note: You need only to identify the port number once--when you perform the initial system configuration tasks.

For **Wollongong** implementations, the file is named:

```
TWG$TCP:  [NETDIST.ETC]SERVICES
```

The following information identifies the port number where clients and servers connect:

```
ORASRV  1525/TCP
```

where

ORASRV	is the name of the server process.
1525	identifies the port number you have chosen for SQL*Net TCP to use.
TCP	identifies the protocol in use.

If your services file does not include this information, add this line to the file:

```
ORASRV  1525/TCP
```

For **Multinet** implementations, the services and host files are combined in one file named:

```
MULTINET:HOSTS.LOCAL
```

The following information identifies the port number where clients and servers connect

```
SERVICE : TCP : 1525 : ORASRV :
```

where

SERVICE	is a required keyword.
TCP	identifies the protocol in use.
1525	identifies the port number you have chosen for SQL*Net TCP to use.
ORASRV	is the name of the server process.

If your services file does not include this information, add this line to the file:

```
SERVICE : TCP : 1525 : ORASRV :
```

Configuring SQL*Net TCP/IP Servers

Identifying the Available Databases (Servers Only)

On server systems, you must identify the databases that are available for client connection requests. You must do this during the initial system configuration and each time you want to identify a new database on the server.

For each database on the server that you want to make available to client processes, you must add an entry to the CONFIG.ORA file. For both ORACLE V6.0 and ORACLE7,CONFIG.ORA should be located in ORA_NETCONFIG. This entry identifies the address and system ID of the database that clients can connect to on your system.

Note: To make each database available to every client on the TCP/IP network when you install your ORACLE database, you must either:

- put each available database in the same group, or
- make each database available system-wide; in this case, the ORACLE account must have SYSNAM privileges

The general syntax for identifying an available database is:

```
SQLNET SIDMAP  systemid = directory [filename]
```

where

SQLNET SIDMAP	Is the keyword used to map the location of the database.
systemid	Is an identifier that clients will specify when connecting to this database. It can be up to six characters in length.
directory	Is the directory where the command file (used to setup access to the database) exists.

filename	This is an optional field. You can enter the name of the command file that sets up the logicals that provide clients with database access. The default is ORAUSER.COM.

Note: The systemid entries must be unique in CONFIG.ORA but do not need to be the actual SID for the database although it is common to do this. Oracle Corporation recommends that you make all SIDs on a single machine unique.

Here is an example of two valid entries and a default entry in a CONFIG.ORA file. Any line beginning with REM indicates a comment.

```
REM  **CONFIG.ORA**
REM
SQLNET SIDMAP Q = DISK$WR4$:[TCPIP]
SQLNET SIDMAP V6TEST = DISK$WR2$:[V6TEST]SETUP.COM
SQLNET SIDMAPDFLT = Q
```

The entry "V6TEST" has been established as the systemid for the database located at DISK$WR2$:[V6TEST].SIDMAPDFLT is the keyword used to map the location of the default database. This entry can be placed on any line in the CONFIG.ORA file. If CONFIG.ORA does not contain a SIDMAPDFLT entry, then the first SIDMAP entry is used as the default.

If the host machine name was HQVMS, users on remote VMS nodes could log onto the database by entering:

```
$  SQLPLUS  SCOTT/TIGER@T:HQVMS:V6TEST
```

If a remote user did not specify a SID and logged on by entering

```
$  SQLPLUS  SCOTT/TIGER@T:HQVMS
```

then the "Q" database would be used, because it is specified as the default (SIDMAPDFLT). For more information about logging on to the database, refer to the section, "Using SQL*Net with TCP/IP," at the end of this chapter.

Access Control (Servers Only)

You can use the CONFIG.ORA file to set different levels of access control.

With SQL*Net TCP/IP V1.2, you can set the following levels of access control:

- host-level access control
- user-level access control
- user mapping table (UMT)
- password-level access control

Host-level Access Control

You can use the CONFIG.ORA file to set up the Valid Node Table (VNT). The VNT identifies nodes on a TCP/IP network that are eligible to connect as client machines with SQL*Net TCP/IP to your machine. The VNT is comprised of entries in the CONFIG.ORA file with the following format:

```
SQLNET VALIDNODE = [qualifier] nodename
```

where

SQLNET VALIDNODE	Is the keyword used to identify a VNT entry.
qualifier	Is an optional field that can be set to indicate whether access is available by setting one of the following parameters: * grants “wildcard” status which identifies all clients as eligible for database access. - denies access to *nodename.* ! enforces password-level access control for connection requests from *nodename.*
nodename	Is the name of the node being qualified for SQL*Net TCP/IP connections.

Note: The machine represented by *nodename* must be identified in the HOSTS file.

For example, setting these entries in the CONFIG.ORA file

```
SQLNET VALIDNODE = libra
SQLNET VALIDNODE = leo
```

would only allow the nodes named “libra” and “leo” access to databases.

If your CONFIG.ORA file looked like this:

```
SQLNET VALIDNODE = -libra
SQLNET VALIDNODE = -leo
SQLNET VALIDNODE = *
```

it would grant access to all nodes on the network except “libra” and “leo”.

Note: VNT entries are in “highest to lowest binding” order. If you use the following entry as the lowest type of binding

```
SQLNET VALIDNODE = *
```

it must be the last entry in the UMT.

User-Level Access Control

User-level access control is enabled if the following entry is set in your CONFIG.ORA file:

```
SQLNET ORASRV_PROXY_LOGIN = YES
```

This entry causes SQL*Net TCP/IP to locally log in each connection under the same username that the client used to log onto the remote machine. This functionality is called a *proxy login*.

If the remote username is not valid for a login to your system, the connection is logged in under a default name. If you omit setting a default name, the default name is the account you are in when you start the server process. You can set a default name by adding the following entry to the CONFIG.ORA file:

```
SQL*NET ORASRV_DFLT_USER = username
```

If you fail to designate a username (as shown below)

```
SQL*NET  ORASRV_DFLT_USER
```

no default name is available for use.

Note: If you enable proxy logins, OPS$ log-ins are possible over a SQL*Net TCP/IP connection. Refer to the *ORACLE RDBMS Database Administrator's Guide* for more information about OPS$ logins.

User Mapping Table

In addition to enabling "straight" proxy logins, you can enable "mapped" proxy logins. A mapped proxy login substitutes a host username of your choice for a client username. To set these mappings, you need to create a User Mapping Table (UNIT). A UMT consists of one or more entries in the CONFIG.ORA file with the following format

```
SQLNET USERNAMEMAP  c_username = username
```

where

SQLNET USERNAMEMAP	Is the keyword used to identify the mapping between c_username and the username.
c_username	Is the username of the client you are qualifying for access control.
username	Is the username designated for your system.

You can use any of the following to qualify a c_username:

*	Is the wild card qualifier that indicates all usernames are eligible for access.
?	Is the qualifier that indicates a c_username is the default username.

If you do not set a qualifier or assign a username for the c_username to run under, that c_username is ineligible for database access on the server system.

Note: The following examples do not contain the entry in the CONFIG.ORA file that activates user-level access control. You must set the following entry in the CONFIG.ORA file to direct SQL*Net TCP/IP to the UMT:

```
SQLNET ORASRV_PROXY_LOGIN = YES
```

The following entry in the UMT

```
SQLNET USERNAMEMAP * =  johnson
```

indicates that all client connections should run under the username "johnson". Using these entries in the UMT

```
SQLNET USERNAMEMAP  rogers = *
SQLNET USERNAMEMAP jones = *
```

grants access to clients running under the usernames "rogers" and jones", but excludes all other users from access.

Note: The wild card qualifier * indicates that the c_username should be the username if you replace username with a * qualifier.

A UMT with the following entries

```
SQLNET USERNAMEMAP rogers =
SQLNET USERNAMEMAP jones = oracle
SQLNET USERNAMEMAP davis = ?
SQLNET USERNAMEMAP  * = *
```

grants all clients on the network access except "rogers". The client "jones" will run under the username "oracle", and "davis" will run under the default username.

Note: UMT enties are in "highest to lowest binding" order. If you use the following entry as the lowest type of binding

```
SQLNET USERNAMEMAP * = *
```

it must be the last entry in the UMT.

Password-Level Access Control

You can use password-level access control to password-protect proxy logins. To enforce password-level control, both the client and server system must be using SQL*Net TCP/IP and the same operating system. If you set the ! qualifier in the VNT for a hostname, all proxy log-ins attempts from that client are subject to password protection. A proxy login attempt will fail unless the password associated with the username is the same for both the client and host machine.

If you set the following entries in the VNT

```
SQLNET VALIDNODE = !SALES_VMS
SQLNET VALIDNODE = !MARKET_VMS
SQLNET VALIDNODE = *
SQLNET ORASRV_PROXY_LOGIN = YES
SQLNET ORASRV_DFLT_USER = GUEST
```

The host machine would:

- grant client access to all nodes on the network
- allow proxy logins from nodes (with a default username of "guest")
- assign a default username of "guest'' to ORASRV processes
- enforces password-protection for proxy login attempts from users named "SALES_VMS" and "MARKET_VMS"

Note: Password-level control access is not supported under the TPS V6.0 release because of a restriction in the link procedure for ORACLE V6.0.

SQL*Net TCP/IP CONFIG.ORA

SQLNET keywords are used in the CONFIG.ORA file to configure the server using the format:

```
    SQLNET keyword = value
```

Valid keywords are defined below:

ORASRV_DFLT_USER=username

Sets a default username for remote login by adding the following entry to the CONFIG.ORA file:

```
SQL*NET  ORASRV_DFLT_USER=username
```

ORASRV_OUTDIR = dir_spec

Specifies the ORASRV .out directory.

ORASRV_PRIORITY = #

Specifies the base priority for the ORASRV process. The default is 4.

ORASRV_PROXY_LOGIN = [YES I NO]

This entry controls whether or not SQL*Net TCP/IP remote users can log in locally under the same username that the client used to log on to the remote machine. See "User-Level Access" earlier in this section. The default is NO.

ORASRV_QUOTA_name = #

Specifies an OpenVMS process quota for the ORASRV process, but only when it runs as a detached process. Otherwise, most of the quotas are pooled with the listener process. The following OpenVMS process quotas can be set (substitute these keywords for name above):

ASTLM
BIOLM
BYTLM
CPULM
DIOLM
ENQLM
FILLM
JTQUOTA
PGFLQUOTA
PRCLM
TQELM
WSDEFAULT
WSEXTENT
WSQUOTA

If these are not specified, they default to the values of the PQL_Dname sysgen parameters. Note that values may be adjusted according to the PQL_Mname sysgen parameters.

ORASRV_WAIT = #

Specifies seconds to wait for the ORASRV processes to start. If not specified, uses a default value of 60 seconds.

TCPSRV_ACCOUNTING = [YES | NO]

Specifies whether or not to write accounting records to the ORA_TCP:ORA_TCPSRV.ACC file. The default is NO.

TCPSRV_ASYNC_IO = [YES | NO]

Specifies whether or not async is enabled. The default is dependent on the TCP/IP implementation. Normally the default is acceptable.

TCPSRV_BACKLOG = #

Sets the listener queue size. Normally the default is acceptable. If not specified, uses a default value of 8.

TCPSRV_BREAKMODE=[I| O]

Sets the server breakmode to inband (I) or outband (O). The default is I. Normally the default is acceptable.

TCPSRV_CLEANUP = [YES | NO | ALWAYS]

Controls the deletion of the ORASRV output log file when the server exits. YES deletes the ORASRV output log file if the server terminates normally. NO preserves all output log files. ALWAYS deletes the server log file regardless of how the server terminated. The default is YES.

TCPSRV_KEEPALIVE = [YES | NO]

Specifies whether or not the system will automatically check that remote nodes on idle connections are still functional. If YES, periodic "heartbeat" packets will be transmitted on idle SQL*Net connections. The remote node should send back a timely acknowledgement. See "Configuring the KEEPALIVE Option" earlier in this chapter for more information. The default is NO.

TCPSRV_MAXCON = #

Specifies the maximum number of simultaneous connections the listener can support. The default is 16.

TCPSRV_NETWAIT = #

Specifies the timeout value for network response. The default is 10.

TCPSRV_RESOURCE_WAIT = [YES | NO]

Specifies whether or not resource wait mode for VMS system services called by the listener is enabled. The default is YES. Oracle Corporation recommends not changing this value.

TCPSRV_REVOKE_DBA = [YES | NO]

Specifies whether or not DBA rights in the server process are enabled. The default is YES. Use this parameter with extreme caution.

Starting and Controlling the Listener Process

Using TCPCTL to Monitor TCPSRV

TCPSRV is the SQL*Net TCP/IP network server listener process that runs on each SQL*Net TCP/IP server node and accepts all incoming connection requests for a specified database on that node. Before SQL*Net clients can connect to the server, the DBA (or anyone with the necessary authority) must start the TCPSRV server process on the host system. The server is usually started automatically by a user-written command file when the ORACLE RDBMS is started.

When a connection request is received, TCPSRV creates art ORACLE server process (ORASRV) and hands over its end of the connection to it, "splicing" together the SQL*Net client process with the ORACLE server process. TCPSRV can then accept the next connection request.

TCPSRV runs as a background process (it is not attached to any terminal). The user interface for TCPSRV is the utility TCPCTL. Following are the uses of TCPCTL:

- starting the SQL*Net TCP/IP server process
- stopping the SQL*Net TCP/IP server process
- listing status information about the server process
- logging important server process information

Using tcpctl to Monitor orasrv Activities

The TCPCTL utility controls, monitors, and obtains information about the status of the ORASRV listening process.

The TCPCTL command is used in the following format

```
TCPCTL [ace] [log] [pause] [reconfig] [sidoff] [start]
 [stat] [stop]
```

where

acc	Toggles between accounting on and off.
log	Set or query the log level.
pause	Toggles between start accepting/stop accepting new connections. Does not affect existing connections.
reconfig	Rereads the CONFIG.ORA file. Using the tcpctl stop followed by tcpctl start has the same effect.
sidoff	Disable new connections for a particular database without affecting the other connections.
sidon	Enable new connections for a particular database without affecting the other connections.
start	Starts the ORASRV process.

stat	Displays the status of ORASRV. You can use this command to check if ORASRV on the remote node is running.
stop	Stops the ORASRV process.

If you invoke TCPCTL without specifying any actions, a help screen showing its syntax is displayed.

Starting the Server Process

SQL*Net TCP/IP uses the TCPSRV process to listen for connection requests from SQL*Net clients and starts the ORACLE server process (ORASRV) when a request is received.

To start the server process, log into the ORACLE account and run the TCPCTL procedure by typing:

```
$ TCPCTL START
```

Note: You should start the SQL*Net TCP/IP server process from a NON-PRIVILEGED account to ensure that accounts using SQL*Net TCP/IP are not given unauthorized access to the host.

When the server process has started, messages similar to the following are displayed:

```
Housekeeping. . .   (hit 'Ctrl Y' to by-pass)
Using DISK$xxx: [ORACLE.V7.NETCONFIG]CONFIG.ORA
Starting server. . .
%RUN-S-PROC_ID , identification of created process is
   0000073D
TCPUTL: listener has been started
```

Other messages may be displayed that indicate the server has not started. Common reasons include:

- The server is already running.
- There is an error in the CONFIG.ORA file.
- The TCP protocol is not properly configured.
- The listener port is not yet available, in which case this message is displayed:

```
TCPUTL: listener port is not (yet) available
```

Stopping the Server Process

You may want to stop the server process for the following reasons:

- to prevent remote connections to local databases
- to reload new information from the CONFIG.ORA file

To stop the server process, type:

```
$ TCPCTL STOP
```

This command stops the server process only when there are no active subprocesses (for example, when there are no more users logged onto ORASRV).

If there are active subprocesses such as users still logged on to ORACLE, TCPSRV will not stop and you see this message:

```
TCPUTL: n active connections;  listener waiting to stop.
```

The server goes into a “stopping” state, refusing any new connection requests, and terminates when the last active subprocess disconnects.

When the server stops, the following message displays:

```
TCPUTL :  listener has been stopped
```

These TCPCTL commands are useful for tasks such as adding SIDMAP entries to the CONFIG.ORA file. To add a new database for remote client connections, follow these steps:

1. Stop the server using TCPCTL STOP.
2. Edit the CONFIG.ORA file as explained earlier in the chapter.
3. Start the server using TCPCTL START.

This process causes the server to read the CONFIG.ORA file again.

Killing the Server Process

If your system is functioning incorrectly and you need to stop the server process when subprocesses are still active (Oracle Corporation does not recommend stopping active subprocesses), type:

```
$ TCPCTL KILL
```

Listing Server Statistics

You can determine the current number of remote connections to your server by typing the following command:

```
TCPCTL STATS
```

A message similar to the following displays:

```
TCPUTL:  listener statistics summary follows:
    Total connections     = 17
    Total rejections      = 2
    Maximum connections   = 5
    SID accesses:
         Q                = 17
         V6TEST           = 2
```

where

Total connections	Is the number of successful connections to the server.
Total rejections	Is the number of unsuccessful comections to the server.
Maximum connections	Is the maximum number of ORASRV connections that have been concurrently active. (that is, the number of users logged on to ORACLE through TCPSRV).
SID accesses	Are access counts for each database SID.
Q and V6TEST	Are SIDs for active databases.

Listing Server Process Status

To check the status of the server process, type:

```
$ TCPCTL STAT
   or
$ TCPCTL STATUS
```

A message similar to the following displays:

```
   TCPUTL: listener is RUNNING; status summary follows:

    Started                  : 24-JAN-94  16:10:47
    Last connection          : 24-JAN-94  16:12:26
    Active connections       : YES (1)
    ORACLE SIDs              : Q, V6TEST
    Default SID              : Q
    Logging level            : NORMAL
    KEEPALIVE option         : ON
    Proxy logins             : NO
    Accounting               : DISABLED
    Version number           : 1.2.9.6.5
```

where

Started	Is the time of TCSRV startup.
Last Connection	Is the time and date of the last SQL*Net connection.
Active connections	Indicates if any ORASRV processes are active and how many are active.

ORACLE SIDS	Are the SIDs of the available databases.
Default SID	Is the default SID (if no SID is specified, the first SID listed in CONFIG.ORA is used).
Logging level	Is the logging level.
KEEPALIVE option	Is the status of the KEEPALIVE option.
Proxy logins	Is the status of proxy logins.
Accounting	Indicates if accounting information is being collected.
Version number	Is the version of SQL*Net TCP/IP you are using.

Listing ORASRV Information

To see information about the ORASRV processes that are providing service to remote clients, type:

```
TCPCTL ACTIVE
```

A message similar to the following displays:

```
TCPUTL:  summary of active connections follows:

Process Name     Pid       User Name SID     I/O Host
---------------  -------   --------- ------- --- -----

ORA_SRVT001_V6T  0000743   ORACLE    V6TEST  129 BOSTON
```

where

Process Name	Is the name of an active process.
Pid	Is the ID number of the active process.
User Name	Is the username of the client requesting the process.
SID	Is the SID of the database that the process is using.
I/O	Is the number of network messages.
Host	Is the name of the remote host.

If there are no active processes, no output is produced.

Displaying the TCPSRV Logfile

To see the latest information (the last 16 lines) about connection requests, type:

```
TCPCTL TAIL
```

TCPCTL Information in the Logfile

When the server process is started, the logging mode is enabled automatically. The log file maintains information about connection requests to the server. The following example of an entry in the log file shows the date and time the remote machine ULTRIXVAX issued a connection request to the server:

```
CONNECTION REQUEST FROM ULTRIXVAX AT 26-AUG-87 13:48:41
```

You can toggle the logging mode off and on by typing:

```
$ TCPCTL LOG
```

Running ORASRV as a Detached Process

By default, the ORASRV server process is created as a subprocess (one ORASRV process is created for every incoming connection request). As a result, there are two possible disadvantages:

1. The VMS quotas associated with the TCPSRV account are pooled (shared) between the TCPSRV process and its subprocesses. This could seriously restrict the number of simultaneous SQL*Net connections.
2. If the TCPSRV process terminates abnormally, all its subprocesses (and all its active SQL*Net connections) will be terminated.

When you enable proxy login, TCPSRV creates ORASRV as a detached process thus avoiding the problems above. In order to enable proxy login for all users, you must enter the following lines in the CONFIG.ORA file:

```
SQLNET ORASRV_PROXY_LOGIN = YES
SQLNET USERNAMEMAP * = ?
```

Note: To effectively enable proxy log-in, the DBA must have the PRMMBX VMS privileges.

Configuring the KEEPALIVE Option

The TCP/IP KEEPALIVE option allows you to automatically check that remote nodes on idle connections are still functional. When you enable the KEEPALIVE option, periodic “heartbeat” packets will be transmitted on idle SQL*Net connections. The remote node should send back a timely acknowledgement.

You enable or disable the KEEPALIVE option on the server end of all SQL*Net connections by including the following entry in the CONFIG.ORA file on the server node:

```
SQLNET TCPSRV_KEEPALIVE = [YES | NO]
```

The default is YES.

Configuring and Connecting Clients

Identifying the Available Hosts (Clients Only)

To make connection requests as a client system, you must identify the host systems that are the targets of your requests. You must do this during the initial system configuration and each time you want to identify a new host system.

SQL*Net uses the TCP/IP protocol facilities to map the name of host machines (servers) to their Internet addresses. The Internet addresses are maintained in the HOSTS file in the SQL*Net directory on the client computer.

For **Wollongong** implementations, the HOSTS file is named:

```
TWG$TCP:[NETDIST.ETC]HOSTS
```

For **Multinet** implementations, the hosts and services files are combined in one file called:

```
MULTINET:HOSTS.LOCAL
```

The mapping for each host listed in the HOSTS file is specified on a single line using this syntax:

```
HOST : internet_addr : hostname [,alias] : misc
```

where

HOST	Is a required keyword.
internet_addr	Is the Internet address of a host computer.
hostname	Is the name of the host.
alias	Is an alternate name for the host. You can have as many as two aliases for any one host.

For example, a host named “BOSTONSALES” might be referenced like this in the HOSTS file:

```
HOST : 89.0.1.100 : BOSTONSALES,BOSTON,ATLANTIC :
```

where

89.0.1.100	Is the Internet address of the remote host.
BOSTONSALES	Is a name for the remote host.
BOSTON	Is an alias setup for the BOSTONSALES host.
ATLANTIC	Is another alias setup for the BOSTONSALES host.

For DEC TCP implementations, the host file is usually not edited directly. Instead, changes are made using the UCX utility which is invoked as follows:

```
$ ucx
UCX>
```

To add a host to the hosts file, use the SET HOST command, whose syntax is as follows:

```
SET HOST hostname/ADDRESS=internet_addr/ALIAS=alias
```

where

internet.addr	Is the Internet address of a host computer.
hostname	Is the name of the host.
alias	Is an alternate name for the host. You can have as many as two aliases for any one host.

For example, a host named BOSTONSALES might be added using the following command:

```
UCX> SET HOST BOSTONSALES/ADDRESS=89.0.1.100/ALIAS=BOSTON
```

where

89.0.1.100	Is the Internet address of the remote host.
BOSTONSALES	Isa name for the remote host.
BOSTON	Is an alias setup for the BOSTONSALES host.

If your HOSTS file is incomplete, edit the file to include additional host names and their Internet addresses.

Setting the Default Host (Clients Only)

You can configure a client system to specify a default host to connect to when using SQL*Net TCP/IP. To designate a default host, modify the HOSTS file on the client computer by adding the alias “SQLNET" for the default host.

For **Wollongong** implementations, the HOSTS file is located in:

```
TWG$TCP:[NETDIST.ETC]HOSTS
```

For **Multinet** implementations, the HOSTS and services files are combined in one file called:

```
MULTINET:HOSTS.LOCAL
```

For example, a host named “BOSTONSALES” might be referenced like this in the HOSTS file:

```
89.0.1.100    BOSTONSALES   SQLNET
```

where

89.0.1.100	Is the Internet address of the remote host.
BOSTONSALES	Is a name for the remote host.
SQLNET	Is an alias setup for the BOSTONSALES host. The SQLNET alias identifies BOSTONSALES (or any host name it is associated with) as the default host. That is, if no host name is specified at logon time, TCP/IP will search for the host name corresponding to the SQLNET alias.

For example, if a user specified

```
    $ SQLPLUS  SCOTT/TIGER@T:
```

then the default database on the default remote machine BOSTONSALES would be used.

Note: Alternatively, OpenVMS clients can specify a default database by defining the ORA_DFLT_HOSTSTR logical name as the default connect string they wish to use. Also, you do not need to use the same default host on all systems on the network. Each client can have its own default SQL*Net TCP/IP host.

OpenVMS Connect String

Any ORACLE application that accepts a username and password on the command line can use SQL*Net TCP/IP to connect to a remote database.

The general syntax for running an application program from the command line is:

```
application  username/password@connect  string
```

The syntax for the connect string is:

```
 T: [hostname:system  ID:  buf_size,  retry_count,  keepalive]
```

where

T	Is the net prefix for the TCP/IP protocol.
hostname	Is the name of the remote system that is the target of your connection request.
system ID	Is the SID of the database you want to access.
buf_size	Is the maximum buffer size for transferring SQL*Net messages. The default is 4096.

retry_count	Is the number of times that the SQL*Net driver should retry to connect if the first attempt fails. The default is zero (0). This option is useful in situations where the TCPSRV is overloaded and slow in answering requests.
keepalive	Is the KEEPALIVE option. Use this parameter to check whether a node is working. See “Configuring the KEEPALIVE Option” earlier in this chapter for more information.

CHAPTER

5

SQL*NET TCP/IP FOR DG AOS/VS

This chapter explains how to configure and use SQL*Net TCP / IP for the Data General AOS / VS operating system.

Before you can use SQL*Net with the TCP / IP protocol, you must have installed and configured the appropriate ORACLE and Data General software on your system. Installation information is contained in the *SQL*Net for DG AOS/VS Installation and User's Guide* which also lists other useful DG documents and provides information not covered in this chapter.

About TCP/IP on DG AOS/VS Systems

TCP/IP protocol for DG AOS/VS is available in two versions: host-based and socket-based. ORANET polls the operating system to determine which driver is installed, then uses that driver.

Socket-based vs. Host-based Implementations

Any AOS/VS II system running TCP/IP II has the capability to support TCP/IP sockets. The socket-based implementation of TCP/IP differs from the host-based implementation in the way packets are passed between the network software and the Oracle software. TCP/IP II moves much of the control of TCP connections into the AOS/VS II kernel where it is more quickly accessed than when using the IPC send/receive mechanism used by host-based TCP/IP. Another advantage of locating control in the AOS/VS kernel is that the maximum number of SQL*Net users increases.

In general, socket-based TCP/IP requires less system overhead than host-based TCP/IP. Use socket-based TCP/IP wherever possible.

Connection Features

The service name on a TCP/IP network is ORASRV. ORASRV must be defined in the :ETC:SERVICES system file or as a network process name (NPN) file in the :NET directory.

When ORANET starts, the local TCP service name contacts the TCP agent on the remote host. If the remote TCP service name does not recognize the name ORASRV, the request is denied.

Transport Service

The TCP/IP Transport Service establishes and maintains a connection between various hosts using TCP/IP.

Configuration Tasks for DG AOS/VS

The following sections describe what tasks you need to complete to use SQL*Net TCP/IP in a DG AOS/VS networking environment.

TCP/IP System Test

TCP/IP must be working correctly before you can configure SQL*Net, To verify that the TCP/IP software is working properly, use TELNET to logon to another system in the TCP/IP network by typing the following command:

```
X TELNET hostname
```

where *hostname* matches the name of an entry in the :ETC:HOSTS file. If you successfully log onto another node, your hardware and software are working and you can use SQL*Net TCP/IP in a networking environment.

Identifying the Available Hosts (Clients Only)

You must identify both server nodes and the databases available at these nodes in a file named CONFIG.ORA A copy of this CONFIG.ORA file must reside in the directory from which you plan to access other nodes in your TCP/IP network, or you must add the directory path to CONFIG.ORA onto the searchlist of each SQL*Net TCP/IP user.

Note: You can add comments to a CONFIG.ORA file by starting the line with REM. Any line in a CONFIG.ORA file can only be 255 characters long. If a line is longer than 80 characters, place an "&" followed by a Newline at the end of the line to continue it.

If you want to identify servers and databases in the CONFIG.ORA file, add a separate entry for each available database at each available server. These entries must use the following syntax:

```
SQLNET DBNAME alias.T:hostname:system ID
```

where

SQLNET	Is the SQL*Net keyword.
DBNAME	Isa required, literal keyword.
alias	Is an alias for a specific database. Users can substitute this alias for the full database specification in the SQL*Net commands they use to connect to this database.

The connect string consists of the following three parameters in the CONFIG.ORA entry:

```
T: hostname:system ID [,buffer_size] [/service_number]
```

where

T	Is the SQL*Net driver prefix "T" for the TCP/IP protocol.
hostname	Is the same as the name in the HST file in your :NET directory (for the node on which the specified database resides), or the reference in the :ETC:HOSTS file. If the HST filename includes a "$", do not include this symbol as part of the hostname.
system ID	Is an up to six uppercase character system ID (SID) that uniquely identifies this database on the node on which it resides. This character string is the same as the one in the ORACLE.SID file associated with the database.
buffer_size	Defines the number of bytes transmitted across the network with each information packet. The default buffer size is 1024.
service_number	When used as a client, the connect string can specify a destination address (other than the ORASRV 1525 default service number), such as a port number on an IBM VM host.

In the following example, a database with the SID "P" and the alias "PROD" is identified as available to the local system at the server node BOSTON.

```
SQLNET DBNAME PROD = T:BOSTON:P
```

Identifying the Available Databases (Servers Only)

Server nodes can contain more than one database to which client nodes can connect. A server node's CONFIG.ORA file keeps track of these databases and their locations. Any databases that are to be accessible to other nodes must be identified in this file.

To identify each database, add a separate entry for each database to CONFIG.ORA using the following syntax:

```
SQLNET SIDMAP system ID = :directory path
```

where

SQLNET	Is the SQL*Net keyword.
SIDMAP	Is a required, literal keyword.

system ID	Is up to six uppercase characters that uniquely identify this database on the node on which it resides. This character string is the same as the one found in the ORACLE.SID file associated with the database.
directory path	Is the path of the directory in which the database files (including the SGA file and ORACLE.SID) reside.

For example:

```
SQLNET SIDMAP A = :ORACLE:  PRODUCTION
SQLNET SIDMAP T = :ORACLE : TEST
```

Identifying the Default Database

To establish a default database on your system, identify that database in your CONFIG.ORA file using the keyword SIDMAPDFLT and the database's SID:

```
SQLNET SIDMAPDFLT = A
```

Identifying Special Database Searchlists

On a system that supports multiple ORACLE databases, the server process may need to use a different searchlist when running different databases.

It may also be necessary to use a different searchlist when SIDMAP points to a directory that does not contain the ORACLE executable files, such as ORANETSRV.PR or ORAKERNEL.PR. In this case, you must include the ORACLE executable in the searchlist.

To identify the searchlist to be used with a particular database, you can add an entry to CONFIG.ORA that uses the following syntax:

```
SQLNET SIDSEA system ID = searchlist
```

where

SQLNET	Is the SQL*Net keyword.
SIDSEA	Is the required, literal keyword.
system ID	Is up to six uppercase characters that uniquely identify this database on the node on which it resides. This character siring is the same as the one in the ORACLE.SID file associated with the database.
searchlist	Is the searchlist to be used when accessing the ORACLE executable.

The following is an example of a CONFIG.ORA file:

```
REM Available Server Nodes Follow
SQLNET DBNAME PROD= T:BOSTON:P
SQLNET DBNAME TEST=T:DALLAS:T
SQLNET SIDMAP A =  :ORACLE:PRODUCTION
SQLNET SIDMAP T = :ORACLE:TEST
SQLNET SIDMAPDFLT = A
SQLNET SIDSEA A = :ORACLE
```

The TCP/IP Listener Process

In a TCP/IP network, an ORACLE application and an ORACLE RDBMS communicate via the ORACLE TCP/IP agent named ORANET.

The ORANET agent provides TCP/IP access to ORACLE databases on a system. ORANET is also called the listener process because it introduces itself to TCP/IP as the agent which 'listens' to requests for a service called ORASRV.

When the TCP agent receives a connection request, it directs it to ORANET. ORANET "requests" the system ID of the remote database, "consults" the CONFIG.ORA file to determine that database's location (as well as the searchlist associated with the database), and then starts a server process to handle the actual database transactions.

Any ORACLE application, such as SQLPLUS, can direct a request to a database on a remote host using TCP/IP. The user application first requests its local TCP agent to connect it with ORASRV on the remote host.

The local TCP agent contacts the TCP agent on the remote host, informing it of the request from SQLPLUS. If the remote TCP agent does not recognize the name ORASRV, the request is denied. For example, this can occur if the NPN file ORASRV has been improperly NETGENed or if ORASRV is missing from the :ETC:SERVICES file.

If the remote TCP agent recognizes ORASRV, it creates a virtual circuit and passes the connection to ORANET. ORANET then establishes a server process in the appropriate database directory and passes the connection to this server. Once connected, the user application interacts directly with the remote server, bypassing ORANET entirely.

A server process executes the request and sends the results back to the user process, which directs these results to the user application. For more information concerning ORANET, refer to the *SQL*Net for DG AOS/VS Installation and User's Guide.*

Defining an NPN File for the Listener Process (Servers Only)

On server systems, you must start the listener process to handle connection requests from SQL*Net client systems. The listener process performs the following tasks:

- listens for client connection requests
- determines the database to which the client wants to connect
- creates an ORACLE database server process in the proper directory
- sets up communication between the client and the database process
- waits for the server process to exit

Before you can start the listener process, you must first identify the process in your network configuration by defining a Network Process Name (NPN) file named "ORASRV".

To define the ORASRV file, run the NETGEN utility. Enter the appropriate information for your system, but when the following message appears:

```
Network Process Name (0-4 bytes) : ORAS
```

do not accept the default. Backspace over "ORAS" and enter "1525", the established Internet service number for ORACLE.

NETGEN creates an NPN file entry, ORASRV, in your :NET directory. You must complete the configuration steps in the NETGEN utility for each system that will act as a server in the TCP/IP network. After each system has been reconfigured, the TCP/IP network processes must be shut down and restarted.

Setting up the ORASRV Service (Servers Only)

The NPN file on which ORANET listens for TCP/IP connections is called ORASRV. Users familiar with SQL*Net Asynchronous should be careful not to confuse the NPN of ORASRV with the ORASRV.PR file, which is used to execute ORACLE on the host machine for an async connection.

Several tasks must be performed to set up a system as a SQL*Net TCP/IP server:

- modify the :ETC:SERVICES file (for socket-based TCP/Ii?)
- apply Data General's TCPIP.PR patch (host-based TCP/IP)
- check :ETC:HOSTS file for remote systems
- create file ORACLE_USE_TCPKP2 (for socket-based TCP/IP)

Modifying :ETC:SERVICES

To support the ORANET server, modify the file :ETC:SERVICES by appending the supplied file SERVICES_ORACLE.PROTO from the ORACLE release directory to the SERVICES file created when TCP/IP was installed. Append this file by logging on as superuser and entering the following command:

```
) COPY/A :ETC:SERVICES SERVICES_ORACLE.PROTO
```

Remember to bring down TCP/IP and restart it so that the new :ETC:SERVICES entry will be recognized.

Applying Patch TCPIP.PR

If you use DG's Version 2.50 release of TCP/IP, a patch file must be applied to the TCPIP.PR file before SQL*Net TCP/IP will operate properly. The patch file is 2.50_VS_TCPIP_PATCH. This bug has been filed as STR#NASC-701-0 with Data General.

Instructions for applying the patch are contained in comments in the patch file itself. Remember to save a copy of your current TCPIP.PR file before applying the patch. You must shut down TCP/IP before attempting to patch TCPIP.PR.

Warning: The patch file will not work properly with versions of TCP/IP earlier than V2.50. If you have an earlier release of TCP/IP, contact Data General to upgrade to the current level.

Checking :ETC:HOSTS

Check that the :ETC:HOSTS file includes the remote host names to be accessed by TCP/IP. This step is included in the installation process for DG TCP/IP. Refer to your DG documentation for details on using the :ETC:HOSTS file on MV systems.

Creating ORACLE_USE_TCPIP2

This procedure is only necessary if you are using socket-based TCP/IP. Follow these steps to create ORACLE_USE_TCPIP2:

1. Create ORACLE_USE_TCPIP2, an ORACLE logical file of file type 200, using the LOGICAL.CLI macro. Place ORACLE_USE_TCPIP2 on the searchlists of the ORANET process and all SQL*Net TCP/IP sockets users.

   The contents of ORACLE_USE_TCPIP2 are not important. When a user enters the connect string for the TCP/IP protocol, ORANET selects the socket-based TCP/IP interface if ORACLE_USE_TCPIP2 exists on the searchlist and TCP/IP II is installed on the MV system.

   Both socket-based and host-based TCP/IP can be used on the same system at the same time for SQL*Net client requests. However, if ORANET on the server has ORACLE_USE_TCPIP2 on its searchlist during startup, ORANET uses socket-based TCP/IP for all incoming requests.

Refer to the first section of this chapter for more information about socket-based TCP/IP. Refer to the *SQL*Net for DG AOS/VS Installation and User's Guide* for details on creating ORACLE logical files.

2. If you use DG's TCP/IP II prior to Version 1.20, you must apply patch file DG_CANCEL_TIMER_PATCH to your system .PR file before using SQL*Net TCP/IP II. Save a copy of your current system .PR file before applying the patch.

   Instructions for applying the patch are included in the comment section in the patch file.

Starting the Listener Process

Before SQL*Net clients can connect to the server, the DBA (or anyone with the necessary authority) must start the listener process on the host system. (The listener is usually started automatically by a network UP.CLI macro when ORACLE is started.)

The listener process requires superuser privilege to create its communication files in the :PER directory. It should normally be started up by the OP process, which has this privilege.

To start the listener process, log into the ORACLE account and run the ORANET macro by typing:

```
) ORANET START
```

ORANET runs in the :PER directory and creates a shared file there for communication with its server processes. To gain write access to the :PER directory, ORANET turns on superuser mode. The DBA or system manager who starts ORANET must have superuser privilege.

Listing Process Status

When the listener process has started, you can check the status of the process by typing

```
)   ORANET STATUS
```

which will produce a display similar to the following example:

```
FROM ORANET:
          Current Status:
Version          1.0.2.4
Active listeners    5
Available  servers  0
Active servers      0

FROM ORANET:
          Current Settings:
MIN_SERVERS      2
```

```
MAX_SERVERS       5
SERVER_WAIT       60 sees.
CONFIG_FILE        :ORACLE:CONFIG.ORA
LOG_PATHNAME       :ORACLE:?ORANET.TMP
```

where

Active listeners	Is the total number of listener tasks enabled.
Available servers	Is the total number of servers available to ‘answer’ a request for connection.
Active servers	Is the total number of servers actually connected to remote users.

The settings display the current values of several configuration parameters. For explanations of MIN.SERVERS, MAX_SERVERS, SERVER_WAIT, CONFIG_FILE, and LOG_PATHNAME, refer to the *SQL*Net for DG AOS/VS Installation and User’s Guide.*

By default, the listener process logs process information such as connection requests in a file named ?ORANET.TMP, which is created in the directory from which you issued the ORANET START command. Error messages from server processes are also logged in ?ORANET.TMP. If you have trouble starting a server, examine the contents of this file.

Disabling Logging

You can disable logging by typing:

```
)   ORANET LOGGING DISABLE
```

Note that since AOS/VS buffers up to 2048 bytes of the logfile before flushing it to disk, the current connection status may not be shown in the log file.

Stopping the Listener Process

To stop the listener process, type one of the following commands:

```
)   ORANET HALT
```

or

```
)   ORANET SHUT
```

If no database connections are active when you issue this command, the listener process will terminate. If any active connections exist, this message displays:

```
FROM ORANET:  HALT in progress:  <n> servers still active
```

where <n> is the number of connections.

Killing the Listener Process

If you need to stop the listener process immediately, type:

```
)   ORANET KILL
```

If you kill the listener process, any active connection will be broken and any ORACLE transaction in progress will be rolled back. For more information about ORANET-related commands and functionality, refer to the *SQL*Net for DG AOS/VS Installation and User’s Guide.*

CHAPTER

6 SQL*NET TCP/IP FOR VM

The SQL*Net TCP/IP protocol makes it possible for a VM machine to communicate with different computing environments. This chapter describes how to modify VM clients that use SQL*Net TCP/IP and how to prepare masters and servers that are part of your network. It also explains the syntax of the connect siring.

For all other details about installing and configuring SQL*Net TCP/IP for VM, consult the *ORACLE for VM User's Guide,* the ORACLE for *VM Installation and System Administrator's Guide* and the *ORACLE7 for VM SQL*Net Configuration and User's Guide.*

Configuration Tasks

The next sections describe the tasks that you must complete in addition to the configuration tasks that are necessary to use a VM machine and SQL*Net TCP/IP.

Modifying the Client Virtual Machine

If the host name is not specified in the "dotted decimal" format, a host name lookup is required. The HOSTS SITEINFO file provides the hosts' names. Contact your DBA if you have questions about specifying host names. For more information about the HOSTS SITEINFO file, consult the IBM TCP/IP documentation.

A client must add the LINK and ACCESS commands to the PROFILE EXEC file before it can perform a lookup in the HOSTS SITEINFO file and gain access to ORACLE through SQL*Net TCP/IP. The following is a PROFILE EXEC file example:

```
LINK TCPMAINT 592 199 RR
ACCESS 199 H
```

Placing the lines in the previous example in the client machine's PROFILE EXEC links the TCPMAINT 592 disk as the ORACLE userid's 199 (H) disk.

Preparing to use TCP/IP for Masters and Servers

Before a master or server machine can invoke the TCP/IP protocol, the IBM TCP/IP communications virtual machine must be running. If you need to stop or restart the IBM TCP/IP communications machine, use the STOP TCPIBM and START TCPIBM coremands to reinitialize the Network Master machine's use of TCP/IP. It is not necessary to restart the database.

To prepare the Network Master machine for the TCP/IP protocol, issue the START TCPIBM command from either the DBA userid (using the SMSG command) or from the ORAMAST file. The format of the ORAMAST START command is:

```
START TCPIbm router socket
```

For more information about the START command, refer to the *ORACLE7 for VM SQL*Net Configuration and User's Guide.*

SQL* Net TCP/IP for VM Connect String

The connect string that the client uses to access the target database must be appropriate for the hardware, operating system, and ORACLE database that is to be the target of the SQL*Net connection. When the client is on VM, then the connect string format can be found in the ORACLE documentation for the target operating system. When the database (server) is on VM, then the connect string must have the following format:

```
driver_prefix: hostname[/tcpip_portnumber] : [sid] [,bufsize]
    [,acctname] [,acctpass] [,servgrp]
```

where

driver_prefix	Indicates the type of TCP/IP protocol to be used: either T for IBM TCP/IP or K for KNET TCP/IP.
hostname	Specifies the name of an entry in the TCP/IP host file in the client's TCP/IP system. This entry must be specified in one of the following ways: • As the fully qualified hostname and any nickname. • As Internet address in decimal (e.g. 126.0.3.73)
/tcpip_ portnumber	Specifies the port number on which the remote database is listening for connection requests. On VM, the port number is specified in the ORAMAST file of the instance. *The tcpip_portnumber* parameter is optional. If it is not specified, it defaults to 1525.
sid	Is the name of the ORACLE database system to which you are connecting. This parameter is ignored when the target database is on a VM system using TCP/IP software; for compatibility purposes, its position is reserved.
bufsize	Specifies the TCP/IP buffer size in bytes. This number must be in the range of 5 to 4096. The default size is 4096.
acctname	Specifies the VM account to be charged for the VM resources used by this client. The same name must be defined with a Network Master machine (ORAMAST) ACCOUNT command. If the client is on an operating system that can supply the client's userid for accounting purposes, then the client's userid is used as the default value for this

	parameter. If the client's userid cannot be provided by the operating system, this parameter must be specified. If the acctname parameter is specified explicitly, it overrides any name supplied by the client's operating system. If you have any questions about specifying the VM account to be charged for resources, contact your DBA.
acctpass	Specifies the password associated with the account to be charged for resources used during the connection. The acctpass must match the password defined in the Network Master machine's ACCOUNT command for the acctname. If <NOAUTH> is specified in the ACCOUNT command, this parameter may be omitted. Otherwise, the password must be specified.
servgrp	Specifies the name of the server group to be used for the requested connection, This parameter is optional.

If an optional parameter is to be omitted, a comma must be used to indicate the missing parameter.

The only parameter that may be case sensitive is the hostname. Depending on the client's operating system, the hostname field mayor may not be case sensitive. Consult the documatation for the communications package you are using to determine if the hostname on the client system is case sensitive.

Sample Connect Strings

The following examples show TCP/IP connect strings that might be used in the VM environment.

Example 1: Any Operating System Client Accessing a VM Server

```
T:SYSBOS/1677: , , IPUSER,IPPASS,XYZ
```

where

T:	Specifies the IBM TCP/IP protocol.
SYSBOS	Is the hostname.
1677	Is the port number.
”	Replace the missing *sid* and *bufsize* parameters.
IPUSER	Is the userid to be charged for the resources used.
IPPASS	Is the password for IPUSER.
XYZ	Is the server group to be used.

Example 2: A VM Client Accessing a VM Server

```
T:HQCMS/1677: , , SCOTT, TIGER
```

where

T:	Specifies the IBM TCP/IP protocol.
HQCMS	Is the hostname.
1677	Is the port number.
”	Replace the missing *sid* and *bufsize* parameters.
SCOTT	Is the userid to be charged for the resources used.
TIGER	Is the password for SCOTT.

Example 3: A VM Client Accessing an MVS Server

```
T:HQMVS:ORAMVS
```

where

T:	Specifies the IBM TCP/IP protocol.
HQMVS	Is the hostname using the default port number 1525
ORAMVS	Is the sid name of the ORACLE database on MVS

Note: Macintosh cannot connect with VM as a client.

CHAPTER

7 SQL*NET TCP/IP FOR MVS

This chapter provides an explanation of connect string sytax for MVS clients and servers that use SQL*Net TCP/IP. We assume that SQL*Net TCP/IP is already installed on your system. For MVS hardware and software requirements and configuration information, refer to this documentation:

- *SQL*Net for MVS System Release Bulletin*
- *SQL*Net for MVS Installation and User's Guide*
- *ORACLE7 for MVS Installation Guide*
- *ORACLE7 for MVS System Administration Guide*

Connecting to an MVS Server

When connecting to an MVS ORACLE subsystem from another TCP/IP node, use the following connect string:

```
T: ip_address:ssn
```

where

ip_address	Is the IP address of the MVS ORACLE subsystem or the name of the MVS TCP/IP node.
SSN	Is the MVS ORACLE subsystem name,

For example, to connect to the MVS ORACLE subsystem NYOR with IP address 80.110.4.78, the following connect string would be used:

```
SCOTT/TIGER@T:80.110.4.78:NYOR
```

Connecting as an MVS Client

The following connect string is used by MVS clients to connect to a remote SQL*Net TCP/IP server from an MVS ORACLE subsystem:

```
@T: ip_address[/port]:sid  [,bufsize] [,acctuser, acctpw,
    servgrp] : [:SMSG] [:GTRACE] [:VERSION; [:TRACE]
```

where

ip_address	Is the IP address or nickname of the host to which you are connecting. The nickname must be defined to your TCP/IP software. The format of the IP address is nnn.nnn.nnn.nnnn where nnn is a decimal integer from 1 to 255. There is no default.
port	This is the TCP port number on which the SQL*Net TCP/IP server is listening. The value maybe from 1000 to 65534. The default is 1525.
sid	Is the name of the ORACLE database system to which you are connecting. If the database is on a VM or MVS systme using the TCP/IP IBM software, this parameter is not required since, in this case, the port number determines which database is selected.
bufsize	This is the buffer size to be used for the SQL*Net TCP/IP connection. The value maybe from 5 to 4096. Oracle Corporation does not recommend changing the default value unless the hardware being used requires a specific value. The default is 4096.
acctuser	This is the accounting userid for connecting to VM servers. There is no default.
acctpw	This is the accounting password associated with *acct* user. There is no default.

servgrp	This is the server group name for connecting to VM servers. There is no default.
SMSG	This parameter requests that session status messages be issued for the connection. Two status messages are issued: *TCP connection open* and *TCP connection closed.* The default is SMSG off.
GTRACE	This parameter requests that GTF tracing be performed by SQL*Net. This is for debugging purposes and should only be specified if requested by Oracle Worldwide Support. The default is GTRACE off. **Note:** This parameter is *not* available for SQL*Net TCP/IP TCPaccess.
VERSION	This parameter requests that a message be issued identifying the SQL*Net driver version number. This is for debugging purposes and should only be specified if requested by Oracle Worldwide Support. The default is VERSION off.
TRACE	This parameter requests that debugging messages be written by the SQL*Net driver. This is for debugging purposes and should only be specified if requested by Oracle Worldwide Support. The default is TRACE off. **Note:** This parameter is only available for SQL*Net TCP/IP TCPaccess.

For example, to connect to an ORACLE database named VAXVMS with an IP address of 80.110.4.78, you would use the following connect string:

```
SCOTT/TIGER@T:80.110.4.78:VAXVMS
```

APPENDIX

A

SQL*NET TCP/IP ERROR MESSAGES

The following section lists the error messages you may receive while using SQL*Net TCP/IP. The error messages have been grouped by TCP/IP-specific messages (6100 through 6121), SQL*Net error messages (6402 through 6417) and SQL*Net TCP/IP TLI error messages (6700 to 6793). Sections describing UNIX and VAX OpenVMS error messages are also included.

If you receive an error message, check to see if it might be due to one of the following general situations:

- The host system may not be defined in your HOSTS file (client systems only). Check your HOSTS file to make sure the host system is defined properly.
- The host system may be down. Check with your system administrator.
- TCP/IP maybe down. Check your communications software and hardware by using the command `telnet hostname`. Otherwise, check the *Installation Guides* for your system.
- The remote ORACLE database or the ORACLE server maybe down. Check with your system administrator or DBA.
- The specific database on the remote server may not be defined or available (server systems only). Check "Identify the Available Databases" for your specific system. Otherwise, contact your system administrator or DBA.

TCP/IP Errors (6100-6121)

ORA-6100: incorrect message type from host

Cause: The SQL*Net TCP/IP driver received a message with an unrecognizable message type.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6101: incorrect number of bytes written

Cause: The SQL*Net TCP/IP driver sent a message that was apparently successful, but the number of bytes transmitted did not match the number of bytes supplied to the driver.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6102 cannot allocate context area

Cause: The SQL*Net TCP/IP driver could not allocate heap space for the context area.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6103: send out-of-band message failed

Cause: The SQL*Net TCP/IP driver failed to send an out-of-band break message across the connection.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6104: cannot setup process group for socket

Cause: SQL*Net TCP/IP driver failed to bind the communications channel to client/server process for out-of- band break handling.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6105: cannot connect to remote host -- remote node is unknown

Cause: SQL*Net TCP/IP driver could not find your remote host information.

Action: Make sure you specified the hostname correctly on the command line. (Also, check your spelling and capitalization.) If the remote host is undefined in the HOSTS file, ask the system administrator on the host machine to add an entry for the new host.

ORA-6106: socket creation failed

Cause: SQL*Net TCP/IP driver failed to create a communication channel for data transfers. The system file table is probably full.

Action: Read the returned operating system error code and contact your system administrator.

ORA-6107: oracle network server not found

Cause: ORASRV is missing from the SERVICES file.

Action: Ask your system administrator to define ORASRV in the SERVICES file.

ORA-6108: connect to host failed

Cause: SQL*Net TCP/IP driver failed to establish connection to the host machine. The ORASRV process on the host machine has not been brought up, or the Ethernet line is down,

Action: Contact your system administrator.

ORA-6109: message receive failure

Cause: SQL*Net TCP/IP driver failed to receive a message from the communication channel.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative. On VAX/VMS, use TCPCTL START to start the server.

ORA-6110 message send failure

Cause: SQL*Net TCP/IP driver failed to send a message across the communication channel.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6111: cannot setup break handler

Cause: SQL*Net TCP/IP driver could not setup out-of-band break handler.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6112: error on bind

Cause: ORASRV failed to assign a name to the communication channel.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

OIU-6113: error on accept

Cause: ORASRV failed to accept a connection request from the client.

Action: Check the ORACLE error message and the operating system error message and then contact your customer support representative.

ORA-6114: SID lookup failure

Cause: The database SID supplied in the connect string was not recognized by the remote host.

Action: Ask your system administrator to add the appropriate SID entry to the CONFIG.ORA file on the remote host. On VAX/VMS, make sure you enclose the connect string system ID in quotes ("Q").

ORA-6115: unable to create environment for ORASRV

Cause: You connected to the remote host, but connection was lost before the ORACLE server image could be started.

Action: Contact your system administrator.

ORA-6116: unable to create ORASRV process

Cause: The remote host was unable to create or start the ORACLE server image.

Action: Read the error message returned by the operating system with this ORACLE error and contact your system administrator.

ORA-6117: unable to get socket device name for ORASRV

Cause: The port number may be undefined in the SERVICES file.

Action: Make sure the SERVICES file contains the correct port number. Otherwise, contact your system administrator.

ORA-6118: unable to complete handshake with ORASRV

Cause: The remote ORACLE server image was started but could not be properly initialized.

Action: Read the error message returned by the operating system with this ORACLE error, and contact your system administrator.

ORA-6119: bad connection request format

Cause: The remote host was unable to process the connection request.

Action: Read the error message returned by the operating system with this ORACLE error and contact your system administrator.

ORA-6120: network driver not loaded

Cause: The TCP/IP network driver is not loaded correctly.

Action: Check that the TCP/IP driver is loaded correctly.

ORA-6121: access to ORASRV denied

Cause: The remote server does not recognize your *orasrv* access rights.

Action: Read the error message returned by the operating system with this ORACLE error and contact your system administrator.

ORA-6122 NETTCP: setup failure

Cause: The host's SQL*Net TCP/IP server is unable to set up the appropriate environment to service this connection request.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6123: NETTCP cannot set KEEPALIVE

Cause: The host's SQL*Net TCP/IP server is unable to set the socket KEEPALIVE option.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6124: NETTCP: timeout waiting for ORASRV

Cause: The ORACLE server process was started but failed to respond after N seconds.

Action: For heavily loaded systems this is not an uncommon occurrence. Increase number of seconds (the default is 30) in the CONFIG.ORA file statement:

```
SQLNET ORASRV_WAIT=<number of seconds>
```

The new value will come into effect the next time the SQL*Net TCP/IP server is started.

ORA-6125: NETTCP: ORASRV exited unexpectedly

Cause: The ORACLE server process was started but exited unexpectedly. Possible causes are:

1. Insufficient quotas to run ORASRV
2. ORACLE is no installed.

Action: See the ORASRV output file for more details; the file is in the ORA_SQLNET directory and has a name in the form ORA_SRVTnn_<SID>.OUT. If the appropriate action is not obvious from the ORASRV output, then contact your customer support representative.

ORA-6126: NETTCP: ORASRV unable to open network connection

Cause: The ORACLE server process was started but was unable to open the socket passed to it by TCPSRV.

Action: Contact your customer support representative.

ORA-6127: NETTCP unable to change username

Cause: The host's SQL*Net TCP/IP server could not establish a PROXY LOGIN connection because the client username is unknown (to the host OS).

Action: Create a new user account on the host.

ORA-6128: NETTCP: unable to create mailbox

Cause: The host's SQL*Net TCP/IP server was unable to create a mailbox (needed for IPC communication with the ORACLE server process).

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6129: NETTCP: unable to transfer socket ownership to ORASRV

Cause: The host's SQL*Net TCP/IP server was unable to transfer the network communication handle to the ORACLE server process.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6130: NETTCP: host access denied

Cause: The host's SQL*Net TCP/IP server rejected this connection request because the client node does not have access privilege as determined by the contents of the Valid Node Table (VNT), a component of the host's CONFIG.ORA file.

Action: To grant access, add appropriate entry to the host's VNT.

ORA-6131: NETTCP: user access denied

Cause: The host's SQL*Net TCP/IP server rejected this connection request because the client username does not have access privilege as determined by the contents of the Username Mapping Table (UNIT), a component of the host's CONFIG.ORA file.

Action: To grant access, add appropriate entry to the host's UMT.

ORA-6132: NETTCP: access denied, wrong password

Cause: The host's SQL*Net TCP/IP server rejected this connection request because the client password did not match the host password.

Action: To grant access, get the passwords in sync.

ORA-6133: NETTCP: file not found

Cause: The host's SQL*Net TCP/IP server could not find the SID mapping file (specified in CONFIG.ORA) associated with this connection request.

Action: Check the CONFIG.ORA file for spelling errors and make corrections as necessary.

ORA-6134: NETTCP: file access privilege violation

Cause: The host's SQL*Net TCP/IP server did not have READ/EXECUTE permission for the SID mapping file (specified in CONFIG.ORA) associated with this connection request.

Action: Change the protection on the SID mapping file.

ORA-6135: NETTCP: connection rejected; server is stopping

Cause: The host's SQL*Net TCP/IP server rejected this connection request because it is in the process of stopping.

Action: Restart the SQL*Net TCP/IP server.

ORA-6136: NETTCP: error during connection handshake

Cause: A network failure occurred while communicating with the host's SQL*Net TCP/IP server.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6137: NETTCP: error during connection handshake

Cause: A network failure occurred while communicating with the host's SQL*Net TCP/IP server.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6138: NETTCP: error during connection handshake

Cause: A network failure occurred while communicating with the host's SQL*Net TCP/IP server.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6139: Reserved.

ORA-6140: NETTCP: no such user

Cause: A proxy login connect attempt failed because the client username has no counterpart on the host.

Action: Verify the proxy login.

ORA-6141: NETTCP: no privilege for user

Cause: A proxy login connect attempt failed because the SQL*Net TCP/IP server has insufficient privileges to access the proxy account.

Action: Change the account protection and the server privileges.

ORA-6142: NETTCP: error getting user information

Cause: A proxy login connect attempt failed because the SQL*Net TCP/IP server was unable to access the proxy account.

Action: See the SQL*Net TCP/IP server log file for more details and then contact your customer support representative.

ORA-6143: NETTCP: maximum connections exceeded

Cause: A connect attempt failed because the maximum concurrent connections supported by the host's SQL*Net TCP/IP server has already been reached.

Action: Wait for a minute or two and then retry.

OM-6144: NETTCP: SID (database) is unavailable

Cause: The database administrator on the host has varied the SID offline.

Action Wait for the SID to be varied online.

ORA-6145: NETTCP: unable to start ORASRV: images not installed

Cause: The host's SQL*Net TCP/IP server was unable to start the ORACLE server process because the ORACLE protected images were not installed.

Action: Install the ORACLE protected images.

ORA-6146-6149: Reserved.

SQL*Net Errors (6402-6417)

ORA-6402: unexpected end-of-file

Cause: The SQL*Net driver detected that its partner has unexpectedly exited.

Action: Contact your customer support representative.

ORA-6403: cannot allocate context area

Cause: The SQL*Net driver could not allocate heap space for the context area.

Action Contact your customer support representative.

ORA-6404: bad database ID string

Cause: The SQL*Net driver detected a syntax error in the database ID portion of the user's logon string.

Action: Check the logon string and then correct the syntax error. Refer to Chapter 2 for more information on using SQL*Net parameters.

ORA-6405: break/reset protocol error

Cause: The SQL*Net driver detected an error while trying to reset the connection from a break state.

Action Contact your customer support representative.

ORA-6406: error servicing break msg interrupt

Cause: The SQL*Net driver could not correctly handle an incoming out-of-band message.

Action: Contact your customer support representative.

ORA-6407: unable to setup break handling environment

Cause: The SQL*Net driver could not setup out-of-band break handler.

Action: Contact your customer support representative.

ORA-6408: bad message type

Cause: The SQL*Net driver received a message having an unrecognizable message type.

Action Contact your customer support representative.

ORA-6409: network write error

Cause: The SQL*Net driver failed to send a message across the communication channel.

Action: Contact your customer support representative.

ORA-6410: network read error

Cause: The SQL*Net driver failed to receive a message from the communication channel.

Action: Contact your customer support representative.

ORA-6411: bad write length

Cause: The SQL*Net driver failed to send a complete message across the communication channel.

Action: Contact your customer support representative.

ORA-6412: bad read length

Cause: The SQL*Net driver failed to receive a complete message from the communication channel.

Action: Contact your customer support representative.

ORA-6413: unexpected end-of-file

Cause: The SQL*Net driver was unable to establish a connection with a remote node.

Action: Contact your customer support representative.

ORA-6414 network close error

Cause: The SQL*Net driver was unable to disconnect the connection cleanly.

Action: Contact your customer support representative.

ORA-6415: driver version numbers are incompatible

Cause: The SQL*Net driver detected that its partner entity was running at a software revision level incompatible with its own.

Action: Contact your customer support representative.

ORA-6416: error on test operation

Cause: The SQL*Net driver detected an error when checking the I/O status of the connection.

Action: Contact your customer support representative.

ORA-6417: unknown target database

Cause: The SQL*Net driver detected on the remote host that the selected database does not exist.

Action: Resubmit the string with the correct database ID.

SQL*Net TCP/IP TLI Error Messages (6700-6793)

ORA-6700: incorrect message type from host

Cause: SQL*Net received a message with an unrecognizable message type.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6701: incorrect number of bytes written

Cause: SQL*Net sent a message that was apparently successful, but the number of bytes transmitted did not match the number of bytes supplied to the driver.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6702: cannot allocate context area

Cause: SQL*Net could not allocate heap space for the context area.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6703: send break message failed

Cause: SQL*Net failed to send a break message across the connection.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6704: receive break message failed

Cause: SQL*Net failed to receive an expected break message.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6705: remote node is unknown

Cause: SQL*Net could not find your remote host information.

Action: Make sure you specified the hostname correctly on the command line. (Check your capitalization and spelling.) For TCP/IP, if the remote host is undefined in the **hosts** file, ask your system administrator to add an entry for the desired host.

ORA-6706: service not found

Cause: SQL*Net could not find service information for the specified service name.

Action: If you specified the service name on the command line or with the environment variable TLI_SERVER, make sure you specified it correctly. Otherwise, the service name defaults to **tcptlisrv.** If the service name is not in the **services** file, ask your system administrator to add it.

ORA-6707: connection failed

Cause: SQL*Net failed to establish the connection to a SQL*Net TCP/IP server due to an error encountered by the remote server, which has supplied a string describing the remote error.

Action: See the section "orasrv Messages" later in the appendix for the specific cause and action.

ORA-6708: message receive failure

Cause: SQL*Net encountered an error receiving a message from the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6709: message send failure

Cause: SQL*Net encountered an error sending a message across the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6710: send interrupt break message failed

Cause: SQL*Net failed to send a break message while handling an interrupt signal from the user.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6711: error on bind

Cause: SQL*Net failed to assign a network address to the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6712: error on accept

Cause: SQL*Net failed to accept a connection request from the client.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6713: error on connect

Cause: SQL*Net failed to connect the client to the remote server. The network line to the remote host maybe down.

Action: Use telnet to make sure that the remote host is accessible.

ORA-6720: SID lookup failure

Cause: The database SID supplied in the connect string was not recognized by the remote host.

Action: Ask your system administrator to add the appropriate SID entry to the **/etc/oratab** file on the remote host. The **tcptliutl** command can be used to find out which SIDs are available on the remote host.

ORA-6721: spurious client request

Cause: The remote SQL*Net server received an undefined request.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6722: connection setup failure

Cause: The remote SQL*Net server rejected the connection request, and the client was unable to retrieve an error code or message.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6730: unable to clone device

Cause: SQL*Net failed to open the Streams clone device associated with the transport provider.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6731: cannot alloc t_call

Cause: SQL*Net cannot allocate space for the client's connection information.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6732: cannot alloc t_discon

Cause: SQL*Net cannot allocate space for the client's disconnection information.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6733: failed to receive disconnect

Cause: SQL*Net failed to receive art expected disconnection message during connection release.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6734: cannot connect

Cause: SQL*Net failed to connect the client to the remote server.

Action: Check that the remote SQL*Net server is running the **tcptliutl** command.

ORA-6735: client failed to close error corm

Cause: SQL*Net failed to properly close a connection after an error was received.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6736: server not running

Cause: SQL*Net timed out while attempting to connect to the remote SQL*Net server.

Action: Using the **tcptliutl** comman d, check that the remote SQL*Net server is running. If it is not, ask your system administrator to start it.

ORA-6737: connection failed

Cause: SQL*Net could not establish a connection to the remote SQL*Net server.

Action: Using the **tcptliutl** command, check that the remote SQL*Net server is running. If it is not, ask your system administrator to start it.

ORA-6741: unable to open protocol device

Cause: The SQL*Net server failed to open the Streams device associated with the transport provider.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6742: cannot alloc t_bind

Cause: The SQL*Net server cannot allocate space for its requested network address.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6743: cannot alloc t_bind

Cause: The SQL*Net server cannot allocate space for its actual network device.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6744: listener cannot bind

Cause: The SQL*Net server failed to assign the correct network address on which to listen for connections.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6745: listener already running

Cause: The network address on which the SQL*Net server awaits connection request is in use, possibly because the server is already running.

Action: Ensure that the SQL*Net server is not already running. If it is not running and this message recurs, contact your Customer Support Representative.

ORA-6746: cannot alloc t_call

Cause: SQL*Net cannot allocate space for the SQL*Net server's connection information.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6747: error in listen

Cause: The SQL*Net server encountered an error while listening for connection requests.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6748: cannot allocate t_discon

Cause: SQL*Net cannot allocate space for the SQL*Net server's disconnection.

Actio: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6749: option not allowed across network

Cause: The requested SQL*Net server command must be issued from the same host on which the server is running.

Action: Log into the remote host and try again.

ORA-6750: sync failed

Cause: The ORACLE process started by the SQL*Net server was unable to synchronize its inherited connection.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6754: unable to get local host address

Cause: The name of the remote host to connect to was not specified, and the name of the local host cannot be retrieved from the **hosts** file. Contact your system administrator.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6755: cannot close transport endpoint

Cause: The SQL*Net server was unable to close a connection after passing it to an ORACLE process.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6756: cannot open /etc/oratab

Cause: The SQL*Net server could not open the file used to define the locations of remote databases.

Action: Ask your system administrator to check that the file exists and has the correct permissions.

ORA-6757: server got bad command

Cause: The SQL*Net server received an invalid command.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6760: timeout reading orderly release

Cause: SQL*Net was not able to retrieve an expected disconnect message while closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6761: error sending orderly release

Cause: SQL*Net encountered an error sending a disconnect message closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6762: error reading orderly release

Cause: SQL*Net encountered an error receiving an expected disconnect message while closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6763: error sending disconnect

Cause: SQL*Net encountered an error sending a disconnect message while closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6764: error reading disconnect

Cause: SQL*Net was not able to retrieve art expected disconnect message while closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6765: error awaiting orderly release

Cause: SQL*Net encountered an error awaiting a disconnect message while closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6766: close failed during release

Cause: SQL*Net failed to close the communication channel after receiving a disconnect message.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6767: alloc failed during release

Cause: SQL*Net cannot allocate space for disconnection information while closing the communication channel.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6770: error sending version

Cause: SQL*Net encountered an error while sending its version information during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6771: error reading version

Cause: SQL*Net encountered an error while awaiting the expected version information during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6772: error sending command

Cause: SQL*Net encountered an error while sending a command message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6773: error reading command

Cause: SQL*Net encountered an error while awaiting the expected command message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6774: error sending break mode

Cause: SQL*Net encountered art error while sending a break-mode message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6775: error reading break mode

Cause: SQL*Net encountered an error while awaiting the expected break-mode message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6776: error sending parameters

Cause: SQL*Net encountered an error while sending the connection parameters during connection establishment.

Action Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6777: error reading parameters

Cause: TCP/IP TLI encountered an error while awaiting the expected connection parameter message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6778: error sending code

Cause: SQL*Net encountered an error while sending the completion status message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6779: error reading code

Cause: SQL*Net encountered an error while awaiting the expected completion status message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6780: recv error code failed

Cause: SQL*Net encountered an error while awaiting an expected error message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6781: error reading negotiation string

Cause: SQL*Net encountered an error while awaiting the expected negotiation message during connection establishment.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6790: poll failed

Cause: SQL*Net was unable to poll the communication channel for possible incoming messages.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6791: poll returned error event

Cause: SQL*Net received an unexpected event while polling the communication channel for possible incoming-messages.

Action: Check the operating system error message or number supplied with this ORACLE error and then contact your Customer Support Representative.

ORA-6792: server cannot exec oracle

Cause: The remote SQL*Net server was unable to start an ORACLE process on behalf of the client.

Action: Note the operating system error message or number and contact your system administrator. The permissions on the remote ORACLE program may be set incorrectly.

ORA-6793: server cannot create new process

Cause: The remote SQL*Net server was unable to start an ORACLE process on behalf of the client.

Action: Note the operating system error message or number and contact your system administrator. The remote host may be unable to create any new processes due to a full process table.

SQL*Net TCP/IP for UNIX Error Messages

The following sections describe the error messages you might receive if you are using SQL*Net TCP/IP and SQL*Net TCP/IP TLI with UNIX systems.

orasrv Error Messages

If the orasrv process does not start properly, you may receive one of the following messages if you are using SQL*Net TCP/IP for UNIX:

tcp/orasrv: unknown service

Cause: This message may occur when:

1. Starting up orasrv on the host machine without a services file.
2. orasrv is not defined in /etc/services.
3. /etc/services is not readable by the user who started orasrv.

Action: Run the checkTCP utility (described in the next section) to determine the exact problem.

orasrv: accept

Cause: This message occurs when orasrv fails to accept the connection from the host.

Action: Run the checkTCP utility (described in Chapter 4) to verify that TCP/IP is installed. If it is installed correctly and you still receive this message, contact your customer support representative.

Bad format in <oratab> on line <line #>

Cause: This message occurs when a bad line in /etc/oratab is found.

Action: Run the checkTCP utility (described in Chapter 4) to determine the problem.

Maximum number of database definitions exceeded.

Cause: This message occurs when /etc/oratab defines more than 26 databases.

Action: Remove some database definitions from the file.

tcpctl Error Messages

If you are using the tcpctl utility to monitor orasrv, you might receive one of the following messages:

orasrv: server already running: Address already in use

Cause: Either the orasrv process was already running, or you tried to startup orasrv right after you shut it down, and the system did not have enough time to free up resource.

Action: In the second case, wait for a minute before re-invoking orasrv; the system needs some time to cleanup resource. If you still get the same message after trying several times, see whether there are ORACLE shadow processes already running, utilizing the SQL*Net TCP/IP interface.

You can do this by using this command:

```
$ ps -fe |fgrep  oracle
```

to see if there are processes listed as oraclesidT:. Ask the remote user to log off ORACLE, or else kill these processes. Then try invoking orasrv again.

tcputl: permission denied

Cause: You attempted to control orasrv, but you did not own it. Only the owner of the orasrv process has permission to control it.

Action: Ask your DBA to perform the desired operation.

tcputl: n subprocesses are active; server will not be stopped

Cause: You could not shut down orasrv because several ORACLE shadow processes are actively using the SQL*Net TCP/IP interface.

Action: Ask remote users to log off ORACLE before you shut down the orasrv process. Do not use the shell command kill -9 to forcibly stop orasrv. If you attempt this, the shadow processes will become “defunct processes” and continue to use network resource. You will not be able to start up orasrv again.

SQL* Net TCP/IP for VAX OpenVMS Error Messages

If you are using the TCPCTL utility to monitor the ORASRV process, you might receive one of the following messages:

TCPUTL: unable to get host address

Cause: The file does not contain an entry for *localhost*.

Action: The administrator must edit the HOSTS file to add the *localhosts* entries, and provide the following information: the servers' addresses, the names of the remote systems, and any optional aliases.

TCPUTL: unable to get server address

Cause: The SERVICES file does not contain an entry for ORASRV.

Action: The administrator must edit the SERVICES file to add the ORASRV entries. The SERVICES file is an operating system file that keeps track of which services are available (such as SQL*Net), as well as the TCP/IP port (socket) number they are attached to. It also contains optional aliases (for example, ORASRV).

TCPUTL: server not running

Cause: The TCPSRV process has not been started.

Action: The administrator must perform the TCPCTL START procedures.

TCPUTL: server is busy/not responding

Cause: The TCPSRV process is temporarily too busy to respond.

Action: Try again.

TCPSRV: error- SERVICES file lookup failure

Cause: The SERVICES file does not contain an entry for ORASRV.

Action: The administrator must edit the SERVICES file to add the orasrv entries. The services file is an operating system file that keeps track of which services are available (such as SQL*Net), as well as the TCP/IP port (socket)number they are attached to. It also contains optional aliases (for example, ORASRV).

TCPSRV: error - no SID mappings loaded

Cause: There are no valid SIDMAP entries in CONFIG.ORA.

Action: The administrator must make an entry in the CONFIG.ORA file indicating the location of the database(s).

TCPSRV: error - invalid default SID specified

Cause: The SIDMAPFLT entry in CONFIG.ORA refers to an SID that has not been defined.

Action: The administrator must define a valid default SID in CONFIG.ORA

TCPSRV: error - can't find SID mapping directory "<dir>"

Cause: The directory was specified in a SIDMAP entry in CONFIG.ORA but does not exist.

Action: The administrator should verify that the SIDMAP entry in CONFIG.ORA is valid and, if not, correct it.

TCPSRV: error - can't find SID mapping file "<file>"

Cause: The file was specified in a SIDMAP entry in CONFIG.ORA but does not exist.

Action: The administrator should verify that the SIDMAY entry in CONFIG.ORA is valid and, if not, correct it.

TCPSRV: error - bad device on SID mapping "<dev>"

Cause: The device was specified in a SIDMAP entry in CONFIG.ORA but is not accessible.

Action: The administrator should verify that the SIDMAP entry in CONFIG.ORA is valid and, if not, correct it.

TCPSRV: error - doing access test on <file>"

Cause: The file was specified in a SIDMAP entry in CONFIG.ORA but does not exist.

Action: The administrator should verify that the SIDMAP entry in CONFIG.ORA is valid and, if not, correct it.

TCPSRV: error - server already running

Cause: The TCPSRV process has already been started.

Action: It's unnecessary to run TCPCTL START at this point.

TCPSRV: error - server port is not (yet) available

Cause: This message appears when the START command is issued immediately after a STOP command.

Action Wait a moment. The STOP command closes the port and it takes a few seconds (longer if the system is very busy) before the port becomes available for use again.

TCPSRV: error - config file <file> not found

Cause: The file CONFIG.ORA doesn't exist.

Action: Contact the system administrator.

TCPSRV: error - insufficient privileges; SYSNAM and DETACH

Cause: TCPSRV requires the VMS privileges SYSNAM and DETACH.

Action: Contact the system administrator.

TCPSRV: error - insufficient privileges; CMKRNL and PRMMBX

Cause: User-level access control has been enabled. TCPSRV requires the VMS privileges CMKRNL and PRMMBX.

Action: Contact the system administrator.

TCPSRV: error - no privilege for usemame "<name>" in mapping

Cause: User-level control has been enabled. The UMT contains a username for which TCPSRV has no privilege.

Action: Contact the system administrator.

TCPSRV: error - bad username "<name>" in mapping table

Cause: User-level control has been enabled. The UMT contains a username that does not exist.

Action: Contact the system administrator.

GLOSSARY

alias A nickname for a host (server) or a set of parameters.

buffer An area of memory used by the network driver to pass data between two points on a network.

client A user, software application, or computer that requests services from another application or computer (the server.)

connect string The set of parameters, including a protocol, that SQL*Net uses to connect to a specific ORACLE instance on the network.

database link An object stored in the local database that identifies a remote database, a communication path to that database, and optionally, a user name and password. Once defined, the database link is used to access the remote database.

default A value used by the system when a parameter has not been specified. You can designate defaults for some parameters and the system provides defaults for other required values.

distributed database: One logical database whose data is spread across two or more computers. Users interact with the single, logical database, not the individual parts.

distributed processing Division of front-end and back-end processing to different computers. SQL*Net supports distributed processing by transparently connecting applications to remote databases.

Ethernet A network system that is fully specified as IEEE 802.3 and commonly used in local area networks.

host A computer that provides a shared resource on a network. Also called a server computer.

hosts file A file used by most TCP/IP implementations that identifies the Internet addresses of the available hosts on the network.

Internet address A number that identifies a TCP/IP network address.

IP (Internet Protocol) A protocol used between Internet nodes.

LAN Local-area network that provides limited distance, high speed communication transmissions, and is commonly supported by Ethernet.

location transparency A distributed database characteristic that allows applications to access data tables without knowing their location.

network prefix An alphanumeric string that specifies a network protocol. The network prefix for TCP/IP is **T.**

node A particular CPU on a network that is identified by a name.

orasrv The server process used by SQL*Net TCP/IP.

oraxdc The server process used by Data General AOS/VS systems that listens for client connection requests.

packet Information including data and control elements (and possibly error control information) that is switched and transmitted as a whole unit of information. The data and control elements (and any error control information) are arranged in a specified format.

parameter Information passed to a program, command, or function, such as a file specification, a keyword, or a constant value.

port an address associated with a specific application; see socket.

process-to-process communication Communication between two tasks, or processes, in which they coordinate to guarantee reliable data transmission. Same as task-to-task or peer-to-peer communication.

protocol A set of standards, or rules, governing the operations of a communications link

proxy login A login across a network to an ORACLE database that does not require a username and password.

server computer A computer where the ORACLE kernel and database(s) reside. A server must run a multi-user operating system and support a multi-user ORACLE database. Same as host.

server process An ORACLE process that receives SQL statements from the client and sends them to the ORACLE database on the server machine. It executes the SQL statements and sends the data and status back to the client.

services file A file that identifies the available services on the network.

socket A TCP/IP port number associated with a specific application.

SQL*Net An ORACLE product that works with the ORACLE RDBMS and enables two or more computers that run the RDBMS to exchange data through a third-party network.

system ID (SID) A code used to identify a particular database on a host system and allows SQL*Net to differentiate between all the ORACLE instances on a network.

task A single operating system process.

task-to-task communication See process-to-process communication.

TCP (Transmission Control Protocol) A protocol that implements socket-based connections and is commonly used for communication over an Ethernet.

TCP/IP A family of related protocols used to exchange information on a network.

tcpctl A SQL*Net TCP/IP control program that monitors the activity of orasrv.

tcptlictl A SQL*Net TCP/IP TLI control program that monitors the activity of tcptlisrv.

tcptlisrv The server process used by SQL*Net TCP/IP TLI to listen for client connection requests.

telnet A TCP/IP utility that connects to another node on a network and provides a means to test the network.

TLI (Transport Layer Interface) The programmatic interface to the TCP/IP protocol used by some AT&T System V UNIX machines.

two-task architecture A software "division of labor" in which two separate tasks (processes) cooperate to accomplish a single job. In a two-task ORACLE RDBMS architecture, the ORACLE kernel acts as the back end (server process) and an ORACLE application acts as the front end (client process).

UserMapping Table (UMT) A table that is set in the VAX/VMS CONFIG.ORA file to identify valid client usernames for access to hosts containing databases.

Valid Node Table (VNT) A table that is set in the VAX/VMS CONFIG.ORA file to identify nodes that are able to access hosts containing databases.

INDEX