

Oracle® Rdb

Oracle SQL/Services Server Configuration Guide

Release 7.1.6 for OpenVMS Alpha and VAX

May 2005

This document contains configuration information specific to Oracle SQL/Services release 7.1.6 for OpenVMS Alpha and VAX and for OCI Services for Oracle Rdb.

Oracle Rdb Oracle SQL/Services Server Configuration Guide, Release 7.1.6 for OpenVMS Alpha and VAX

Copyright © 1995, 2005, Oracle Corporation. All rights reserved.

The Programs (which include both the software and documentation) contain proprietary information of Oracle Corporation; they are provided under a license agreement containing restrictions on use and disclosure and are also protected by copyright, patent, and other intellectual and industrial property laws. Reverse engineering, disassembly, or decompilation of the Programs is prohibited.

The information contained in this document is subject to change without notice. If you find any problems in the documentation, please report them to us in writing. Oracle Corporation does not warrant that this document is error free. Except as may be expressly permitted in your license agreement for these Programs, no part of these Programs may be reproduced or transmitted in any form or by any means, electronic or mechanical, for any purpose, without the express written permission of Oracle Corporation.

If the Programs are delivered to the U.S. Government or anyone licensing or using the programs on behalf of the U.S. Government, the following notice is applicable:

U.S. GOVERNMENT RIGHTS Programs, software, databases, and related documentation and technical data delivered to U.S. Government customers are "commercial computer software" or "commercial technical data" pursuant to the applicable Federal Acquisition Regulation and agency-specific supplemental regulations. As such, use, duplication, disclosure, modification, and adaptation of the Programs, including documentation and technical data, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement, and, to the extent applicable, the additional rights set forth in FAR 52.227-19, Commercial Computer Software--Restricted Rights (June 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065

The Programs are not intended for use in any nuclear, aviation, mass transit, medical, or other inherently dangerous applications. It shall be the licensee's responsibility to take all appropriate fail-safe, backup, redundancy, and other measures to ensure the safe use of such applications if the Programs are used for such purposes, and Oracle Corporation disclaims liability for any damages caused by such use of the Programs.

Oracle is a registered trademark, and Oracle Rdb, Oracle SQL/Services, Oracle Rdb7, and SQL*Net are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

The Programs may provide links to Web sites and access to content, products, and services from third parties. Oracle is not responsible for the availability of, or any content provided on, third-party Web sites. You bear all risks associated with the use of such content. If you choose to purchase any products or services from a third party, the relationship is directly between you and the third party. Oracle is not responsible for: (a) the quality of third-party products or services; or (b) fulfilling any of the terms of the agreement with the third party, including delivery of products or services and warranty obligations related to purchased products or services. Oracle is not responsible for any loss or damage of any sort that you may incur from dealing with any third party.

Contents

Send Us Your Comments	xiii
Preface	xv
Intended Audience	xv
Operating System Information	xv
Structure	xvi
Related Documentation	xvi
Conventions	xvii
1 Overview	
1.1 Oracle SQL/Services	1-1
1.1.1 Server Management Utilities	1-5
1.1.2 Privileges Needed to Manage a Server	1-6
1.1.3 Running the SQLSRV_MANAGE Utility	1-6
1.1.4 Running the Oracle SQL/Services Manager GUI	1-7
1.2 Online Versus Offline Server Management	1-7
1.3 OCI Services for Oracle Rdb	1-10
1.3.1 Oracle Call Interface	1-11
1.3.2 Server-Side Solution	1-11
1.3.3 Common Application Development	1-12

2 Managing an Oracle SQL/Services System

2.1	Getting Started.....	2-1
2.2	Planning an Oracle SQL/Services Server Configuration	2-2
2.3	Setting Shared Memory Size	2-3
2.4	Managing Server Components	2-6
2.4.1	Managing a Server	2-6
2.4.2	Managing a Dispatcher.....	2-10
2.4.3	Managing a Service.....	2-13
2.5	Setting Up Dispatchers and Transport Selection.....	2-17
2.6	Setting Up Services and Types of Reuse.....	2-18
2.6.1	Session Reusable Universal Services.....	2-20
2.6.2	Session Reusable Database Services.....	2-21
2.6.3	Transaction Reusable Database Services	2-22
2.6.4	When to Use Session Reusable Versus Transaction Reusable Database Services	2-23
2.7	Setting Database Access Authorization.....	2-24
2.7.1	Specify a Default Connect User Name for the SQL/Services Protocol	2-27
2.7.2	Grant or Restrict Access to a Service.....	2-28
2.7.3	Provide Arbitrary or Predefined Access to Data.....	2-29
2.8	Setting Up Security on Servers.....	2-29
2.8.1	Client Identification and Authentication	2-30
2.8.2	Service Access Authorization	2-31
2.8.3	Database and Data Access Authorization	2-32
2.8.4	How Server Security Tiers Work Together for the SQL/Services Protocol	2-33
2.9	Understanding Database Access Authorization Models for Oracle SQL/Services.....	2-36
2.9.1	Accessing an Oracle Rdb Database.....	2-36
2.9.2	Setting the Process User Name and the Oracle Rdb System User Name	2-39
2.10	Considering Security for Selecting the Service Owner User Name.....	2-40
2.10.1	Execution Environment for Database Requests	2-40
2.10.2	Execution Environment for External Functions and Procedures	2-42
2.11	Setting the Attributes for Number of Executors.....	2-44
2.11.1	Configuring a Fixed Number of Executors for a Service.....	2-44
2.11.2	Configuring a Variable Number of Executors for a Service.....	2-44
2.12	Using a SQL Initialization File.....	2-45
2.13	Using SQL/Services Logical Names	2-46

3 Maintaining an Oracle SQL/Services Server

3.1	Monitoring Server Activity	3-1
3.2	Monitoring Client Connections	3-2
3.2.1	Client Connection States for Session Reusable Services	3-2
3.2.2	Client Connection States for Transaction Reusable Database Services	3-3

4 OCI Services for Oracle Rdb Processing

4.1	OCI Services for Oracle Rdb Features	4-2
4.1.1	OCI Message Mapping	4-2
4.1.2	Cursor Management	4-2
4.1.3	Data Types	4-2
4.1.4	Data Definition Language	4-2
4.1.5	SQL Cursor Semantics	4-3
4.1.6	Oracle SQL ALTER SESSION Statement	4-3
4.1.7	Data Formatting	4-3
4.1.8	Statement Parsing	4-3
4.1.9	Data Type Descriptions	4-4
4.1.10	Oracle Data Dictionary	4-6
4.1.11	Multischema Emulation	4-7
4.2	Initializing Your Server Environment	4-7
4.3	Compatibility with Oracle Forms	4-9
4.4	Using Oracle SQL/Services with Universal Services	4-9
4.5	Using the RDB_NATCONN71.COM Command File	4-10
4.5.1	Preparing a Database	4-11
4.5.2	Upgrading a Database	4-11
4.5.3	Removing OCI Services for Oracle Rdb	4-11
4.5.4	Adding Users and Passwords	4-12
4.5.5	Modifying Passwords	4-12
4.5.6	Removing a User	4-14
4.5.7	Showing Users	4-14
4.6	Defining Character Sets	4-14
4.6.1	Defining Character Sets on Server Systems	4-16
4.6.2	Defining Character Sets on Client Systems	4-17
4.6.3	Rules and Recommendations	4-17
4.7	Referencing an Oracle Rdb Database as a Database Link	4-17

4.7.1	CREATE DATABASE LINK Example	4-18
4.7.2	Database Link Restrictions.....	4-19
4.8	Using SQL EXPORT/IMPORT for OCI Services for Oracle Rdb Tables	4-20

5 SQL ALTER SESSION Statement

ALTER SESSION Statement	5-2
-------------------------------	-----

6 Management Commands

6.1	Syntax Conventions.....	6-1
6.2	How SQLSRV_MANAGE Commands Work	6-4
	–input Switch.....	6-8
	–output Switch.....	6-9
	@ Command.....	6-10
	ALTER DISPATCHER Command.....	6-11
	ALTER SERVER Command	6-16
	ALTER SERVICE Command.....	6-20
	CLOSE Command.....	6-28
	CONNECT TO SERVER Command.....	6-29
	CREATE DISPATCHER Command	6-32
	CREATE SERVER Command.....	6-36
	CREATE SERVICE Command	6-40
	DISCONNECT SERVER Command.....	6-47
	DROP Command.....	6-48
	DROP SERVER Command	6-50
	EXIT Command	6-51
	EXTRACT Command	6-52
	GRANT USE ON SERVICE Command.....	6-55
	HELP Command	6-57
	KILL EXECUTOR Command.....	6-58
	OPEN Command.....	6-60
	RESTART SERVER Command	6-61

REVOKE USE ON SERVICE Command	6-63
SET CONFIGURATION_FILE Command	6-65
SET CONFIRM Command	6-67
SET CONNECTION Command	6-68
SET OUTPUT Command	6-70
SET VERIFY Command	6-71
SHOW CLIENTS Command	6-72
SHOW CONNECTIONS Command	6-76
SHOW DISPATCHER Command	6-77
SHOW SERVER Command	6-80
SHOW SERVICE Command	6-82
SHOW SETTINGS Command	6-85
SHOW VERSION Command	6-86
SHUTDOWN DISPATCHER Command	6-87
SHUTDOWN SERVER Command	6-88
SHUTDOWN SERVICE Command	6-89
START DISPATCHER Command	6-90
START SERVER Command	6-91
START SERVICE Command	6-93

7 Logging and Troubleshooting

7.1	Problem Reporting	7-1
7.2	Error Messages	7-2
7.3	Log Files on the Server	7-2
7.3.1	Oracle SQL/Services Monitor Log File	7-3
7.3.2	Oracle SQL/Services Dispatcher Log Files	7-3
7.3.3	Oracle SQL/Services Executor Log Files	7-5
7.3.4	Using Debug Flags in the Executor Log File for OCI Services for Oracle Rdb	7-5
7.3.5	Enabling Logging from SQL and Oracle Rdb	7-7
7.3.6	Disabling Logging in SQL/Services	7-7
7.3.7	Moving Dispatcher Log Files Off the System Disk	7-7
7.4	Enabling Full Logging for OCI Services for Oracle Rdb	7-7

7.5	Inspecting SQL/Services API Log Files.....	7-8
7.5.1	Client and Driver Logging	7-8
7.5.2	Winsock Logging	7-9
7.5.3	Oracle Net Logging.....	7-9
7.5.4	ODBC Tracing	7-10
7.6	Process Failures	7-10
7.6.1	Monitor Process Failures.....	7-10
7.6.2	Dispatcher Process Failures	7-10
7.6.3	Executor Process Failures	7-11
7.7	Investigating Different Types of Problems	7-11
7.7.1	Network Transport Problems	7-11
7.7.2	User Authentication and Authorization Problems	7-12
7.7.3	Executor Failures During Service Startup.....	7-12
7.7.4	Executor Problems During Client Connect.....	7-12
7.7.5	Executor Problems During Client Request Execution	7-13
7.7.6	Server Failed Due to an Internal Error	7-13
7.7.7	Connections from Clients No Longer Work After Installing Oracle SQL/Services	7-14
7.7.8	Network Errors.....	7-14
7.8	Error Messages Returned to OCI Client Applications	7-15
7.8.1	Logon Error	7-15
7.8.2	Database Setup Error.....	7-16
7.8.3	SQL Initialization File Error	7-16
7.8.4	Errors Attaching to an Rdb Database or with Oracle SQL/Services Database Service..	7-16
7.9	Errors When Oracle SQL/Services Server or OCI Dispatcher Is Not Available	7-17
7.9.1	ERROR: ORA-12203: TNS:unable to connect to destination	7-17
7.9.2	Error When Oracle Net Service Name Is Not Defined	7-17

A Tables and Relations

A.1	Oracle Metadata Objects	8-1
A.2	Current Session Information.....	8-5
A.3	Handling 31-Character Object Names.....	8-6

Index

List of Examples

2-1	Default Universal Service with Database Access Authorization Set to Connect User Name	2-25
2-2	Universal Service with Database Access Authorization Set to Service Owner	2-25
2-3	Session Reusable Database Service with Access Authorization Set to Connect User Name	2-26
2-4	Transaction Reusable Database Service with Access Authorization Set to Service Owner..	2-27
4-1	Inserting an Oracle Date Literal into an ANSI Date Column	4-4
4-2	Inserting the Word CALL into a Procedure Call	4-4
4-3	Specifying the Western European Character Set	4-16
4-4	Specifying the French Language and Territory.....	4-16
4-5	TNSNAMES.ORA File.....	4-18
4-6	Database Link Definition	4-18
4-7	Valid Oracle SQL UPDATE Statement	4-19
4-8	Invalid Oracle SQL UPDATE Statement.....	4-20
4-9	Using PL/SQL When DML Does Not Work	4-20

List of Tables

1-1	Oracle SQL/Services Server Management Online and Offline Commands	1-8
1-2	OCI Services for Oracle Rdb Processing Features	1-12
2-1	Default Settings for Server Object Attributes	2-9
2-2	Default Settings for Dispatcher Object Attributes	2-12
2-3	Default Settings for Service Object Attributes	2-15
2-4	Oracle SQL/Services Service Attributes	2-20
2-5	When to Use Session Reusable Versus Transaction Reusable Database Services	2-24
2-6	SQL/Services Logical Names	2-46
4-1	Supported Character Sets	4-15
6-1	Oracle SQL/Services Objects and How Each Object Is Acted Upon by a Command	6-5
6-2	SQLSRV_MANAGE Environment Commands and Switches	6-7
7-1	Error Code Files for DECnet	7-15
7-2	Error Code Files for TCP/IP	7-15
A-1	OCI Services for Oracle Rdb Objects	8-2
A-2	ORA_SESSION Table	8-6

List of Figures

1-1	Simplest Client/Server Architecture.....	1-1
1-2	Oracle SQL/Services Server System.....	1-2
1-3	Oracle SQL/Services Client/Server Architecture.....	1-3
1-4	Client/Server Processing	1-11
2-1	Oracle SQL/Services Session Reusable Universal Services	2-21
2-2	Oracle SQL/Services Session Reusable Database Services	2-22
2-3	Oracle SQL/Services Transaction Reusable Database Services	2-23
2-4	Oracle SQL/Services Server Security	2-34
3-1	Client Connection States for Session Reusable Services	3-2
3-2	Client Connection States for Transaction Reusable Database Services.....	3-4

Send Us Your Comments

Oracle Rdb SQL/Services Server Configuration Guide, Release 7.1.6 for OpenVMS Alpha and VAX

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

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

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

- Electronic mail: nedc-doc_us@oracle.com
- FAX: 603.897.3825 Attn: Oracle Rdb
- Postal service:
Oracle Corporation
Oracle Rdb Documentation
One Oracle Drive
Nashua, NH 03062-2804
USA

If you would like a reply, please give your name, address, telephone number, and (optionally) electronic mail address.

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

Preface

Oracle Rdb is a general-purpose database management system based on the relational data model.

Oracle SQL/Services, a client/server component of Oracle Rdb, enables a client application program, invoked on a remote client computer running on a supported operating system or transport, to access Oracle Rdb databases.

OCI Services for Oracle Rdb (previously known as SQL*Net for Oracle Rdb) allows you to run existing OCI applications to access Rdb databases.

This manual describes how to maintain and tune Oracle SQL/Services server and OCI Services for Oracle Rdb systems.

Intended Audience

This manual is written for the system manager responsible for maintaining and fine-tuning Oracle SQL/Services and OCI Services for Oracle Rdb. System managers should refer to the installation guide, which provides important information about the installation of an Oracle SQL/Services system.

Operating System Information

You can find information about the versions of the operating system and optional software that are compatible with this release of Oracle Rdb and Oracle SQL/Services in the installation guides and release notes for Oracle Rdb and Oracle SQL/Services.

Contact your Oracle representative if you have other questions about compatibility.

Structure

This manual contains the following chapters:

Chapter 1	Introduces the Oracle SQL/Services system.
Chapter 2	Describes how to manage an Oracle SQL/Services system.
Chapter 3	Describes how to maintain an Oracle SQL/Services server.
Chapter 4	Explains in detail the features and benefits of OCI Services for Oracle Rdb.
Chapter 5	Explains how to use the ALTER SESSION command for OCI Services for Oracle Rdb.
Chapter 6	Describes the Oracle SQL/Services system management commands.
Chapter 7	Describes how to troubleshoot and enable logging for Oracle SQL/Services and OCI Services for Oracle Rdb.
Appendix A	Defines the tables, views, and other database objects that comprise the emulated Oracle data dictionary that the OCI Services for Oracle Rdb product stores in your Rdb database. This appendix also lists the file name, target location, and purpose of all files installed with OCI Services for Oracle Rdb software.

Related Documentation

For more information, see the other manuals in this documentation set, especially the following:

- *New and Changed Features for Oracle Rdb, Release 7.1*
- *Oracle Rdb Guide to SQL Programming*
- *Oracle Rdb SQL Reference Manual*
- *Guide to Using the Oracle SQL/Services Client API*
- The release notes and installation documents for Oracle Rdb release 7.1 and Oracle SQL/Services release 7.1.6. As part of the installation, the Oracle Rdb Oracle SQL/Services Release 7.1.6 Release Notes are provided as a PostScript file in the SYSS\$HELP directory.
- *Oracle Rdb7 Guide to Database Maintenance*

Conventions

In this manual, Oracle Rdb refers to Oracle Rdb for OpenVMS software. Release 7.1 of Oracle Rdb software is often referred to as V7.1.

The SQL interface to Oracle Rdb is referred to as SQL. This interface is the Oracle Rdb implementation of the SQL standard adopted in 1999, in general referred to as the ANSI/ISO SQL standard or SQL:1999. See the *Oracle Rdb Release Notes for OpenVMS Alpha* for additional information about this SQL standard.

Beginning with release 7.1, Oracle SQL/Services is a multiversion-only kit. The installation installs files using a variant naming convention. That is, variant file names and names of utilities may have a two-digit version number appended as the last two characters of its name. For example, the management client is SQLSRV_MANAGE71 and its log files are *71.log, and so forth.

OpenVMS means the OpenVMS Alpha or OpenVMS VAX operating system.

The following conventions are also used in this manual:

Convention	Meaning
. . .	Vertical ellipsis points in an example mean that information not directly related to the example has been omitted.
\$	The dollar sign represents the DIGITAL Command Language prompt in OpenVMS.
boldface text	Boldface type in text indicates a term defined in the text.
monospaced boldface text	Monospaced boldface type in text indicates user input.

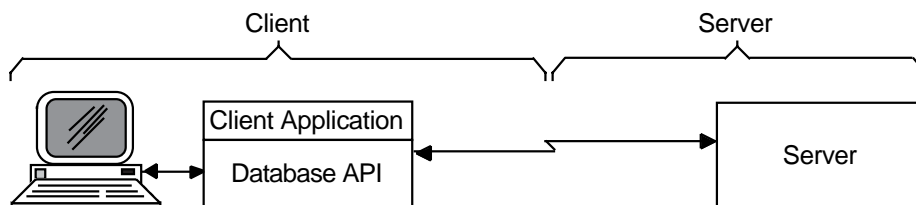
See [Section 6.1](#) for more information on syntax conventions used by the SQLSRV_MANAGE utility.

This chapter describes the components of Oracle SQL/Services and OCI Services for Oracle Rdb (formerly known as SQL*Net for Oracle Rdb) systems, and provides an overview of managing a SQL/Services system.

1.1 Oracle SQL/Services

A client/server system in its simplest form consists of a client, a network, and a server system. A **client** is a software program that uses a database application programming interface (API) to make database requests of a server, as shown in [Figure 1-1](#). The client may reside on the same platform as the server. Typically, however, the client application runs on a workstation or PC and accesses a database on a large server platform using a network that supports several transport protocols.

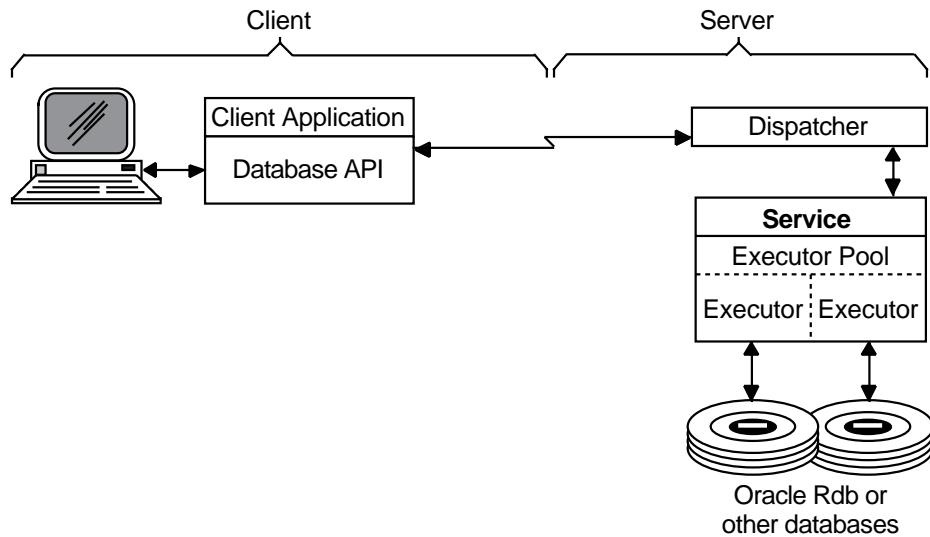
Figure 1-1 *Simplest Client/Server Architecture*



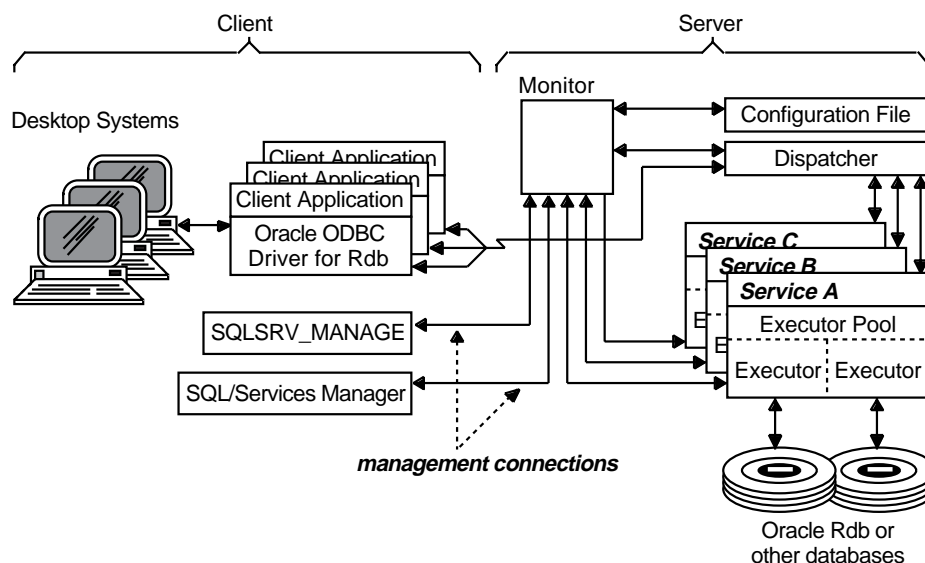
An Oracle SQL/Services **server** is a collection of cooperating processes on one node that includes a dispatcher process and a pool of executor processes that work on behalf of a service, as shown in [Figure 1-2](#). The **dispatcher** process handles all network communication between the client and the server. It reads client requests, queues these

requests for the executors of a service, and returns the responses from the executors of a service back to the client. An **executor** process works on behalf of a service and accepts the client requests from the dispatcher's queue, invokes the database engine to process the requests, and returns the results to the dispatcher. A **service** is a set of attributes that describes how clients access a database.

Figure 1–2 Oracle SQL/Services Server System



The Oracle SQL/Services server system also includes a monitor process to manage and control the server, a `SQLSRV_MANAGE` utility that runs on an OpenVMS local or remote system, or an Oracle SQL/Services Manager GUI server management utility that runs as a Windows client, and a configuration file in which to store server component definitions, as shown in [Figure 1–3](#).

Figure 1-3 Oracle SQL/Services Client/Server Architecture

An Oracle SQL/Services **client** is a software program that accesses data by selecting a service provided by a server using an agreed upon interface such as the Oracle SQL/Services API (which includes Microsoft ODBC connections), the Oracle OCI interface, the Oracle RMU interface, or the Oracle Rdb JDBC interface. The server responds by receiving and processing client requests and sending the results back to the client.

A **network** is made up of communications hardware and software through which the client APIs communicate with the Oracle SQL/Services server. Request messages from the client and response messages from the server travel over a DECnet, Transmission Control Protocol/Internet Protocol (TCP/IP), NetWare (IPX/SPX), AppleTalk-DECnet gateway, or Oracle Net communications link.

Server

An Oracle SQL/Services **server** describes the attributes of a collection of cooperating processes on one node that provides one or more services. The server in general includes all server component attribute definitions, which are contained in a configuration file. See [Service](#) and [Dispatcher](#) topics, included in this list, for more information about these server components. For the server object specifically, the attributes include information such as the

version of the server, the configuration file specification, the size of shared memory, and what network transports are supported for server management.

There can be only one server defined per configuration file. You can have only one server per version of Oracle SQL/Services started on a node at any given time.

Service

An Oracle SQL/Services **service** is a set of attributes that describes how clients access a database. Oracle SQL/Services accommodates the needs of different clients by supporting a range of service attributes that you tailor for each service provided by a server. The definition of a service includes information such as who can use the service, the database that is accessed by the service, the database engine version used by the service, how many clients can simultaneously use the service, and the number of executors that will be working on behalf of the service.

Executor

An Oracle SQL/Services **executor** is the process that works on behalf of a service.

An executor accepts client requests from dispatchers, calls SQL to process the requests, and returns the results to dispatchers. There is a pool of executor processes for each service that is started.

Dispatcher

An Oracle SQL/Services **dispatcher** is a process that is responsible for handling network communications for the clients and for the routing and scheduling of client requests to executors of a service. A dispatcher supports all services defined for a server. A single dispatcher typically supports more than one network transport, but can support only one service protocol. All clients except system management clients connect directly to a dispatcher.

Monitor

An Oracle SQL/Services **monitor** process provides overall management and control for the server, such as server startup and shutdown, reading and writing the configuration database, monitoring functions, and other management operations.

SQLSRV_MANAGE Client

The Oracle SQL/Services server management command-line interface lets you manage an Oracle SQL/Services server from an OpenVMS system.

Oracle SQL/Services Manager Client GUI

The Oracle SQL/Services server management graphical user interface (GUI) utility lets you manage an Oracle SQL/Services server from any Windows client. The Oracle SQL/Services Manager GUI runs on Windows 95, Windows 98, Windows 2000, Windows 2003, Windows NT X86, Windows ME, and Windows XP operating systems.

Configuration file

A **configuration file** contains all defined attributes for one server and its components, which include all service definitions, dispatcher definitions, and the list of users that are authorized to access the services provided by that server. This is also known as an **Oracle SQL/Services server configuration**, in that it represents one set of component definitions that are managed together for a server. Only one server can be defined in a configuration file. Typically, each server node has its own configuration file; however, it is possible to share a configuration file among multiple nodes in an OpenVMS cluster.

1.1.1 Server Management Utilities

You can manage an Oracle SQL/Services server in the following ways:

- Use the SQLSRV_MANAGE utility

You can use the SQLSRV_MANAGE utility from a local or remote node on an OpenVMS system and manage the server online or offline (you must be on a local node to manage the server offline).

Usually, you use the SQLSRV_MANAGE utility to manage a server configuration online by establishing a system management connection to a running server, then performing system management functions that operate on the running server as well as on the configuration file. In addition, you can use the SQLSRV_MANAGE utility to manage a server configuration offline by directly manipulating server component attributes in a configuration file. The only system management functions that you must perform offline are creating a new server configuration and starting a server. The SQLSRV_MANAGE utility accepts commands from the standard input device or from script files, and can be run interactively or in a batch job on an OpenVMS system.

- Use the Oracle SQL/Services Manager graphical user interface (GUI)

You can use the Oracle SQL/Services Manager GUI from a remote Windows client and manage the server online using one of the following clients: Windows 95, Windows 98, Windows 2000, Windows 2003, Windows NT X86, Windows ME, or Windows XP.

Use the Oracle SQL/Services Manager GUI to manage a server configuration online by establishing a system management connection to a running server, then performing

system management functions that operate on the running server or the configuration file or both.

You cannot use the Oracle SQL/Services Manager GUI to perform the offline server management functions of creating a new server configuration, starting a server, or shutting down a running server; nor can you execute script files. However, you can modify and restart a running server using the Oracle SQL/Services Manager GUI.

Usually, you will use either of these two utilities interactively to manage the server and its components online. For most server management tasks, use the Oracle SQL/Services Manager GUI because it is easy to learn and use. The only time you may need to use the `SQLSRV_MANAGE` utility is to start a server or to run any special scripts you might create.

See [Section 2.4](#) for more information about managing a server using the `SQLSRV_MANAGE` utility. For more information about managing a server using the Oracle SQL/Services Manager GUI, invoke the utility on a Windows client and select the Help pull-down menu.

1.1.2 Privileges Needed to Manage a Server

To start a server using the `SQLSRV_MANAGE` utility on OpenVMS, you must use an account that has been granted the `SETPRV` privilege or that has been granted all privileges. To make offline modifications to a server using the `SQLSRV_MANAGE` utility on OpenVMS, you must use an account that has been granted the `NETMBX`, `SYSLCK`, and `SYSPRV` privileges. To make online modifications to a server using the `SQLSRV_MANAGE` utility or the Oracle SQL/Services Manager GUI, you must use an account that has been granted use of the `SQLSRV_MANAGE` system management service for that server; however, you are not required to use an account that has been granted elevated privileges.

These privilege requirements are either less restrictive or identical to those needed to install Oracle SQL/Services on the OpenVMS platform. For more information, see the installation documentation for Oracle Rdb and Oracle SQL/Services.

1.1.3 Running the `SQLSRV_MANAGE` Utility

To run the `SQLSRV_MANAGE` utility, you first define a symbol to invoke the utility as follows:

```
sqlsrv_manage71 ::= $SYS$SYSTEM:sqlsrv_manage71
```

You then enter the command `sqlsrv_manage71` to invoke the `SQLSRV_MANAGE` utility. To use the `SQLSRV_MANAGE` utility interactively, invoke the utility, then enter system management commands in response to the `SQLSRV>` command-line prompt. To manage a

server online, the first command you use is usually the `CONNECT TO SERVER` command. To manage a server offline, you first use a `SET CONFIGURATION_FILE` command to specify the name of the server configuration file, if the file is not stored in the default location (see the [SET CONFIGURATION_FILE Command](#) for more information).

You can also use scripts with the `SQLSRV_MANAGE` utility. A `SQLSRV_MANAGE` script is a file containing the same commands that you would enter at the `SQLSRV>` prompt. You can invoke a `SQLSRV_MANAGE` script interactively at the `SQLSRV>` prompt using the `@` command. Alternatively, you can invoke the `SQLSRV_MANAGE` utility to read system management commands directly from a script. See the [-input Switch](#) in [Chapter 6](#) for more information.

Scripts are a practical tool for making changes to a server on a regular basis. For example, suppose you want to increase the minimum and maximum number of executors for a service to meet a peak load condition. You can use one script to increase the values and another to decrease the values. You can automate the execution of the scripts using batch jobs.

1.1.4 Running the Oracle SQL/Services Manager GUI

To run the Oracle SQL/Services Manager GUI, click the Oracle SQL/Services Manager GUI icon to bring up the Connect To Server window and enter the node name, user name, and password. Select the transport and the TCP/IP port ID or DECnet object name that you want to use to establish the connection. Once connected, you can begin to manage the server.

1.2 Online Versus Offline Server Management

You can manage a server either online or offline using the `SQLSRV_MANAGE` utility.

Online Server Management

Typically, you manage the server online. To manage a server online, you always connect to the server using the `CONNECT TO SERVER` command. Once connected, any changes you make to the server are written to the configuration file. If you alter a dynamic attribute, the change is also made to the running server. See [Section 2.4.1](#), [Section 2.4.2](#), and [Section 2.4.3](#) for a list of dynamic attributes. If you alter a nondynamic attribute of an object that is started, the system management utility displays a message that the object must be restarted for the change to take effect. The only time you need to restart the server is if the change to the server is to a nondynamic attribute of the server object itself, in which case changes take effect upon a server restart operation.

Offline Server Management

On occasion, you may need to manage a server offline to recover from an alteration that rendered the server unusable, such as setting too low a value for shared memory. To manage a server offline, you must use the SQLSRV_MANAGE utility. You cannot manage a server offline using the Oracle SQL/Services Manager GUI. If the configuration file is not stored in the default location (see the [SET CONFIGURATION_FILE Command](#) for more information), you must first select the configuration file by using the SET CONFIGURATION_FILE command before issuing any system management commands. Usually, you will manage a server offline only when the server is not running. However, you can manage a server offline even if the server is running. Any changes you make to the server configuration are written to the configuration file but *do not* affect the running server until the objects that have been changed are restarted. You must restart the entire server for a change to an attribute of the server object itself to take effect. You need only shut down and start the particular dispatcher or service for a change to an attribute of a dispatcher or service object to take effect. The only exception is that if you grant or revoke use of a service to or from a user name or identifier, then the change takes effect immediately.

[Table 1–1](#) summarizes which Oracle SQL/Services server management commands can be performed online, offline, or both and any restrictions that may apply.

Table 1–1 Oracle SQL/Services Server Management Online and Offline Commands

Command	Online	Offline	Comments
ALTER DISPATCHER	X	X	Offline changes do not affect a running dispatcher.
ALTER SERVER	X	X	Offline changes do not affect a running server.
ALTER SERVICE	X	X	Offline changes do not affect a running service.
CONNECT TO SERVER	X	–	For online server management only.
CREATE DISPATCHER	X	X	Can create a dispatcher either online or offline.
CREATE SERVER	–	X	Can only create a server offline.
CREATE SERVICE	X	X	Can create a service either online or offline.
DISCONNECT SERVER	X	–	For online server management only.
DROP DISPATCHER	X	X	Can delete a dispatcher either online or offline.
DROP SERVER	–	X	Can only delete a server offline.
DROP SERVICE	X	X	Can delete a service either online or offline.

Table 1–1 Oracle SQL/Services Server Management Online and Offline Commands

Command	Online	Offline	Comments
GRANT USE ON SERVICE	X	X	Offline changes affect running server.
KILL EXECUTOR	X	–	Can only kill an executor online.
RESTART SERVER	X	–	Can only restart a server online.
REVOKE USE ON SERVICE	X	X	Offline changes affect running server.
SET CONFIGURATION_FILE	–	X	For offline server management only.
SET CONNECTION	X	–	For online server management only.
SHOW DISPATCHER	X	X	Can display definitional attributes of a dispatcher online or offline; can only show the run-time attributes of a dispatcher (such as its state) online.
SHOW SERVER	X	X	Can display definitional attributes of a server online or offline; can only show the run-time attributes of a server (such as its state) online.
SHOW SERVICE	X	X	Can display definitional attributes of a service object online or offline; can only show the run-time attributes of an object (such as its state) online.
SHUTDOWN DISPATCHER	X	–	Can only shut down a dispatcher online.
SHUTDOWN SERVER	X	–	Can only shut down a server online.
SHUTDOWN SERVICE	X	–	Can only shut down a service online.
START DISPATCHER	X	–	Can only start a dispatcher online.
START SERVER	–	X	Can only start a server offline.
START SERVICE	X	–	Can only start a service online.

[Chapter 2](#) and [Chapter 3](#) describe managing and maintaining the server. [Chapter 6](#) contains reference material that describes `SQLSRV_MANAGE` commands. These chapters are provided primarily for the Oracle SQL/Services system administrator who is using the `SQLSRV_MANAGE` utility and its command-line interface. As appropriate, only the command-line interface of `SQLSRV_MANAGE` commands and syntax are used in text and examples in these chapters.

An Oracle SQL/Services system administrator who is using the Oracle SQL/Services Manager GUI client from a Windows-based PC system can refer to [Chapter 2](#) and [Chapter 3](#), but should be aware that the names of commands and syntax vary between the SQLSRV_MANAGE command-line interface and the Oracle SQL/Services Manager GUI, and no attempt is made to describe these differences, as most differences are relatively minor. Refer to the Oracle SQL/Services Manager GUI client Windows help for more specific information.

1.3 OCI Services for Oracle Rdb

OCI Services for Oracle Rdb (formerly known as SQL*Net for Oracle Rdb) provides an environment in which you can run existing OCI applications to access data in Oracle Rdb databases. The OCI applications use the Oracle Call Interface (OCI) to access and manage data in a database.

OCI Services for Oracle Rdb connects Oracle clients to Oracle Rdb servers. The unique advantage offered by OCI Services for Oracle Rdb is the ability to use Oracle SQL semantics to access data in Oracle Rdb databases.

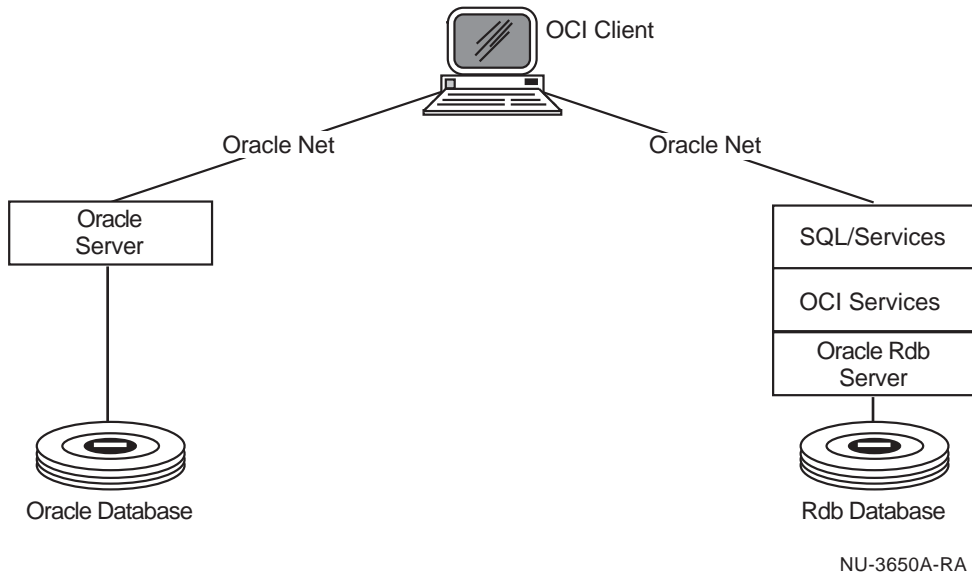
OCI Services for Oracle Rdb:

- Identifies itself to the client applications as an Oracle database server
- Emulates many of the Oracle SQL semantics
- Uses Oracle SQL/Services for network communications

OCI Services for Oracle Rdb broadens the range of your client applications by letting you build a single source code stream that runs against either an Oracle Rdb or Oracle RDBMS database instance.

For example, you can substitute the Oracle Rdb server for the Oracle RDBMS server when your application requires functions supplied by an Oracle Rdb database. If you use only the Oracle Rdb server, OCI Services for Oracle Rdb provides many of the capabilities of the OCI architecture to your Oracle Rdb applications.

[Figure 1–4](#) shows the client/server relationships in a OCI Services for Oracle Rdb environment.

Figure 1–4 Client/Server Processing

NU-3650A-RA

1.3.1 Oracle Call Interface

The unique capabilities of OCI Services for Oracle Rdb are made possible through the Oracle Call Interface (OCI), a key, open technology.

OCI applications and tools run in the OCI services client/server environment, enabling diverse combinations of server and client hardware and operating system environments. Because the OCI architecture separates the client user interface from the server implementation, it is possible to add new and different user interfaces to existing servers, and change the server implementation without any effect on the user interface.

1.3.2 Server-Side Solution

OCI Services for Oracle Rdb capitalizes on the flexibility of OCI by connecting your Oracle client applications directly to an Oracle Rdb server.

Because OCI Services for Oracle Rdb is designed as a server-side solution, it is as easy and cost-effective to use with a diverse set of client platforms as it is to use the Oracle server with a diverse set of client platforms.

The application programming interface (API) software that you use on client systems is distinctly separate from your OCI Services for Oracle Rdb server-side implementation. Any API software that you use to code client applications, including OCI or any of the Pro* compilers, must be separately purchased and installed for each client system.

To build new OCI applications, you must install the particular OCI software needed to develop and build new OCI applications. Your existing OCI applications will run without the need to purchase, install, configure, or manage additional client software to use OCI Services for Oracle Rdb.

OCI Services for Oracle Rdb appears as an Oracle server to the client, and the client interacts with OCI Services for Oracle Rdb in the same way it interacts with the Oracle RDBMS server. The client typically queries the Oracle data dictionary to obtain metadata information about the target database, and performs a number of other OCI calls to query and manipulate the data in the database.

Note: The metadata for the Oracle Rdb SQL dialect is very different from that of the Oracle server. The Oracle data dictionary you create using the supplied Oracle Rdb SQL program allows OCI Services for Oracle Rdb to emulate most aspects of the Oracle data dictionary that are important to client software.

1.3.3 Common Application Development

OCI Services for Oracle Rdb was built to help SQL programmers create software that can run against both the Oracle Rdb server and the Oracle RDBMS server.

To help you to run OCI Services for Oracle Rdb client applications and tools against an Oracle Rdb server, OCI Services for Oracle Rdb augments the features of Oracle Rdb SQL with the processing features described in [Table 1–2](#).

Table 1–2 OCI Services for Oracle Rdb Processing Features

Function	Description
Cursor management	Manages OCI cursors for each statement, then ties the cursors to Oracle Rdb SQL statements.
OCI message mapping	Maps OCI calls to Oracle Rdb dynamic SQL calls.
Oracle data types	Describes and converts Oracle Rdb data types as Oracle data types.
Data formatting	Performs Oracle style formatting in which the Oracle server formats data for the client or receives formatted data from the client. Formatted information is described to the server by the Oracle SQL ALTER SESSION statement.

Table 1–2 OCI Services for Oracle Rdb Processing Features(Cont.)

Function	Description
Statement changes	Reprocesses a failed SQL statement after performing the necessary modifications to make the statement comply with Oracle Rdb syntax. This processing is done for a limited number of syntax differences between Oracle and Oracle Rdb. This reprocessing usually allows the SQL statement to succeed.
Statement type	Obtains the type of SQL statement being parsed from Oracle Rdb SQL and returns it to the client.
Data definition language (DDL)	Provides some DDL and SQL cursor semantics that provide behavior similar to what you get from an Oracle server. For example, before and after each DDL request, a COMMIT statement is issued.
Data dictionary	Provides a collection of views and stored procedures that emulate the Oracle data dictionary to provide the style of metadata tables typical to Oracle.
Multischema emulation	Emulates a multischema environment that is similar to what you get with Oracle multischema databases (all Oracle databases are multischema databases, while most Oracle Rdb databases are not). A table name cannot be used in more than one schema, but the data dictionary provides a multischema appearance.

All these features allow for common application development between the Oracle RDBMS server and the Oracle Rdb server (using OCI Services for Oracle Rdb).

[Chapter 4](#) describes OCI Services for Oracle Rdb processing in more detail.

Managing an Oracle SQL/Services System

Managing an Oracle SQL/Services system requires knowledge of the client and network components, together with dispatchers, services, and a server, as described in [Chapter 1](#). You should have a general understanding of how each component works with other components in the client/server architecture and how the components within the server system operate. This chapter describes how to create and manage the server components.

Unless otherwise indicated, the information in this chapter applies to SQL/Services and OCI Services for Oracle Rdb protocols.

2.1 Getting Started

After you install and start the default Oracle SQL/Services server, you may want to perform some additional tasks to ensure its optimum performance and to troubleshoot problems. These tasks include:

- Planning an Oracle SQL/Services server configuration
- Setting shared memory size
- Managing server components
- Setting up dispatchers and transport selection
- Setting up services and types of reuse
- Setting up security on servers
- Understanding how Oracle SQL/Services implements the database access authorization models
- Considering security when selecting the service owner user name
- Setting executor attributes

- Using a SQL initialization file
- Using SQL/Services logical names

Each topic is discussed in the sections that follow.

2.2 Planning an Oracle SQL/Services Server Configuration

Your initial working Oracle SQL/Services server is defined by a configuration file. That file contains object definitions and characteristics for the server, dispatchers, services, and a set of authorized users for each service. You can display the current definition of each object with a SHOW command, read through the attribute settings, and from this basic understanding, take the following steps to plan your server configuration:

1. Determine your own requirements for your server system.
2. Learn about each object and how best to manage it.
3. Apply what you learned toward meeting your server system requirements.

Determining Server System Requirements

As the Oracle SQL/Services system administrator, you must determine the requirements for your server system. You should investigate the following:

- Is Oracle SQL/Services installed on a single node or in a cluster? Do different nodes require different dispatchers and services?
- What do you know about your user community? How many clients are there in total? How many clients will use the system at peak periods?
- What transports are available for client/server communication? How many ports are available for each transport?
- What version of Oracle Rdb do you have installed?
- What are the specific applications users want to run? Are users attaching to the same database or many different databases? What kinds of transactions will be run?

These are the most important questions to answer. As you proceed, other questions may arise that will help you to understand your own server requirements. You should also begin to devise a plan for how to best meet the server needs of your user community and how to tune your server system to achieve maximum performance.

Learning About Server Objects

To start, ask the following questions about each server object:

- Which attributes do I need to monitor?
- Which attributes should I be most concerned about managing?

To answer these questions, it is important to understand the meaning of the default value of each attribute and then determine which attributes need to be monitored and adjusted. In general, all default settings of attributes for the default server system are sufficient to get started. [Table 2–1](#), [Table 2–2](#), and [Table 2–3](#) provide a summary of the default values for the server, dispatcher, and service objects. Following each table is a brief description of which attributes to monitor and adjust.

Achieving Server System Requirements

By answering specific questions about the most important attributes for each server component, you can determine what modifications you may need to make to your server system. As you implement your plan, you learn how to create and alter server component objects and apply these changes toward meeting your server system requirements.

As you learn how to monitor and tune each object, you can begin to optimize the performance of the server and tailor your Oracle SQL/Services server to make it ideal for your database client/server environment. For example, once you know what applications your users want to run, you can decide on the kinds of services to provide for these client applications.

The most important items that you should consider for establishing a running server are discussed in [Section 2.4](#) through [Section 2.8](#).

After you tailor an Oracle SQL/Services server to meet your client/server requirements, the next task is to understand more about maintaining the server (see [Chapter 3](#) for more information).

2.3 Setting Shared Memory Size

You can set the size of shared memory that the server uses by specifying a value for the `MAX_SHARED_MEMORY_SIZE` argument of the `ALTER SERVER` command. By default, the server uses 2000 kilobytes (2 megabytes) of shared memory.

Setting the `MAX_SHARED_MEMORY_SIZE` argument is important for optimizing the resource usage of the server system. The goal is to use the smallest amount of shared memory possible to provide the required services. This section explains how the Oracle SQL/Services server uses shared memory and how to set the `MAX_SHARED_MEMORY_SIZE` argument for the best resource usage.

You can change the value for shared memory using the ALTER SERVER command. However, this is not a dynamic attribute and requires that you restart the server. For example, to set the value to 4000 kilobytes:

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> ALTER SERVER MAX_SHARED_MEMORY_SIZE 4000;  
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect  
SQLSRV> RESTART SERVER;  
Disconnected from Server
```

The following are the two main costs associated with allocating shared memory:

- Disk space for the system page file
You must ensure that you have sufficient system page file space available to accommodate an increase in the size of shared memory. You must also ensure that the GBLPAGFIL SYSGEN parameter is set high enough to accommodate an increase in the size of shared memory.
- Virtual memory for each component process of the server
Mapping shared memory makes each component process of the server use more virtual memory, and thus consumes incremental operating system resources.

Internally, Oracle SQL/Services manages shared memory in units of 65,536 bytes. The actual size of shared memory may be less than the MAX_SHARED_MEMORY_SIZE attribute because the size is rounded down to the nearest even 65,536-byte unit.

The server shared memory does not shrink or grow as the server runs. More or less of it may be in use at a given time. When you issue a SHOW SERVER command in the SQLSRV_MANAGE application for the server that you are connected to, SQLSRV_MANAGE will show three values:

- Total memory
This number is static for a given run of the server. You can alter the MAX_SHARED_MEMORY_SIZE argument for a server, and it takes effect when you issue a RESTART SERVER command. Total memory is the number of shared memory units mapped by the server.
- Free memory
This is the number of shared memory units that are completely unused by the Oracle SQL/Services server.
- Partly allocated memory

A shared memory unit may not be completely used in one piece. It is often subdivided into smaller pieces. Shared memory units that are subdivided and partly used are reported as partly allocated memory. It is currently not practical to display the usage within the subdivided unit.

Free memory and partly allocated memory describe the shared memory units that can still be allocated. By subtracting those units from the total units, you can determine the shared memory units that are entirely used.

The minimum value for `MAX_SHARED_MEMORY_SIZE` is 132 KB, which provides two shared memory units. This is sufficient to start the monitor, connect to it from the `SQLSRV_MANAGE` application, and run one or two executors serving one or two clients.

The maximum value for `MAX_SHARED_MEMORY_SIZE` is 2,000,000 KB. Lower values should suffice for all applications.

In general, plan for the following shared memory usage:

- For each executor and dispatcher that you plan to run, allow about 3 KB.
- For each Oracle SQL/Services client connection that you plan to support, you need to take into account the base shared memory usage for a client and add to that the memory used for communication buffers.

The base shared memory usage is about 11 KB.

An Oracle SQL/Services application minimally consumes two communication buffers. The default buffer size is 1.3 KB, so the minimum size for an Oracle SQL/Services client is 15 KB (11 KB base + 4 KB for messages buffers).

If you use a 5 KB message buffer size, the minimum size is about 21 KB (11 KB base + 2 * 5 KB for message buffers).

However, not all Oracle SQL/Services applications use only two buffers. When a multi-tuple fetch or insert operation is initiated, you may get additional buffers for the client. How many additional buffers you get is based on the application. The dispatcher imposes a limit of 11 buffers that can be used at any one time.

A strategy for determining optimal shared memory size is as follows:

1. Pick a generous size for your shared memory based on the rough sizing method mentioned previously.
2. Run your system under normal load.
3. Occasionally issue a `SHOW SERVER` command from `SQLSRV_MANAGE` on the server that you are managing. It will show you the memory usage.
4. Adjust your shared memory size:

- Downward, if you see a constant number of free memory units.
- Upward, if you see no free memory units. You may also see client connections terminated by the server due to a lack of shared memory. This is reported in the log files. In certain rare situations, the entire server can fail due to insufficient shared memory.

As you add new users and applications to the server, review the shared memory usage.

2.4 Managing Server Components

Managing server components consists of managing the server, dispatchers, and services and performing tasks such as creating these objects, starting, shutting down, and restarting these objects, altering object attributes, and deleting these objects. [Section 2.4.1](#) through [Section 2.4.3](#) describe managing each of these objects.

2.4.1 Managing a Server

Managing a server involves knowing how to create a server; how to start, stop, and restart a server; and how to tailor the attributes of a server to suit the specific requirements of your client/server configuration.

Creating a Server

When you install Oracle SQL/Services, the installation procedure automatically creates and starts a server on that node. Unless you encounter a nonrecoverable error condition that renders the configuration file unusable, you normally will not have to create or re-create a server on a node on which you performed the Oracle SQL/Services installation. However, you should periodically save a backup copy of your configuration file. See [Copying a Configuration File](#) in this section for details of how to make a copy of a configuration file.

If your configuration file becomes corrupted, due perhaps to a disk failure, and you do not have a backup copy, you can delete the corrupted file and re-create your initial server configuration using the SYSSMANAGER:SQLSRV_CREATE71.COM command procedure.

In an OpenVMS cluster environment, the installation procedure creates and starts a server only for the node on which you perform the installation. If you plan to use Oracle SQL/Services on other nodes in the cluster, you must create and start a server on each of those nodes or make a single configuration file available to the other nodes, then start the server on those nodes.

There are two ways to create and start a server on other nodes in an OpenVMS cluster:

- Use the SYSSMANAGER:SQLSRV_CREATE71.COM procedure provided by the installation

The preferred method to create and start a server on another node in a cluster is to invoke the SYSSMANAGER:SQLSRV_CREATE71.COM DCL command procedure provided by the Oracle SQL/Services installation procedure (see the *Oracle SQL/Services Installation Guide* for more information). This procedure is used by the installation procedure itself and so will create and start a server that is identical to the one created on the node where the original installation was performed.

- Copy a configuration file from another node in the cluster

Another way to create a server on another node in a cluster is to copy a configuration file to that node, make any necessary changes for the node, then start the server on that node. This approach is more difficult because it can be error-prone, but nevertheless is an option. See [Copying a Configuration File](#) in this section for more information.

Alternatively, you may choose to share a single configuration file among multiple nodes in a cluster. The simplest way to make a single configuration file available to all nodes in a cluster is to shut down the server on the node on which you performed the installation, then rename the SQLSRV_CONFIG_FILE71.DAT file from the SYSSSPECIFIC:[SYSMGR] directory to the SYSSCOMMON:[SYSMGR] directory. If you choose to share a single configuration file among multiple nodes in a cluster, you must take care not to delete an object on one node if you intend to continue to use it on other nodes.

You do not need to perform additional tasks if you want to provide exactly the same dispatchers and services on each node in the cluster. However, if you need to support different network protocols or provide specific services on different nodes in the cluster, then you must tailor your configuration accordingly. To provide different dispatchers or services on different nodes, you must set the AUTOSTART attribute to OFF for any services and dispatchers that should not be started on all nodes, then write a SQLSRV_MANAGE script for each node that starts only the required dispatchers and services for that node. Note that you cannot configure a service or dispatcher object in a shared configuration file to have different attributes for different nodes.

Caution: Oracle recommends that you do not make offline modifications to a configuration file if there is a server running that is using the same file. In this situation, the SQLSRV_MANAGE utility does not prevent you from deleting a dispatcher or service object offline while the dispatcher or service is running.

Similarly, neither the SQLSRV_MANAGE utility nor the Oracle SQL/Services Manager GUI utility prevents you from deleting a dispatcher or service object online while the dispatcher or service is running on a different node in an environment where two or more nodes share the same configuration file. If this happens, then the SQLSRV_MANAGE utility displays a warning message if you show a dispatcher or service that has been deleted but that is still running.

The Oracle SQL/Services Manager GUI utility displays the delete icon (no-entry sign) for a dispatcher or service that has been deleted but is still running.

Starting, Shutting Down, and Restarting a Server

Oracle recommends that you add the Oracle SQL/Services startup command to the system startup file:

```
@SYS$STARTUP:SQLSRV$STARTUP71
```

Place the command after the Oracle Rdb startup command file RMONSTART. After you add this command, the server is started whenever the system boots.

Oracle also recommends that you add the shutdown command to the system shutdown command procedure:

```
@SYS$STARTUP:SQLSRV$SHUTDOWN71
```

Place the command before the Oracle Rdb shutdown procedure RMONSTOP. After you add this command, the server is stopped whenever the system is shut down.

Generally, the only time you will need to restart a server is if you alter a nondynamic attribute of the server object, in which case you must restart the server for the change to take effect.

Altering a Server

Once you create a server, you may need to alter some server attributes, such as the maximum amount of shared memory available to the server. [Table 2-1](#) lists all of the

attributes of a server, their default values, and indicates if an attribute can be modified dynamically. Following the table is a brief description of the major server attributes.

Table 2–1 Default Settings for Server Object Attributes

Attribute	Default Setting	Dynamic Attribute
MAX_SHARED_MEMORY_SIZE	2000 kilobytes	
Configuration File	SYSSMANAGER: SQLSRV_CONFIG_FILE71.DAT	
PROCESS_STARTUP_TIMEOUT	0	Yes
PROCESS_SHUTDOWN_TIMEOUT	0	Yes
Network ports	DECnet - SQLSRV_SERVER	
Network ports	TCP/IP - 2199	

Oracle SQL/Services uses shared memory for interprocess communications. The MAX_SHARED_MEMORY_SIZE attribute is the only server attribute you need to monitor on a periodic basis using the [SHOW SERVER Command](#). [Section 2.3](#) describes what to look for and when to make adjustments. The server uses network ports to listen to system management clients. These network ports must be unique in a multiversion environment because you can only have one version of Oracle SQL/Services using the default network ports. During a multiversion installation, you must specify what alternate network ports you want the server to use. You need not make any further changes to these network ports unless you decide to make the current version of Oracle SQL/Services the default, and you want to use the default system management network ports. If system management clients are having problems connecting, use the SHOW SERVER command to monitor these network ports and to ensure each is running.

See the [ALTER SERVER Command](#) for more information about altering server attributes.

If you alter a dynamic attribute of a running server online, the change takes effect immediately. However, if you alter a nondynamic attribute of a running server online, you must restart the server for the change to take effect.

If you alter a nondynamic attribute of a running server using the SQLSRV_MANAGE utility, it displays a success status indicating that you must restart the server for the change to take effect. For example:

```
SQLSRV> ALTER SERVER MAXIMUM_SHARED_MEMORY_SIZE 4000;
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect
SQLSRV>
```

The SQLSRV_MANAGE utility displays the values of any altered nondynamic attributes that will take effect when the server is restarted. For example:

```
** The Server will be updated as follows when it is restarted **  
    Max Shared memory size: 4000
```

If you alter a nondynamic attribute of a running server using the Oracle SQL/Services Manager GUI utility, it displays a success status indicating that you must restart the server for the change to take effect and inquires if you want to restart the server immediately. The Oracle SQL/Services Manager GUI utility highlights the values of any altered nondynamic attributes that will take effect when the server is restarted by displaying a flag-shaped icon next to each attribute.

When you restart a server, all dispatchers and services of the server are also restarted, and all client network connections to the server are disconnected. Therefore, you should schedule alterations to the server object when few or no clients will be using the server.

Copying a Configuration File

You can use the DCL COPY command to make a copy of a configuration file only if there is no running server using the file. To make a copy of a configuration file currently being used by a running server, you must use the DCL BACKUP/IGNORE=INTERLOCK command.

Deleting a Server

The only time you need to delete a server is when the configuration file has become corrupt, due perhaps to a disk failure, and is completely unusable. Deleting a server is an offline operation and deletes the configuration file (see the [DROP SERVER Command](#)). Alternatively, you can use the DCL DELETE command.

If you must delete a running server, first shut it down online (see the [SHUTDOWN SERVER Command](#)) and then delete it offline using the DROP SERVER command.

2.4.2 Managing a Dispatcher

Managing a dispatcher involves knowing how to create a dispatcher; how to start, stop, and restart a dispatcher; and how to tailor the attributes of a dispatcher to suit the specific requirements of your client/server configuration.

Creating a Dispatcher

The Oracle SQL/Services installation procedure and the SYSS\$MANAGER:SQLSRV_CREATE71.COM command procedure creates and starts three dispatchers named

SQLSRV_DISP for use by Oracle SQL/Services (ODBC for Oracle Rdb) clients, OCI_DISP for use by OCI clients, and RMU_DISP for use by Oracle RMU clients.

If you plan to use the Oracle Net network transport, then you will create another dispatcher after you decide which network ports you will use. You might also create other dispatchers if you decide to provide individual dispatcher processes for each transport available on your network. When you create a new dispatcher, you must ensure that the network ports that you specify are not used by any other dispatchers on the node. If a dispatcher is unable to listen on any of its network ports, it writes an error message to its log file and terminates.

Starting, Shutting Down, and Restarting a Dispatcher

Dispatchers that have the AUTOSTART attribute set to ON are automatically started when you install Oracle SQL/Services and whenever a server is started. If necessary, you can disable this action by starting a server with the START SERVER AUTOSTART OFF command. Dispatchers are automatically shut down when the server shuts down. One of the few times you must shut down a dispatcher is if it failed to start. A failed dispatcher is always left in a failed state. The reason for failure, which can be due to an incorrectly specified argument value in its definition, can be corrected using an ALTER DISPATCHER command. You can either shut down the dispatcher, make the correction, and start the dispatcher using the START DISPATCHER command, or you can make the change while the service is in a failed state, and then you must shut down and restart the dispatcher after you make the change.

Generally, the only time you will need to restart a dispatcher is if you alter a nondynamic attribute of a dispatcher object, in which case you have to restart the dispatcher for the change to take effect.

Altering a Dispatcher

As circumstances change, you may find it necessary to alter some dispatcher attributes. For example, to support additional users, you may need to increase the maximum number of connections allowed to a dispatcher. To provide better performance, you may want to increase the maximum client buffer size.

If you run multiple versions of the Oracle SQL/Services server, you may want to alter the network port specifications to use the default network ports when you stop using one version of Oracle SQL/Services.

Table 2–2 lists all of the attributes of a dispatcher and their default values, and indicates if an attribute can be modified dynamically. Following the table is a brief description of the major dispatcher attributes.

Table 2–2 Default Settings for Dispatcher Object Attributes

Attribute	Default Setting	Dynamic Attribute
AUTOSTART	ON	
MAX_CONNECTIONS	100	
IDLE_USER_TIMEOUT	0	Yes
MAX_CLIENT_BUFFER_SIZE	5000	
Log File	SYSS\$MANAGER:<dispatcher-name>.LOG	
Dump File	SYSS\$MANAGER:<dispatcher-name>.DMP	
Message Protocol	SQLSERVICES	
Network ports	DECnet - 81	
Network ports	TCP/IP - 118	
Network ports	IPX/SPX - 0x84b1	
Network ports	Oracle Net - no default, see listener.ora file for a list of listener objects you can use	

Set a higher value for the MAX_CONNECTIONS argument if you expect more than 100 clients to connect to the dispatcher at the same time.

Set a higher value for the MAX_CLIENT_BUFFER_SIZE argument if you know certain applications will benefit by using a larger buffer size.

The dispatcher uses network ports to listen to Oracle SQL/Services, Oracle ODBC Driver for Oracle Rdb, Oracle RMU, Oracle OCI, and Oracle Rdb JDBC clients. These network ports must be unique in a multiversion environment because you can have only one version of Oracle SQL/Services using the default network ports for a dispatcher on a node. During a multiversion installation, you must specify which alternate network ports you want the dispatcher to use. You need not make any further changes to these network ports unless you want to create one dispatcher listening exclusively on DECnet network ports and another dispatcher listening exclusively on TCP/IP network ports, and so forth, because of the network traffic. If clients are having problems connecting to dispatchers, use the SHOW DISPATCHER command to monitor these network ports and to ensure each is running.

See the [ALTER DISPATCHER Command](#) for more information about altering dispatcher attributes.

If you alter a dynamic attribute of a running dispatcher online, the change takes effect immediately. However, if you alter a nondynamic attribute of a running dispatcher online, you must restart the dispatcher for the change to take effect.

If you alter a nondynamic attribute of a running dispatcher using the `SQLSRV_MANAGE` utility, it displays a success status indicating that you must restart the dispatcher for the change to take effect. For example:

```
SQLSRV> ALTER DISPATCHER sqlsrv_disp MAX_CLIENT_BUFFER_SIZE 10000;
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect
```

The `SQLSRV_MANAGE` utility displays the values of any altered nondynamic attributes that will take effect when the dispatcher is restarted. For example:

```
** This Dispatcher will be updated as follows when it is restarted **
      Max client buffer size:    10000 bytes
```

If you alter a nondynamic attribute of a running dispatcher using the Oracle SQL/Services Manager GUI utility, it displays a success status indicating that you must restart the dispatcher for the change to take effect and inquires if you want to restart the dispatcher immediately. The Oracle SQL/Services Manager GUI utility highlights the values of any altered nondynamic attributes that will take effect when the dispatcher is restarted by displaying a flag-shaped icon next to each attribute.

When you restart a dispatcher, all client network connections to the dispatcher are disconnected. Therefore, you should schedule alterations to a dispatcher when few or no clients will be using the network ports managed by that dispatcher.

Deleting a Dispatcher

To delete a dispatcher as an online operation, you must first shut it down (see the [SHUTDOWN DISPATCHER Command](#) and the [DROP Command](#) in [Chapter 6](#)). The only time you want to delete a dispatcher is if it is no longer needed.

2.4.3 Managing a Service

Managing a service involves knowing how to create a service; how to start, stop, and restart a service; and how to tailor the attributes of a service to suit the specific requirements of your client/server configuration.

Creating a Service

The Oracle SQL/Services installation procedure and the `SYSS$MANAGER:SQLSRV_CREATE71.COM` command procedure create and start three services: a universal service named `GENERIC` for use by Oracle SQL/Services (ODBC for Oracle Rdb) clients, a

database service named OCI_SAMPLE for use by Oracle OCI clients, and an Oracle RMU service named RMU_SERVICE for use by Oracle RMU clients.

As the server administrator, you may need to create other services for different versions of Oracle Rdb. Similarly, you may want to create one or more database services for specific Oracle Rdb databases on your system. When you create a service, you must decide who will be authorized to access the service, how many executors will be needed to support clients who will use the service, and so forth. Also see [Section 2.6](#) for more information about universal and database services.

Starting, Shutting Down, and Restarting a Service

Services that have the AUTOSTART attribute set to ON are automatically started when you install Oracle SQL/Services and whenever a server is started. If necessary, you can disable this action by starting a server with the START SERVER AUTOSTART OFF command. Services are automatically shut down when the system shuts down. Usually, you set the AUTOSTART attribute to ON for most services you create so that they are available to clients all of the time.

However, you may decide to start certain services manually. For example, you may create a transaction reusable service for a particular database to determine if you can achieve better performance than using a session reusable service. In this situation, you might choose to set the AUTOSTART attribute to OFF while you test the new service.

One reason to shut down a service is if you must prevent clients from accessing the database provided by a service. For example, you would shut down a service while you restored a database after encountering a disk failure. Another reason you must shut down a service is if it failed to start. A failed service is always left in a failed state. The reason for failure, which can be due to an incorrectly specified argument value in its definition, can be corrected using an ALTER SERVICE command. You can either shut down the service, make the correction, and start the service using the START SERVICE command, or you can make the change while the service is in a failed state, and then you must shut down and restart the service after you make the change.

Generally, the only time you will need to restart a service is if you alter a nondynamic attribute of a service object, in which case you have to restart the service for the change to take effect.

Altering a Service

After you create a service, you may need to tune the performance of your system by adjusting the number of executors or the number of clients per executor for a service. If new users are added to the network, you may need to authorize access to a service to those users. If you upgrade a database to a later version of Oracle Rdb, you will need to alter a service to specify a new SQL version to be used by the executors of the service. [Table 2–3](#) lists all of

the attributes of a service, their default values, and indicates if an attribute can be modified dynamically. Following the table is a brief description of the major service attributes.

Table 2–3 Default Settings for Service Object Attributes

Attribute	Default Setting	Dynamic Attribute
AUTOSTART	ON	
DEFAULT_CONNECT_USERNAME	None	Yes
REUSE_SCOPE	SESSION	
SQL_VERSION	STANDARD	
PROTOCOL	SQL/Services	
PROCESS_INITIALIZATION	None	
ATTACH	None	
OWNER	None	
SCHEMA	None	
SQL_INIT_FILE	None	
DATABASE_AUTHORIZATION	CONNECT USERNAME	
APPLICATION_TRANSACTION_USAGE	SERIAL	Yes
IDLE_USER_TIMEOUT	0	Yes
IDLE_EXECUTOR_TIMEOUT	1800	Yes
MIN_EXECUTORS	0	Yes
MAX_EXECUTORS	1	Yes
CLIENTS_PER_EXECUTOR	1	Yes

Create as many service objects as you need to accommodate the databases accessed by applications that your user community intends to run. See [Section 2.6](#), [Section 2.8](#), and [Section 2.9](#) for more information.

Set the MIN_EXECUTORS, MAX_EXECUTORS, and IDLE_EXECUTOR_TIMEOUT attributes for each service based on user activity over time to provide efficient services to your clients. See [Section 2.11](#) for more information.

You may need to adjust the CLIENTS_PER_EXECUTOR attribute value to attain the best performance when tuning a transaction reusable service.

Giving users or identifiers access to services or modifying their current access is another task you need to perform on a continual basis. Use the GRANT USE ON SERVICE and

REVOKE USE ON SERVICE commands to perform these tasks. Use the SHOW SERVICE command to determine the users or identifiers who currently have access to a particular service.

See the [ALTER SERVICE Command](#) for more information about altering service attributes.

If you alter a dynamic attribute of a running service online, the change takes effect immediately. However, if you alter a nondynamic attribute of a running service online, you must restart the service for the change to take effect.

If you alter a nondynamic attribute of a running service using the SQLSRV_MANAGE utility, it displays a success status indicating that you must restart the service for the change to take effect. For example:

```
SQLSRV> ALTER SERVICE payroll SQL_INIT_FILE PAYROLL_DIR:PAYROLL.SQLINIT;  
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect
```

The SQLSRV_MANAGE utility displays the values of any altered nondynamic attributes that will take effect when the dispatcher is restarted. For example:

```
** This Service will be updated as follows when it is restarted **  
   SQL init file:           payroll_dir:payroll.sqlinit
```

If you alter a nondynamic attribute of a running service using the Oracle SQL/Services Manager GUI utility, it displays a success status indicating that you must restart the service for the change to take effect and inquires if you want to restart the service immediately. The Oracle SQL/Services Manager GUI utility highlights the values of any altered nondynamic attributes that will take effect when the service is restarted by displaying a flag-shaped icon next to each attribute.

When you restart a service, all client network connections from applications using the service are disconnected. Therefore, you should schedule alterations to a service when few or no clients will be using the service.

Deleting a Service

The only time you may need to delete a universal service is when there are no more databases for that specific version of Oracle Rdb in use. Similarly, you may want to delete a database service if it is no longer used or if there are too few users using it to justify this type of service. In either case, to delete the service online, you must first shut it down (see the [SHUTDOWN SERVICE Command](#) and the [DROP Command](#) in [Chapter 6](#)).

2.5 Setting Up Dispatchers and Transport Selection

A client communicates with the server and dispatcher by using a network transport supported on your system. The Oracle SQL/Services release 7.1 and later server supports the TCP/IP, DECnet, Oracle Net, and NetWare (IPX/SPX) transports.

When you create a dispatcher object, you can specify whether you want the dispatcher to support one or more transports. If you want a dispatcher to support only one transport, you must create additional dispatchers to support each of the transports that your Oracle SQL/Services clients use. For Oracle SQL/Services, you can use one or more dispatchers for each server configuration. Each dispatcher defined must be listening on one or more unique network port IDs or objects.

The following example illustrates how to create a dispatcher that supports the Oracle Net transport:

```
SQLSRV> CONNECT SERVER;
Connecting to server ...
Connected
SQLSRV> CREATE DISPATCHER sqlnet_disp NETWORK_PORT SQLNET LISTENER_NAME 'LISTENER';
SQLSRV> START DISPATCHER sqlnet_disp;
```

See the *Guide to Using the Oracle SQL/Services Client API* for more information on using the Oracle Net transport.

The following example illustrates how to shut down and delete a dispatcher that supports two transports, and create two other dispatchers, each supporting just a single transport. First, ensure that no clients are using any transports supported by the dispatcher that you plan to delete. Shutting down a dispatcher will disconnect the network connections from any clients that are using the dispatcher. If no clients are using the dispatcher, then shut down and delete the dispatcher. Finally, create and start the new dispatchers.

```
SQLSRV> CONNECT SERVER;
Connecting to server ...
Connected
SQLSRV> SHOW CLIENTS;
```

Service: SQLSRV_MANAGE

Connect	Client		Executor
Username	Node	State	PID
Application			
root	127.0.0.1	RUNNING BOUND	00000ec3

```
SQLSRV_MANAGE
SQLSRV> SHUTDOWN DISPATCHER sqlsrv_disp;
SQLSRV> DROP DISPATCHER sqlsrv_disp;
```

```
SQLSRV> CREATE DISPATCHER sqlsrv_tcpip NETWORK_PORT TCPIP;  
SQLSRV> CREATE DISPATCHER sqlsrv_decnet NETWORK_PORT DECNET;  
SQLSRV> START DISPATCHER sqlsrv_tcpip;  
SQLSRV> START DISPATCHER sqlsrv_decnet;
```

Note: In order to use OCI Services for Oracle Rdb, you must define LISTENER_NAME in the LISTENER.ORA file. Refer to Chapter 4 in *Oracle SQL/Services Installation Guide Release 7.1.6* for more information.

2.6 Setting Up Services and Types of Reuse

The Oracle SQL/Services server provides universal services and database services. Unless otherwise specified, the information in this section applies to Oracle SQL/Services and OCI Services for Oracle Rdb protocols. For information about RMU Services, refer to the RMU Backup command in the *Oracle RMU Reference Manual*.

Universal Service

A universal service allows a client application to determine which database is to be accessed. An executor process for a universal service, therefore, is not preattached to a specific database. Each time a client application connects to a universal service, it must issue one or more database attach statements before performing any data access operations.

You can use universal services with Oracle Rdb to provide access to local and remote Oracle Rdb databases.

Database Service

A database service allows a client application to access data within a specific database. An executor process for a database service is preattached to a single database. When a client connects to a database service, it can immediately begin to access data in the preattached database.

You can use database services with Oracle Rdb to provide access to local and remote databases with the restriction that you must set the database authorization attribute to the service owner to access remote databases (see [Section 2.7](#) for more information).

If an executor process provides service to local or remote databases using Oracle Rdb release 6.1 and later, the following SQL statements cannot be prepared:

- ATTACH

- DECLARE DATABASE
- CREATE DATABASE
- ALTER DATABASE
- DROP DATABASE
- CONNECT
- SET CONNECT
- DISCONNECT

Note: A client connected to a database service can access data only from the preattached database; it cannot access data from any other database.

Types of Reuse

The Oracle SQL/Services server provides services that have either a session reuse or transaction reuse attribute.

Session Reuse An executor for a session reusable service processes requests for one client session at a time. A session begins when a client connects to the service using either a `sqlsrv_associate` call, an ODBC connect function, or an OCI connect, and the connection is bound to an executor process. A session ends when a client disconnects from the service and the connection is unbound from the executor process. A session reusable service is so named because an executor does not begin to process a new session until the current session ends. The session reuse attribute may be applied to either universal or database services; this attribute is the only one that may be applied to a universal service. See [Section 2.6.1](#) and [Section 2.6.2](#) for more information.

Transaction Reuse An executor for a transaction reusable service processes requests for one transaction for a client at a time; however, the executor is shared by many concurrent client sessions. A transaction begins when a client issues a SQL statement that either implicitly or explicitly starts a transaction. A transaction ends when the client issues a successful SQL COMMIT or ROLLBACK statement or executes a stored procedure that commits or rolls back a transaction. A transaction reusable service is so named because an executor does not begin to process a new transaction until the current transaction ends. The transaction reuse attribute may be applied only to database services. See [Section 2.6.3](#) for more information.

Note: Transaction reusable database services are not supported for OCI Services for Oracle Rdb.

Table 2–4 summarizes the attributes and settings associated with each service.

Table 2–4 Oracle SQL/Services Service Attributes

Service Definition Attribute	Service		
	Session Reusable Universal	Session Reusable Database	Transaction Reusable Database
Prestarted	Yes	Yes	Yes
Preattached	No	Yes	Yes
Execute ATTACH statement	Yes	No	No
Execute multiple attachments	Yes	No	No
Number of clients per executor	1	1	>1

2.6.1 Session Reusable Universal Services

An executor for a session reusable universal service processes requests for a single client session at one time and is not preattached to a specific database.

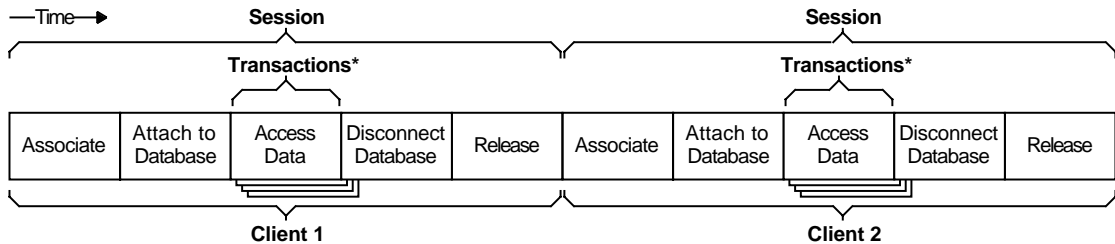
You use a universal service when one of the following conditions apply:

- you want to allow client applications to determine which database to use
- you have a node with a large number of infrequently accessed databases for which it would be impractical to provide individual database services
- you have legacy or third-party applications that can select the database to be used only by connecting to a universal service and executing a SQL ATTACH statement
- the application developers need full control over the database they are using

Executor processes for universal services may be prestarted or started on demand. By prestarting a sufficient number of executor processes for a universal service, you enable clients to avoid the process startup delay when they connect to the service. Clients will always incur the overhead of attaching to the required database when using a universal service.

Figure 2–1 illustrates how a universal service works. When a client connects to a universal service, the client connection is assigned and bound to an executor process. Once bound, the client attaches to one or more databases, accesses data, and finally disconnects from any attached databases. When the client releases the connection, the executor process unbinds from the client connection. The executor process is then available for use by another client.

Figure 2–1 Oracle SQL/Services Session Reusable Universal Services



* can be single or multiple

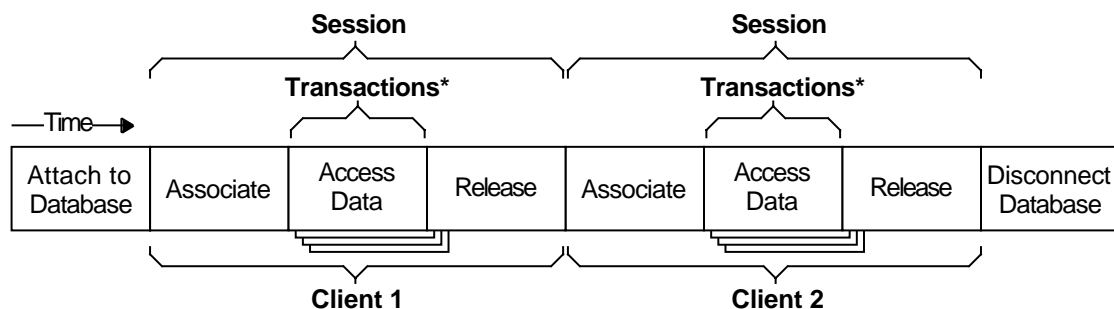
2.6.2 Session Reusable Database Services

An executor for a session reusable database service processes requests for a single client session at one time and is preattached to a single database.

You use a session reusable database service when you want to provide clients with a service that accesses a specific database whose transactions are of long or unknown duration. Executor processes for session reusable database services may be prestarted or started on demand. By prestarting a sufficient number of executor processes for a session reusable database service, you enable clients to avoid process startup and database attach delays when they connect to the service.

Figure 2–2 illustrates how a session reusable database service works. When a client connects to a session reusable database service, the client connection is assigned and bound to an executor process. Once bound, because the executor is preattached to a database, the client can immediately access data in the database. When the client releases the connection, the executor process unbinds from the client connection. The executor process is then available for use by another client.

Figure 2–2 Oracle SQL/Services Session Reusable Database Services



* can be single or multiple

2.6.3 Transaction Reusable Database Services

Note: Transaction reusable database services are not supported for OCI Services for Oracle Rdb.

An executor for a transaction reusable database service is preattached to a single database, processes requests for the transaction of one client at one time, and is shared by many concurrent client sessions. Once assigned to a particular executor process, a specific client connection remains assigned to that executor process until the client application disconnects from the service.

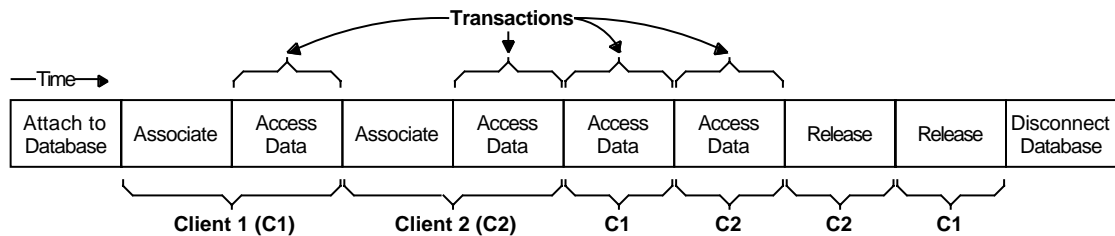
You use a transaction reusable database service to provide clients with a service that accesses a specific database where the database workload consists of transactions of known, relatively short duration. When used in the appropriate situations, transaction reusable database services can improve performance by reducing system resource usage and database contention. This is because multiple clients share a single executor process, thus reducing the total number of executor processes required on the system.

Transaction reusable database services are not well suited to situations where transactions are of long or varying duration. If transaction reusable database services are employed in such a situation, users will tend to experience unpredictable response times because a client executing a long transaction will tie up an executor process, making it unavailable for other users.

Executor processes for transaction reusable database services are always prestarted so that the server can distribute client connections evenly across the set of executor processes started for the service. Because multiple client connections share a single transaction reusable executor process, you need not prestart as many executor processes as when using session reusable executors. Fewer executor processes, with a high number of clients per executor, are required when the workload consists of very short transactions. More executor processes, with a lower number of clients per executor, are required as the transaction duration increases.

Figure 2–3 illustrates how a transaction reusable database service works. When a client connects to a transaction reusable database service, the client connection is assigned to an executor process; however, the client connection does not stay bound to the executor process after the executor has processed the initial connection. Multiple client connections may be assigned to a single executor process. A client connection is bound to an executor process when a transaction is started, at which time the client accesses data in the database. When the client ends the transaction, by using a SQL COMMIT or ROLLBACK statement or by executing a stored procedure that commits or rolls back a transaction, the executor process unbinds from the client connection. The executor process then becomes available for use by another client connection. When the client releases the connection, the client connection is deassigned from the executor process.

Figure 2–3 Oracle SQL/Services Transaction Reusable Database Services



2.6.4 When to Use Session Reusable Versus Transaction Reusable Database Services

Note: This section does not apply to OCI Services for Oracle Rdb because transaction reusable database services are not available for that protocol.

Table 2–5 summarizes the factors to consider in deciding whether to use session reusable database services or transaction reusable database services.

Table 2–5 When to Use Session Reusable Versus Transaction Reusable Database Services

Attribute	Database Service	
	Session Reusable	Transaction Reusable
If client transactions are:	Long duration Unknown length	Short duration Known length
If service use frequency is:	Infrequent Set no. executors: Min=0 Max=high value	Set number of executors: Min=Max (required)
	Frequent Set no. executors: Min=Max	Set number of executors: Min=Max (required)
If number of clients per executor is:	1 (required)	>1 If short transactions, set to a higher number. If longer transactions, set to a lower number.

2.7 Setting Database Access Authorization

For OCI Services for Oracle Rdb, database access authorization must be the connect user name.

For the SQL/Services protocol, the following guidelines can help you understand and decide what type of service to provide to clients and whether or not to set database access authorization to the connect user name or the service owner.

Universal Services

Database Access Authorization Set to Connect User Name For clients using a universal service, set database access authorization to the connect user name if you want client applications to attach to and access databases by using the client-supplied user name, the DECnet proxy user name, or the default connect user name. With database access authorization set to the connect user name, client access to databases is based on the use granted to individual users or groups of users using the underlying database security mechanisms.

[Example 2–1](#) illustrates how to create the universal service named `GENERIC`. Note that `GENERIC` is the service name that an Oracle SQL/Services or Oracle ODBC Driver for Oracle Rdb client will use by default if no service name is supplied. This universal service has database access authorization set to the connect user name, access granted to all users, a minimum of 1 executor process, and a maximum of 20 executor processes.

Example 2–1 Default Universal Service with Database Access Authorization Set to Connect User Name

```
SQLSRV> CREATE SERVICE GENERIC
_SQLSRV>     OWNER 'SQLSRV$DEFLT'
_SQLSRV>     DATABASE_AUTHORIZATION CONNECT USERNAME
_SQLSRV>     MIN_EXECUTORS 1
_SQLSRV>     MAX_EXECUTORS 20;
SQLSRV> GRANT USE ON SERVICE GENERIC TO PUBLIC;
SQLSRV> START SERVICE GENERIC;
```

Database Access Authorization Set to Service Owner For clients using a universal service, set database access authorization to the service owner only if you need client applications to attach to and access databases by using a single, fixed user name, the service owner user name. You can use the `GRANT USE ON SERVICE` command to restrict the users that can access such a service.

Caution: If you set database access authorization to the service owner for a universal service, be sure that the service owner user name does not have access to any databases containing secure or sensitive data that would otherwise be protected against access from unauthorized users.

Usually, you will *not* set database authorization to service owner for a universal service.

[Example 2–2](#) illustrates how to create a universal service that might be used for testing purposes that has database access authorization set to the service owner. Authorization to use the service is granted to only two development accounts in addition to the service owner user name account.

Example 2–2 Universal Service with Database Access Authorization Set to Service Owner

```
SQLSRV> CREATE SERVICE GEN_DEVEL OWNER 'noprivs'
_SQLSRV>     DATABASE_AUTHORIZATION SERVICE OWNER
_SQLSRV>     MIN_EXECUTORS 0
_SQLSRV>     MAX_EXECUTORS 5;
```

```
SQLSRV> GRANT USE ON SERVICE GEN_DEVEL TO 'develop', 'test';
SQLSRV> START SERVICE GEN_DEVEL;
```

Database Services

Database Access Authorization Set to Connect User Name For clients using a database service, set database access authorization to the connect user name if you want clients to access the database by using the client-supplied user name, the DECnet proxy user name, or the default connect user name. With database access authorization set to the connect user name, client access to the database is based on the use granted to individual users or groups of users using the underlying database security mechanisms.

[Example 2–3](#) illustrates how to create a database service to access the policies and procedures database of a company where the database is accessed under the client's user name. Access to the service is granted to all users, while access to data in the database is based on the underlying database security mechanisms. Unknown users are authorized to use the service under the default connect user name 'readpp', which has read-only access to data in the database. The service is owned by the 'ppdb' account, which will be used to attach to the database when an executor process is started.

Example 2–3 Session Reusable Database Service with Access Authorization Set to Connect User Name

```
SQLSRV> CREATE SERVICE P_AND_P
_SQLSRV>           ATTACH 'FILENAME pp_disk:[pp]pp_database'
_SQLSRV>           OWNER ppdb
_SQLSRV>           DATABASE_AUTHORIZATION CONNECT USERNAME
_SQLSRV>           DEFAULT_CONNECT_USERNAME readpp
_SQLSRV>           MIN_EXECUTORS 0
_SQLSRV>           MAX_EXECUTORS 10;
SQLSRV> GRANT USE ON SERVICE P_AND_P TO PUBLIC;
SQLSRV> START SERVICE P_AND_P;
```

Database Access Authorization Set to Service Owner For clients using a database service, set database access authorization to service owner if you want client applications to access the database by using the service owner user name. Use this approach when you want to grant access to specific data within the database and to specific database operations to a single user name by using the underlying database security mechanisms, and then grant use of the service to a restricted set of user names by using the GRANT USE ON SERVICE command.

[Example 2–4](#) illustrates how to create a database service to access the order-entry database of a company where the database is accessed under the service owner user name, 'ordent'.

Access to the service is granted only to the 'ordent1', 'ordent2', 'ordent3', and 'ordmgr' users, in addition to the service owner and privileged users with SYSPRV privilege. The database name OE_DISK:[OE]OE_DATABASE is defined as a logical name 'oe_database', so the database can be physically moved if necessary without having to modify the service definition.

The transaction workload characteristics of the database allow the service to be transaction reusable, support up to 100 users distributed over five executor processes, and have up to 20 users per process.

Example 2–4 Transaction Reusable Database Service with Access Authorization Set to Service Owner

```
SQLSRV> CREATE SERVICE ORD_ENT REUSE SCOPE IS TRANSACTION
_SQLSRV> ATTACH 'FILENAME OE_DATABASE'
_SQLSRV> OWNER ordent
_SQLSRV> DATABASE_AUTHORIZATION SERVICE OWNER
_SQLSRV> MIN_EXECUTORS 5
_SQLSRV> MAX_EXECUTORS 5
_SQLSRV> CLIENTS_PER_EXECUTOR 20;
SQLSRV> GRANT USE ON SERVICE ORD_ENT TO ordent1,
_SQLSRV> ordent2,
_SQLSRV> ordent3,
_SQLSRV> ordmgr;
SQLSRV> START SERVICE ORD_ENT;
```

2.7.1 Specify a Default Connect User Name for the SQL/Services Protocol

The following guidelines can help you decide whether or not to specify the default connect user name to authorize unknown users' access to databases on your system through either a universal service or a database service.

Using Universal Services

Specify a default connect user name for a universal service only if you need to allow unknown users access to databases on your system. You may choose this approach in a development environment to allow simple access to databases used for testing and debugging.

Caution: If you specify a default connect user name to authorize use of a universal service to unknown users, ensure that any databases containing secure or sensitive data are protected with the appropriate access restrictions at the database level.

Usually, you will not specify a default connect user name to authorize use of a universal service to unknown users.

Using Database Services

Specify a default connect user name to authorize use of a database service to unknown users if you want to allow access to data in a particular database without requiring a user name and password. For example, you may consider providing access to nonsensitive, public-access data in a database by using this mechanism in combination with database access authorization set to the connect user name (see [Example 2–3](#)).

2.7.2 Grant or Restrict Access to a Service

The following guidelines can help you decide whether to grant access to a service to all users or restrict access to a service to a specified list of users.

Grant Access to a Service to All Users If:

- You have universal or database services where database access authorization is set to the connect user name and you want to provide all users with the most flexible method of access to data in databases on your system, subject to underlying database security in individual databases.
- You have a database service where database access authorization is set to the service owner, but access to the database by using the service owner user name is restricted to nonsensitive, public-access data that you want to make available to all users.

Restrict Access to a Service to a Specified List of Users If:

- You have a universal service with database access authorization set to the service owner in order to access a set of databases using a fixed user name.
- You have a database service with database access authorization set to the service owner where you want to grant access to data in a database to a single user name by using the underlying database security mechanisms, and then control access to that data by using Oracle SQL/Services security mechanisms.

Caution: Restricting access to services to a specified list of user names by using Oracle SQL/Services does not prevent other users from trying to log in to your system and attempting to access the same databases (using a tool such as interactive SQL) provided by those services. Even if you restrict access to a service to a specified list of user names, you should still protect secure and sensitive data in databases by using underlying database security mechanisms.

2.7.3 Provide Arbitrary or Predefined Access to Data

The following guidelines can help you decide whether to provide arbitrary access to data or predefined access to data.

Arbitrary Access to Data

You restrict the tables that users can access and the operations that they can perform on those tables by using either underlying database security mechanisms alone or in combination with Oracle SQL/Services security mechanisms. However, once access to data has been granted, users can then execute arbitrary SQL statements against that data, subject to the access they have been granted. For example, if users have INSERT access to a table, they can insert any data they wish into that table. In some situations, allowing arbitrary access to data in a database may not be desirable.

Predefined Access to Data

In some situations, it is desirable to restrict users' ability to manipulate data to a set of predefined operations. You do this as part of the database setup by creating a set of definer's rights stored procedures. The procedures provide all of the necessary access to data in one or more tables. By restricting access to the tables to the user name of the stored procedures' definer, you prevent access from all other users. You then grant access to the stored procedures by using either underlying database security mechanisms alone or in combination with Oracle SQL/Services security mechanisms.

2.8 Setting Up Security on Servers

Oracle SQL/Services, in combination with the underlying database engine, provides various security mechanisms that you can employ to control the services and data that users are allowed to access. An Oracle SQL/Services server environment can be viewed as having three tiers where security is checked. The tiers are as follows:

- **Tier 1: Client identification and authentication**

The Oracle SQL/Services server first checks the identification and authentication of users requesting access to the server. This occurs when the client first connects to the server.

- **Tier 2: Service access authorization**

The Oracle SQL/Services server next checks that each user requesting access to a particular service has been authorized to use that service.

- **Tier 3: Database and data access authorization**

Finally, the underlying database engine checks each database access request made by an executor process.

Each of these security tiers is discussed in the sections that follow.

2.8.1 Client Identification and Authentication

The first server security tier is client identification and authentication. This occurs when the client application first connects to the server. The result of the successful completion of the first tier is a connect user name that is used for authorization checks in subsequent tiers.

Verification of user name and password are accomplished in one of the following ways:

- **OCI Services for Oracle Rdb**

For OCI Services for Oracle Rdb, the client supplies the user name which must be located in the USERS\$ table in the database. The USERS\$ table also contains the encrypted password. The password is returned to the client which returns the key to decrypt the password. User name and password are then given to Oracle SQL/Services for authorization. Access to an OCI service must be by connect user name.

- **Oracle SQL/Services**

For Oracle SQL/Services, a user supplies a user name and password when accessing a service. When a client connects to an Oracle SQL/Services server, the server ensures that the user's account exists on the system and that the password is valid. Following successful authentication, the client-supplied user name is used as the connect user name. If the user name and password check fails or if the password has expired, then the connection is rejected and an error message is returned to the client. If the user does not supply a user name and password, the server then checks the network transport of the connection.

If the client selected the DECnet transport, then the server checks to see if a proxy exists for the node name or user name of the client or both. The server first looks up the client's DECnet node name and DECnet user name, if any, in the Oracle SQL/Services proxy file, `SYSS$STARTUP:SQLSRV$PROXY.DAT`. If a match is found, then the local proxy user name is used as the connect user name. If no match is found, but the client is on the same node as the server, then the user name of the client process is used as the connect user name.

For the system management client only, the server uses the user name of the client process as the connect user name, if the user:

- Selected the TCP/IP transport
- Logged in to the server node
- Has `SYSPRV` or `BYPASS` privileges

As a system administrator, you can choose to allow access to a service without requiring a user name and password by specifying a default connect user name. If the user does not supply a user name and password and a default connect user name is not specified (unknown users are not authorized to access the service), then the connection is rejected and an error message is returned to the client. If the client does not supply a user name and password and a default connect user name has been specified (unknown users are authorized to access the service), then the connect user name is set to the default connect user name. If the client does supply a user name, then the user name is used as the connect user name, regardless of whether or not a default connect user name is specified.

When a system administrator connects to a system management service of a server, the server performs the same user name and password check as when a client connects to a service. If the user name and password checks fail, then the connection is rejected and an error message is returned to the system management application. You cannot specify a default connect user name for the system management service; therefore, you cannot authorize unknown users to access the system management service.

2.8.2 Service Access Authorization

The second server security tier verifies that the user is authorized to access the selected service.

Each service has a list of user names and identifiers that are authorized to access the service. When you create a new service, only the service owner is authorized to access the service. As a system administrator, you are responsible for granting appropriate users access to services provided by the server. You can grant access to a service based on an individual

user name, an identifier, or you can grant access to a service to all users (for example, GRANT USE ON SERVICE GENERIC TO PUBLIC).

When a user connects to a service, the server checks to see if the connect user name or an identifier held by the connect user name has been authorized to use the service, or if access to use the service has been granted to all users. If the user is not authorized to access the service, then the connection is rejected and an error message is returned to the client.

A system management service of a server also has a list of user names and identifiers that are authorized to access the service and thus manage the server. When you create a server, typically done as part of the installation, only the privileged user with SYSPRV privilege is authorized to manage the server. As a system administrator, you are responsible for granting access to any additional users who will manage the server. If an unauthorized user attempts to connect to a system management service of a server, then the connection is rejected and an error message is returned to the system management application.

2.8.3 Database and Data Access Authorization

The third and final server security tier occurs at the database level in an executor process. Whenever an executor process executes a SQL statement, the underlying database engine performs a security check to determine if the user name executing the request is authorized to do so. Oracle SQL/Services allows database requests to be executed using either the connect user name or the service owner, depending on the type of service you are providing and the version of Oracle Rdb specified for the service. As a system administrator, you determine which user name is authorized by the database engine by specifying the database access authorization attribute of each service to be either the connect user name or service owner.

- Database access authorization set to connect user name

If you set the database access authorization to connect user name, then the underlying database uses the connect user name to determine if a client is authorized to execute a database request. The connect user name is the client-specified user name, a DECnet proxy user name, or the default connect user name.

Note: You cannot use database services with database access authorization set to the connect user name to provide access to remote Oracle Rdb databases. However, Oracle SQL/Services does allow you to provide a database service for a remote Oracle Rdb database if you create the service with database access authorization set to the service owner.

- Database access authorization set to service owner

If you set the database access authorization to service owner, then the underlying database uses the service owner's user name to determine if a client is authorized to execute a database request. The service owner account must have SELECT access to the database to which you are attaching. The executor process inherits the OpenVMS privileges from the service owner userid. NETMBX and TMPMBX are all that is needed as AUTHORIZED privileges. Oracle SQL/Services uses the privileges list instead of the DEFAULT privilege list.

2.8.4 How Server Security Tiers Work Together for the SQL/Services Protocol

Figure 2-4 illustrates how the three server security tiers work together for three connect examples in which a client logs in to the system. Each example shows client identification and authentication, service access authorization, and the resulting database and data access authorization based on the service definition for each service.

Figure 2–4 Oracle SQL/Services Server Security

	Connect 1	Connect 2	Connect 3
Service Definition	Service name=X Owner='fred' Attach='payroll_db' Database authorization= service owner No default connect username argument Grant use to 'freda', 'ned'	Service name=Y Owner='bert' No attach argument Database authorization= connect username No default connect username argument Grant use to PUBLIC	Service name=Z Owner='joe' Attach='account_db' Database authorization= connect username Default connect username='jane' Grant use to 'janet', 'jane'
Client Connect to Server	User name='ned' Password='pwned' Service name=X	User name='holly' Password='pwholly' Service name=Y	No user name No password Service name=Z
Tier 1: Client Identification and Authentication	User name='ned' Authenticated using password 'pwned' Connect user name set to 'ned'	User name='holly' Authenticated using password 'pwholly' Connect user name set to 'holly'	Default connect username='jane' Connect user name set to 'jane'
Tier 2: Service Access Authorization	Connect user name 'ned' authorized access to service X using 'ned'	Connect user name 'holly' authorized access to service Y using PUBLIC	Connect user name 'jane' authorized access to service Z using 'jane'
Tier 3: Database and Data Access Authorization	Database attached using service owner user name 'fred' Data accessed using service owner user name 'fred'	Database attached using connect user name 'holly' Data accessed using connect user name 'holly'	Database attached using service owner user name 'joe' Data accessed using connect user name 'jane'

First Connect Example

In the first connect example, a user requests access to service X. The user specifies a user name and a password, so these are authenticated by the server in the first security tier and

the connect user name is set to 'ned'. No default connect user name is specified, so unknown users are not allowed to access service X; therefore, all users requesting access to service X must supply a valid user name and a password. In the second security tier, the server checks that the connect user name, 'ned' in this example, is authorized to access the service. Users 'freda' and 'ned', as well as a privileged user with SYSPRV privilege, have been granted the right to use service X. So user 'ned' is authorized to access the service. Service X is a database service; therefore, the executor process attaches to the 'payroll_db' database by using the service owner's user name, 'fred'. Database access authorization for service X is set to the service owner, so all database attachments and data access requests are also made under the service owner's user name, 'fred' in this example.

Second Connect Example

In the second connect example, a user requests access to service Y. The user specifies a user name and a password, so these are authenticated by the server in the first security tier and the connect user name is set to 'holly'. No default connect user name is specified, so unknown users are not allowed to access service Y; therefore, all users requesting access to service Y must supply a valid user name and password. In the second security tier, the server checks that the connect user name, 'holly' in this example, is authorized to access the service. All users have been granted the right to use service Y, so user 'holly' is authorized to access the service. Service Y is a universal service; therefore, an executor is not preattached to a specific database. Database access authorization for service Y is set to the connect user name, so all database attachments and data access requests are made under the connect user name, 'holly' in this example.

Third Connect Example

In the third connect example, a user requests access to service Z. The user does not specify a user name and a password, so the server checks if unknown users are authorized to access the requested service. A default connect user name is specified, so unknown users are allowed to access service Z as user 'jane'; therefore, the connect user name is set to 'jane'. In the second security tier, the server checks that the connect user name, 'jane' in this example, is authorized to access the service. Users 'janet' and 'jane', as well as a privileged user with SYSPRV privilege have been granted the right to use service Z, so user 'jane' is authorized to access the service. Service Z is a database service; therefore, the executor process attaches to the 'account_db' database by using the service owner's user name, 'joe'. However, database access authorization for service Z is set to the connect user name, so all data access requests are made under the connect user name, 'jane' in this example.

2.9 Understanding Database Access Authorization Models for Oracle SQL/Services

In [Section 2.7](#) you learned that Oracle SQL/Services allows you to authorize database access using the service owner user name or the connect user name. You also learned how these models affect the environment within which database requests and external functions are executed. This section describes in detail how Oracle SQL/Services implements database authorization by connect user and by service owner.

2.9.1 Accessing an Oracle Rdb Database

To understand how Oracle SQL/Services implements database authorization by connect user name and by service owner, it is first necessary to understand that four user names are involved in accessing an Oracle Rdb database in the Oracle SQL/Services environment:

- Operating system process user name
- Oracle Rdb system user name
- Oracle Rdb session user name
- Oracle Rdb current user name

Note: Access to an OCI service must be by connect user name.

Following is an explanation of the four user names.

2.9.1.1 Operating System Process User Name

The process user name is the user name under which an Oracle SQL/Services executor process runs a local attach, or the user name of the Oracle Rdb remote server process in a remote attach.

The process user name is set based on the SERVICE OWNER service attribute for local attaches, whereas it is based on the ATTACH statement and the configuration of the remote Oracle Rdb server node for remote attaches. Associated with the process user name are a number of process attributes. These attributes include:

- UIC
- Privileges
- Rights list

- Account name
- Default directory
- Logical names, including
 - SYS\$DISK
 - SYS\$LOGIN_DEVICE
 - SYS\$LOGIN
 - SYS\$SCRATCH
 - LNMS\$GROUP (for group logical name table)

2.9.1.2 Oracle Rdb System User Name

Each attached database in an executor process has a value for the system user name. The Oracle Rdb system user name is used to determine if the process is authorized to attach to the database and also serves as the default value for the Oracle Rdb session user name.

The Oracle Rdb system user name for an attached database defaults to the process user name but may be overridden by the SQL ATTACH statement attribute of a database service or by a client application accessing a universal service, depending on the type of service being provided, the attributes of that service, and the version of Oracle Rdb being used.

The Oracle Rdb system user name for an attached database is established at the time of attachment to the database and remains fixed for the life of the attachment. You can override the default value for the system user name when using Oracle Rdb release 6.1 or later by specifying a user name and a password in the attach-string argument of a SQL ATTACH statement or in the connect-string argument of a SQL CONNECT statement. See the *Oracle Rdb SQL Reference Manual* for more information on the SQL ATTACH and CONNECT statements.

The number of attached databases in an executor process providing a universal service is determined by the client application. Different attached databases may have different system user names.

An executor process providing a database service has only one attached database.

2.9.1.3 Oracle Rdb Session User Name

All database requests are executed within the context of a SQL connect. Each SQL connect in an executor process has a value for the session user name. The session user name for a SQL connect defaults to the Oracle Rdb system user name, but may be overridden by Oracle

SQL/Services or by a client application, depending on the type of service being provided, the attributes of that service, and the version of Oracle Rdb being used.

The session user name for a SQL connect is determined at the time the SQL connect is established and remains fixed for the life of the SQL connect. You can override the default value for the session user name when using a universal service with Oracle Rdb release 6.1 or later by specifying a user name and a password as arguments to the SQL CONNECT statement. See the *Oracle Rdb SQL Reference Manual* for more information on the SQL CONNECT statement.

The number of SQL connects in an executor process providing a universal service is determined by the client application. Different SQL connects may have different session user names. A SQL connect in an executor process providing a universal service can reference one or more database attaches.

The number of SQL connects in an executor process providing a database service is determined by the version of Oracle Rdb in use and the service reuse attribute. See [Section 2.6](#) for information on reuse attributes.

The ATTACH statement of a database service is always executed in the context of the default SQL connect. You cannot use a SQL CONNECT statement to attach to a database using a database service. The session user name for the default connect defaults to the system user name. If a SQL initialization file is specified for the service, then the statements contained therein are executed in the context of the default SQL connect after the SQL ATTACH statement.

For Oracle Rdb release 6.1 and later, a new SQL connect is created for each client application that connects to the service. If the service is defined with database authorization by the service owner, then the session user name for each SQL connect of a client application defaults to the system user name. If the service is defined with database authorization by connect user, then each SQL connect of a client application is created using the connect user name for each individual client connection. When a client application disconnects from the service, the SQL connect of the client application is deleted. For a session reusable database service, there is a maximum of one client application SQL connect per executor. For a transaction reusable database service, there is one client application SQL connect for each concurrent client connection.

2.9.1.4 Oracle Rdb Current User Name

The current user name is always set to the value of the session user name except during the execution of a definer's rights stored procedure, in which case, the current user name is set to the definer's user name.

Whenever a database request is started, Oracle Rdb must determine if the process issuing the request is authorized to execute the request. To perform this check, Oracle Rdb first merges

the system privileges of the process accessing the database with the database privileges of the current user name. For a local attach, the process accessing the database is the Oracle SQL/Services executor process. For a remote attach, the process accessing the database is the Oracle Rdb server process.

The process privilege mask of the operating system is used as the system privileges for the executor process.

After Oracle Rdb merges the privileges, it then determines if the combination of these privileges is sufficient to execute the request. Because Oracle Rdb combines the privileges in this way, you must carefully choose the service owner user name for a database service. See [Section 2.10](#) for more information.

For example, consider a database service called PAYROLL that is defined with a service owner user name of SYSTEM and with database authorization set to the connect user name. User SMITH might not normally be authorized to update a table called EMPLOYEE_PAY in the payroll database. However, if user SMITH accesses the payroll database using the PAYROLL service, the database privileges for user name SMITH, when combined with the system privileges for the SYSTEM user name, which include SYSPRV and BYPASS, allow this user full access to the EMPLOYEE_PAY table and all other tables in the database.

2.9.2 Setting the Process User Name and the Oracle Rdb System User Name

To set the Oracle Rdb system user name, Oracle SQL/Services uses a process user name impersonation mechanism to set the process user name and all associated process attributes of an executor process. By setting the process user name, Oracle SQL/Services automatically establishes the correct default for the Oracle Rdb system user name. Furthermore, by setting the process user name, Oracle SQL/Services also establishes the correct environment for the consistent execution of external functions and procedures that execute within the context of the executor process.

Oracle SQL/Services sets the process user name at different times, based on the type of service you provide:

- Universal service

Oracle SQL/Services sets the process user name every time a new client connect is assigned to an executor process for a universal service. This ensures the correct environment at all times for the execution of external functions and procedures that execute within the context of the executor process.

- Database service

Oracle SQL/Services sets the process user name once for an executor process for a database service at the time the executor process is first started. However, to ensure the

correct and successful execution of database requests once an executor is attached to a database, Oracle SQL/Services cannot and does not reset the process user name when a new client connect is assigned to an executor process. This behavior provides a consistent environment for the execution of external functions and procedures that execute within the context of the executor process. However, it means that all such functions and procedures are executed under the service owner user name, rather than the connect user name for a service with the database authorization attribute set to connect user. See [Section 2.10.2](#) for more information on using external functions and procedures with Oracle SQL/Services, including information on how to define external functions and procedures to execute within the context of an independent server process with the rights and privileges of the connect user name.

When Oracle SQL/Services creates an executor process, the Oracle SQL/Services monitor process merges the *authorized* and *default* privileges of the service owner account. The combination of these privileges becomes the *authorized privilege* mask of the executor process. When Oracle SQL/Services resets the process user name of an executor process, it sets the *process privilege* mask and *current privilege* mask of the executor process by merging the *authorized* and *default* privileges of the new process user name. However, Oracle SQL/Services cannot set the *authorized privilege* mask of an executor process. Therefore, you must ensure that a service owner account does not have excess *authorized* or *default* privileges. Typically, you will grant only the TMPMBX and NETMBX privileges to a service owner account.

2.10 Considering Security for Selecting the Service Owner User Name

The security criteria that you use to select and configure an account for use as a service owner account is based on the platform on which the server is running, the type of services you are providing, and the database authorization attribute of the services.

2.10.1 Execution Environment for Database Requests

The following guidelines can help you select and configure an account for use as a service owner account on OpenVMS systems based on the service type and the database authorization attribute of the service.

Universal Services

Database Access Authorization Set to Connect User Name For a universal service with database authorization set to connect user name, you should select an account with a nonsystem user identification code (UIC) that has minimal privileges. The Oracle SQL/Services installation procedure creates a nonprivileged account named

SQLSRV\$DEFLT that may be used for all universal services with database authorization set to connect user name.

Database Access Authorization Set to Service Owner For a universal service with database authorization set to service owner, you should select an account with a nonsystem UIC that has minimal privileges and that has been granted the necessary database access to only those databases that are designed to be accessed by the service. To ensure the security and integrity of your data, the account you select will usually be severely restricted in the access it has to databases on your system and the data contained therein.

Database Services

Database Access Authorization Set to Connect User Name For a database service with database authorization set to connect user name, you should select an account with a nonsystem UIC that has minimal privileges. Because all database requests are executed using the connect user name, the account you select as the service owner user name need only be granted the right to attach to the database. For example, by granting to the SQLSRV\$DEFLT account the right only to attach to the database, you can use the nonprivileged account created by the Oracle SQL/Services installation procedure.

Database Access Authorization Set to Service Owner For a database service with database authorization set to service owner, you should select an account with a nonsystem UIC that has minimal privileges and that has been granted the right to access certain specific data within the database and that has been granted the right to execute certain specific operations against that data. The amount of access you grant to the service owner account will be specific to each database for which you provide a database service with database authorization set to service owner.

You must configure each service owner account with these minimum privileges and quotas:

- TMPMBX and NETMBX privileges
- Default account quota values suffice, with the following exceptions:
 - ENQLM - set quota of 2000
 - JTQUOTA - set quota of 4096

If your database has hundreds of storage areas, you might also need to increase the PGFLQUOTA (paging file limit) for the process, using AUTHORIZE, or the PAGFILCNT and VIRTUALPAGECNT system parameter values, using the System Generation (SYSGEN) utility. Allow 60 pages per storage area.

You use AUTHORIZE to verify and change user accounts. You must have system privileges to use AUTHORIZE. At the AUTHORIZE prompt (UAF>), enter the SHOW command with an account name to check that particular account. For example:

```
$ SET DEFAULT SYS$SYSTEM
$ RUN AUTHORIZE
UAF> SHOW SMITH
```

To change quotas and privileges, use the MODIFY command:

```
MODIFY account-name /quota-name=NNN /PRIVILEGE=(priv-name) /DEFPRIV=(priv-name)
```

The following example changes the FILLM quota for the SMITH account, and gives it the TMPMBX and NETMBX privileges:

```
UAF> MODIFY SMITH /FILLM=300 -
_UAF> /PRIVILEGE=(TMPMBX,NETMBX) /DEFPRIV=(TMPMBX,NETMBX)
```

Users must log out and log in again for changes made in AUTHORIZE to take effect. For more information on modifying account quotas, see the description of the OpenVMS Authorize utility in the OpenVMS system management documentation.

See the *Oracle Rdb Installation and Configuration Guide, Release 7.1* for more information on Oracle Rdb requirements.

2.10.2 Execution Environment for External Functions and Procedures

You can define external functions and procedures to execute within the context of the executor process or in an independent server process that Oracle Rdb creates specifically to execute external functions and procedures.

To define an external function or procedure to execute within the context of the executor process, use the SQL BIND ON CLIENT SITE syntax. From the perspective of the Oracle Rdb database engine, the database client is the Oracle SQL/Services executor process, not the Oracle SQL/Services client. To define an external function or procedure to execute in an independent server process, use the SQL BIND ON SERVER SITE syntax.

See [Section 2.7](#) for a complete, in-depth discussion of how Oracle SQL/Services implements the database access authorization models. See the *Oracle Rdb SQL Reference Manual* and the *Oracle Rdb7 Guide to SQL Programming* for more information on defining external functions and procedures.

2.10.2.1 External Functions and Procedures Executing in the Context of the Executor Process

Because you can define external functions and procedures to execute within the context of the executor process, you should consider this when you configure service owner accounts.

Universal Services

Database Access Authorization Set to Connect User Name External functions and procedures defined to execute in the context of the executor process always execute with the rights and privileges of the connect user name using this type of service.

Database Access Authorization Set to Service Owner External functions and procedures defined to execute in the context of the executor process always execute with the rights and privileges of the service owner user name under this type of service.

Database Services

Database Access Authorization Set to Connect User Name External functions and procedures defined to execute in the context of the executor process always execute with the rights and privileges of the *service owner user name* under this type of service. This is because Oracle SQL/Services cannot reconfigure an executor process once it has attached to the database. To have external functions and procedures execute with the rights and privileges of the connect user name, you must define the external functions and procedures to execute in an independent server process using the SQL BIND ON SERVER SITE syntax.

Database Access Authorization Set to Service Owner External functions and procedures defined to execute in the context of the executor process always execute with the rights and privileges of the service owner user name under this type of service.

2.10.2.2 External Functions and Procedures Executing in the Context of an Independent Process

You can define external functions and procedures to execute within the context of an independent server process. With this model, the execution environment for external functions and procedures is based on the database authorization attribute regardless of whether you are using universal or database services.

Universal and Database Services

Database Access Authorization Set to Connect User Name

External functions and procedures defined to execute in the context of an independent server process always execute with the rights and privileges of the connect user name with this type of authorization.

Database Access Authorization Set to Service Owner

External functions and procedures defined to execute in the context of independent server process always execute with the rights and privileges of the service owner user name with this type of authorization.

2.11 Setting the Attributes for Number of Executors

The use of some services on your system may be fairly constant over time, whereas the use of other services may vary over time with peaks and lulls in the user activity. By setting appropriate values for the `MIN_EXECUTORS`, `MAX_EXECUTORS`, and `IDLE_EXECUTOR_TIMEOUT` attributes for a service, you can provide an efficient service to your clients.

You must always set the `MIN_EXECUTORS` attribute to the same value as the `MAX_EXECUTORS` attribute for a transaction reusable service. This is to allow Oracle SQL/Services to distribute new client connections evenly over the pool of available executor processes for a service.

2.11.1 Configuring a Fixed Number of Executors for a Service

For a service that has a fairly constant number of users connected to it over time, Oracle recommends that you set the `MIN_EXECUTORS` attribute to the same value as the `MAX_EXECUTORS` attribute. This ensures that a constant number of executors are prestarted and are always available to client applications. This avoids the delay while an executor process is started for a new client connection.

2.11.2 Configuring a Variable Number of Executors for a Service

For a service where the number of connected users varies over time, with more users at peak times and fewer users at less busy times, you can choose to adjust the number of executors to suit any load. To do this, you can choose to have the Oracle SQL/Services server automatically start new executor processes as they are needed, or you can prestart new executor processes in anticipation of increased demand at peak times.

2.11.2.1 Starting New Executor Processes as They Are Needed

The simple approach to handling peaks and lulls in the demand for a service is to set the `MIN_EXECUTORS` attribute to a value that supports the activity for the service at normal

times, set the `MAX_EXECUTORS` attribute to a value that supports the activity for the service at peak times, then let Oracle SQL/Services create new executor processes as demand increases during peak periods of use. By choosing a suitable value for the `IDLE_EXECUTOR_TIMEOUT` attribute, you can ensure that executors remain active once they have been started, even if demand might decrease for a little while. Although this approach is very easy to configure and manage, a disadvantage with this approach is that new users who connect to the service at the beginning of a peak period will encounter a slight delay if new executor processes must be created.

2.11.2.2 Prestarting New Executor Processes Ahead of Increased Demand

A more complex approach to handling peaks and lulls in the demand for a service is to set the `MAX_EXECUTORS` attribute to a value that supports the activity for the service at peak times, then create `SQLSRV_MANAGE` scripts that can be used to increase the value of the `MIN_EXECUTORS` attribute for the service at peak times and decrease the value of the `MIN_EXECUTORS` attribute for the service at a time when demand for the service starts to decrease.

You can automatically invoke the `SQLSRV_MANAGE` scripts to increase the `MIN_EXECUTORS` attribute value in anticipation of the increase in the number of users of a service and decrease the `MIN_EXECUTORS` attribute value at the end of a peak period by writing command procedures for batch jobs. The advantage of this approach of prestarting executors ahead of demand is that new users who connect to the service at the beginning of a peak period will not encounter delays as executor processes are created. A disadvantage of this approach is that it is more complex to manage.

2.12 Using a SQL Initialization File

You can use the `SQL_INIT_FILE` argument of the `CREATE SERVICE` or `ALTER SERVICE` command to specify a file containing SQL statements that tailor the SQL environment for a client connection. For example, you can set the SQL dialect and default character set by using a SQL initialization file. The statements in a SQL initialization file are executed every time a client connects to a service.

See [Section 6.1](#) for more information about syntax conventions used in a SQL initialization file.

2.13 Using SQL/Services Logical Names

Table 2–6 summarizes logical names that you can use with Oracle SQL/Services and OCI Services for Oracle Rdb.

Table 2–6 SQL/Services Logical Names

Logical Name	Description
SQLSRV_EXEC_LOG	Disables the service log for all services
SQLSRV_DISP_LOGPATH	Specifies the location of the dispatcher log directory
SQLSRV_DISP_DUMPPATH	Specifies the location of the dump file directory
SQLNET_DEBUG_FLAGS	Enables additional information to be logged in the SQL/Services executor log file.
SQLNET_RECO_USER	If you are using OCI Services for Oracle Rdb from a database link and you are doing two-phase commit, if either the Oracle system or the Oracle Rdb system fail while in the middle of a transaction, the Oracle Transaction Manager tries to reconnect to the Oracle Rdb database to verify the results of the transaction. In order for OCI Services for Oracle Rdb to be able to make this connection, it needs a user name and password. The user needs access only to the ORA_COMM_TRANS table which records the results of transactions as they are in progress.
SQLNET_BLOB	In previous versions of OCI Services for Oracle Rdb, Oracle Rdb segmented strings were treated as long character strings by the Oracle tool. Much of the OCI LOB interface is now implemented, so it is possible to treat segmented strings as Oracle LOBs. To enable this, you must define this logical to be "Y" or "y". If you define the logical as anything else, or if you do not define it, segmented strings in Oracle Rdb will be treated like long character strings as they have been in previous releases.
SQLNET_DOMAIN	OCI Services for Oracle Rdb assumes that the default domain name is .WORLD. You can define the logical SQLNET_DOMAIN to change the domain name. This is especially relevant for people accessing OCI Services for Oracle Rdb using dblinks from an Oracle database. The logical can be defined in the process initialization file for the service.
SQLNET_BUGCHECK_FILE	You can define the logical SQLNET_BUGCHECK_FILE to specify the name of the dump file.

Table 2–6 SQL/Services Logical Names

Logical Name	Description
SQLSRV\$MAX_EXECUTOR_FAILURES	You can define the logical SQLSRV\$MAX_EXECUTOR_FAILURES to change the maximum number of failures allowed before an executor fails. The value assigned to the logical overrides the default value of two. In this way, you can control how often executors and services shut down during routine database maintenance.
SQLSRV\$LOG_CONNECTIONS	You can define the logical SQLSRV\$LOG_CONNECTIONS to "NO" so that successful connections are not logged to dispatcher log files, and the size of the dispatcher log files is reduced. If the logical is undefined or assigned to any other value, the successful connections are logged. Because this logical is evaluated when a dispatcher is started, the dispatcher must be restarted if the logical is changed.
RDB\$DDTM_XG_INFO	When the logical RDB\$DDTM_XG_INFO is defined to be SYSTEMS<nodename>, it specifies that the XA Gateway is to be used for two-phase commit transactions. If this logical is defined, the SQLNET_RECO_USER logical must also be defined.
SQLNET_VALIDATE_PROGRAM	If you define the SQLNET_VALIDATE_PROGRAM logical to 'Y', OCI Services for Oracle Rdb validates each user name and program during logon processing against entries in the ORA_VALID_PROGRAMS table and rejects any that do not have matching entries.
SQLNET_STRUCTURED_DATE_TYPES	Use this logical to turn structured date types on and return true data types to the OCI client. See Section 4.1.9 for the actual data types that are returned by OCI Services for Oracle Rdb.
SQLNET_TIMESTAMP_DATE_TYPE	Use this logical to turn on timestamp data types. When this logical is set to "Y", TIME and TIMESTAMP data types are returned as TIMESTAMP; otherwise, the data types are returned as DATE.

If you want to disable the service log for all services, you must define the SQLSRV_EXEC_LOG logical before a service is started, as shown in the following example:

```
$DEFINE/SYSTEM SQLSRV_EXEC_LOG NOLOG
```

Oracle recommends that you do not disable the service log because it is needed if a problem occurs.

The following example shows how you can specify the location of the dispatcher log directory:

```
$ DEFINE/SYSTEM/EXEC SQLSRV_DISP_LOGPATH DKA100:[USER1.LOG]
```

The following example shows how you can specify the location of the dump file directory:

```
$ DEFINE/SYSTEM/EXEC SQLSRV_DISP_DUMPPATH DKA100:[USER1.DUMP]
```

The `SQLSRV_DISP_LOGPATH` and `SQLSRV_DISP_DUMPPATH` logical names must be defined as system logical names. If you do not define the `SQLSRV_DISP_LOGPATH` logical name or the `SQLSRV_DISP_DUMPPATH` logical name, the default directory for dispatcher log files and dump files is the `SYSSMANAGER` directory.

Once you define either the `SQLSRV_DISP_LOGPATH` or `SQLSRV_DISP_DUMPPATH` logical name, you must restart the dispatcher.

Maintaining an Oracle SQL/Services Server

After you set up Oracle SQL/Services and configure one or more servers, you should periodically perform maintenance tasks, which include:

- Monitoring server activity
- Monitoring client connections

Each of these topics is described in the sections that follow.

3.1 Monitoring Server Activity

Monitoring server activity consists in part of using the `SHOW` commands to show the operational state of objects. For example, for service and dispatcher objects, a `SHOW` command will inform you if the object is running. If you find that a service or dispatcher object is not running and should be running, you should check the log and dump files to determine why the object stopped running. After resolving the problem, issue either a `START SERVICE` or `START DISPATCHER` command and specify the service or dispatcher name of the object you want to start up. Perform another `SHOW` command to confirm that the service or dispatcher object is running.

Using the `SHOW SERVICES` command, you can also monitor client activity during peak load periods for all services provided on that server. For example, if the number of active clients approaches the maximum number allowed, you should consider increasing the maximum number of clients allowed to reduce the chances of client connection failures. You can dynamically increase the `MAX_EXECUTORS` value for a particular service by using the `ALTER SERVICE` command.

3.2 Monitoring Client Connections

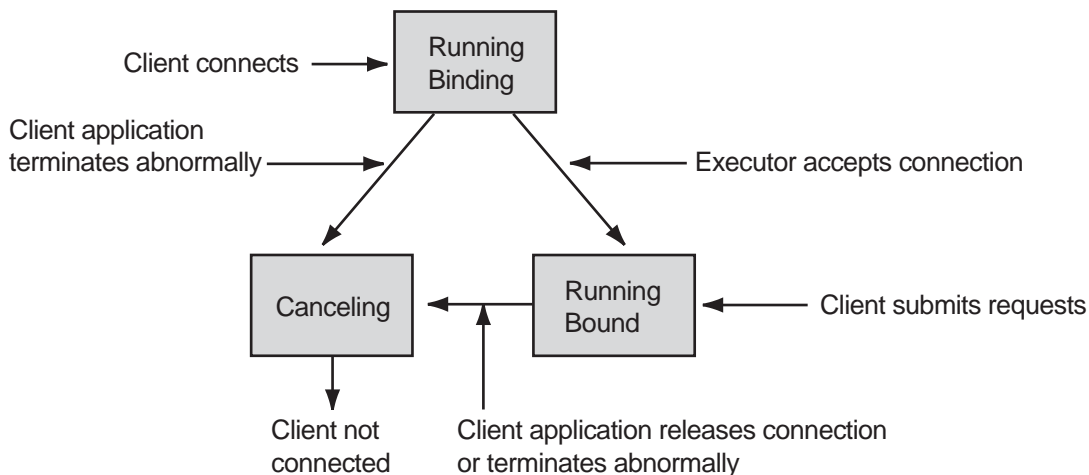
You can use the `SHOW CLIENTS` command to show the state of clients as each connects to a service, submits requests, and releases the connection. The occurrence, sequence, and duration of connection states are different for each type of service. The client state can help you determine what each connection is doing and if connections are being serviced normally. However, the connection state information by itself may not be sufficient for troubleshooting all problems. For more information on troubleshooting problems, see [Section 7.7](#).

[Section 3.2.1](#) and [Section 3.2.2](#) describe the states that a client connection can display, the sequences that can occur, and the relative duration of each state when serviced by either a session reusable service or a transaction reusable database service.

3.2.1 Client Connection States for Session Reusable Services

[Figure 3–1](#) shows the three possible connection states that a `SHOW CLIENTS` command can display for a client connection when serviced by an executor process for a session reusable service relative to client and executor events.

Figure 3–1 *Client Connection States for Session Reusable Services*



The connection from a client attempting to connect to a session reusable service is in a **Running Binding** state while it waits for an executor to accept the connection. A connection is in the **Running Binding** state only momentarily if a free executor process is available to

accept the connection. However, a connection remains in the Running Binding state for a longer period of time if a new executor process must be created for the connection, which may take several seconds.

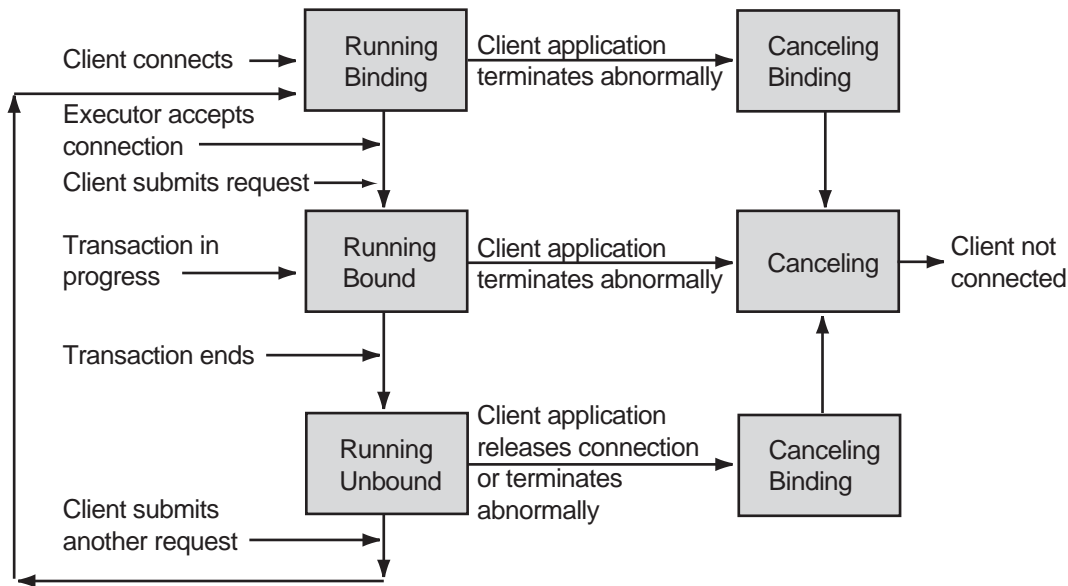
When an executor process accepts a connection, the connection state transitions from Running Binding to Running Bound. Once an executor for a session reusable service accepts a connection, the executor remains bound to that connection for the duration of the connection.

A connection transitions to the Canceling state when the application releases the connection normally, or if the application terminates abnormally. A connection typically remains in the Canceling state only momentarily. However, a connection may remain in the Canceling state for a longer period of time if other database activity delays the cleanup of an outstanding database transaction.

3.2.2 Client Connection States for Transaction Reusable Database Services

An executor for a transaction reusable service processes requests for one transaction for one client at a time; however, the executor is shared by many concurrent client connections. A transaction begins when a client issues a SQL statement that either implicitly or explicitly starts a transaction. A transaction ends when the client issues a successful SQL COMMIT or ROLLBACK statement or executes a stored procedure that commits or rolls back a transaction. Once assigned to an executor process, a client connection remains tied to that process for the life of the connection; no other executor process can be used to process transactions on behalf of a particular connection. A new client connection is normally assigned to the executor with the least number of existing connections; however, for certain applications, it may be necessary to change this behavior using the ALTER SERVICE APPLICATION TRANSACTION USAGE CONCURRENT attribute.

Figure 3–2 shows the five possible connection states that a SHOW CLIENTS command can display for a client connection when serviced by executors for a transaction reusable database service relative to client and executor events.

Figure 3–2 Client Connection States for Transaction Reusable Database Services

The connection from a new client attempting to connect to a transaction reusable service is in a Running Binding state while it waits for the assigned executor to accept the connection. Likewise, when an existing client begins a new transaction, the connection is in a Running Binding state while it waits for the assigned executor to process the new transaction. For well-designed applications that are executing short transactions, connections remain in the Running Binding state for short periods of time. However, this time increases as the rate at which clients execute transactions increases and as the average length of transactions increases.

When an executor process binds to a new or an existing connection, the connection state transitions from Running Binding to Running Bound. Once bound to a connection, the executor remains bound to that connection until the end of the transaction. In a new connection, the executor remains bound to the connection only for the time necessary to establish a new database session for the new connection. At the end of a transaction, or after accepting a new connection, the executor unbinds from the connection, and the connection state transitions from Running Bound to Running Unbound.

A connection in the Running Binding or Running Unbound state transitions to the Canceling Binding state when the application releases the connection normally or when the application terminates abnormally. When the executor completes the transaction for the currently bound

connection, plus any other transactions for connections that may be queued up already waiting for the executor, the Canceling Binding connection transitions from the Canceling Binding state to the Canceling state. A connection in the Running Bound state transitions directly to the Canceling state when the application releases the connection normally or when the application terminates abnormally. When in the Canceling state, the executor cleans up the database session of the connection, then unbinds from the connection for the last time.

OCI Services for Oracle Rdb Processing

OCI Services for Oracle Rdb (formerly known as SQL*Net for Oracle Rdb) provides an environment in which you can run existing OCI applications to access data in Oracle Rdb databases. The unique advantage offered by OCI Services for Oracle Rdb is the ability to use Oracle SQL semantics to access data in Oracle Rdb databases.

Within Oracle Rdb SQL, the Oracle Level1 dialect, the Oracle Level2 dialect, and the Oracle functions were built specifically with OCI Services for Oracle Rdb in mind. The Oracle Rdb SQL features help SQL programmers create client applications that can run against both the Oracle Rdb server and the Oracle server.

This chapter provides more information about the following aspects of OCI Services for Oracle Rdb processing:

- [Section 4.1](#) describes how OCI Services for Oracle Rdb augments Oracle Rdb SQL with many processing features to allow common application development between the Oracle RDBMS server and the Oracle Rdb server.
- [Section 4.2](#) describes the SQL initialization file that sets specific session parameters to initialize the Oracle SQL/Services execution environment.
- [Section 4.3](#) describes how to establish compatibility with Oracle Forms.
- [Section 4.4](#) describes how to use Oracle SQL/Services universal services.
- [Section 4.5](#) describes how to use the RDB_NATCONN71 command file to prepare or upgrade an Oracle Rdb database, drop OCI-associated tables and functions from an Oracle Rdb database, or to add, remove, modify, or show users with encrypted passwords in a prepared Oracle Rdb database.
- [Section 4.6](#) describes character set usage.
- [Section 4.7](#) describes how to reference an Oracle Rdb database as a database link from an Oracle Rdb database.

- [Section 4.8](#) describes how to retain OCI Services for Oracle Rdb functionality after an IMPORT/EXPORT cycle.

4.1 OCI Services for Oracle Rdb Features

The following sections provide a more detailed description of these OCI Services for Oracle Rdb processing features.

4.1.1 OCI Message Mapping

With OCI, you can open and fetch a number of rows with a single call. To emulate this capability, OCI Services for Oracle Rdb implicitly performs a number of steps to achieve the same result:

1. Opens a cursor
2. Fetches the specified number of rows
3. Responds with the data

The order in which a cursor is opened and described is different when using OCI compared to using Oracle Rdb dynamic SQL statements. OCI Services for Oracle Rdb hides the differences by manipulating the message order and presenting the OCI message order to the client.

4.1.2 Cursor Management

Because OCI uses a cursor for every statement, OCI Services for Oracle Rdb manages a virtual OCI cursor for each statement. OCI Services for Oracle Rdb then ties these virtual OCI cursors to Oracle Rdb dynamic SQL statement IDs for most statements or to Oracle Rdb dynamic SQL cursors for SELECT statements.

4.1.3 Data Types

OCI Services for Oracle Rdb fetches data in machine native data types, and then converts the data to Oracle portable data types prior to sending it to the client.

Conversely, OCI Services for Oracle Rdb receives data in portable data types, and then converts and passes them to Oracle Rdb dynamic SQL as machine native data types.

4.1.4 Data Definition Language

Before and after each DDL request, OCI Services for Oracle Rdb mimics Oracle behavior by issuing a COMMIT statement.

4.1.5 SQL Cursor Semantics

For all dynamic SQL declare cursor statements, OCI Services for Oracle Rdb adds the WITH HOLD PRESERVE ALL clause. The benefit of this action is that cursors stay open across transactions. This behavior mimics the Oracle server behavior.

4.1.6 Oracle SQL ALTER SESSION Statement

The ALTER SESSION statement, as documented by the *Oracle Server SQL Language Reference Manual*, is processed by OCI Services for Oracle Rdb to provide a variety of information. OCI Services for Oracle Rdb supports only the ALTER SESSION statement syntax described in [Chapter 5](#).

4.1.7 Data Formatting

The Oracle server formats data for the client and receives formatted data from the client. The formatted information is passed to the server using the default data formats set by the ALTER SESSION statement. This enables three important Oracle features:

- The correct processing of the TO_DATE, TO_CHAR, and TO_NUMBER functions.
- The correct handling of date literals in the format specified by the ALTER SESSION statement.
- The correct formatting of date and numeric data when fetched as text according to the information specified in the ALTER SESSION statement.

See [Chapter 5](#) for more information about the SQL ALTER SESSION statement.

4.1.8 Statement Parsing

Oracle Rdb SQL provides Oracle Level1 and Oracle Level2 dialects. However, even with these dialects, there are a number of Oracle constructs that Oracle Rdb SQL does not accept. If Oracle Rdb SQL rejects a statement for specific reasons, such as a date conversion error, OCI Services for Oracle Rdb examines the statement and replaces the Oracle format date literal with an equivalent statement that Oracle Rdb SQL accepts.

Note: Parsing does not occur unless the statement fails. For example, if you provide a valid OpenVMS date literal, it will be processed without the assistance of OCI Services for Oracle Rdb.

The following examples demonstrate statement parsing.

Example 4–1 Inserting an Oracle Date Literal into an ANSI Date Column

This example attempts to insert an Oracle date literal into an ANSI date column:

```
INSERT INTO ATABLE VALUES ('3-AUG-46');
```

Because this statement is rejected by Oracle Rdb SQL, OCI Services for Oracle Rdb replaces it with the following:

```
INSERT INTO ATABLE VALUES (CAST (TO_DATE('3-AUG-46') AS DATE
ANSI));
```

The TO_DATE function supplied by OCI Services for Oracle Rdb is similar to the Oracle TO_DATE function used to format strings into dates. Because the TO_DATE function supplied by OCI Services for Oracle Rdb returns a DATE VMS date, you must use the CAST function to match the DATE ANSI format.

Note: Oracle recommends the use of DATE VMS dates whenever possible. DATE VMS most closely resembles the Oracle DATE data type.

Example 4–2 Inserting the Word CALL into a Procedure Call

This example shows an Oracle procedure call:

```
BEGIN UPDATE_EMPLOYEE_NAME('FIRST_NAME', 'LAST_NAME'); END;
```

Because this statement is not recognized by Oracle Rdb, it returns an error. OCI Services for Oracle Rdb recognizes the statement as a procedure call and inserts the word CALL into the statement. The resulting statement can be executed correctly by Oracle Rdb.

```
BEGIN CALL UPDATE_EMPLOYEE_NAME('FIRST_NAME', 'LAST_NAME'); END;
```

To see how OCI Services for Oracle Rdb modifies SQL statements, you can turn on logging with the ALTER SESSION LOG BRIEF statement (described in [Chapter 5](#)). To make them readily identifiable, the modified statements have the comment "-- GTW Fixed up" added to them.

4.1.9 Data Type Descriptions

OCI Services for Oracle Rdb describes all the Oracle Rdb data types in terms of Oracle data types:

- On OpenVMS Alpha, you can request that true data types be returned to the OCI client by setting the `SQLNET_STRUCTURED_DATE_TYPES` logical to "Y" or "ON" or by issuing the `ALTER SESSION SET SQLNET_STRUCTURED_DATE_TYPES` command. The following table shows the data types that are returned to the client.

Oracle Rdb Data Type	Data Types Returned to Client	
	<code>SQLNET_STRUCTURED_DATE_TYPES = ON</code> or <code>YES</code>	<code>SQLNET_STRUCTURED_DATE_TYPES = OFF</code> or <code>NO</code>
DATE	DATE	DATE
TIME	DATE	DATE
TIMESTAMP	TIMESTAMP	DATE
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH	CHAR
INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND	CHAR

When this argument is set to ON or YES and the OCI client specifies structured data types, structured data types are returned to the client.

- On OpenVMS Alpha, you can turn on timestamp data types by setting the `SQLNET_TIMESTAMP_DATE_TYPE` logical to "Y" or by issuing the `ALTER SESSION SET SQLNET_TIMESTAMP_DATE_TYPE ON` command. If you have enabled the `TIMESTAMP` data type, `TIME` and `TIMESTAMP` data types are returned as `TIMESTAMP`. If the `TIMESTAMP` data type has not been turned on, the data types are returned as `DATE`.
- All `CHAR` data types greater than 2000 bytes and `VARCHAR` data types greater than 4000 bytes are described as `LONG`.
- The `LIST OF BYTE VARYING` column is described as `LONG`, or `BLOB` if the logical `SQLNET_BLOB` is defined as Y.
- All the numeric data types are described as Oracle numbers:
 - An `INTEGER` is described as a `NUMBER(10,0)`.
 - A `BIGINT(2)` is described as a `NUMBER(19,2)`.

Data type precision is described as one more place of precision than can actually be represented. This is because Oracle Rdb uses native binary data types whose range does not map directly to a decimal range. Oracle numbers use decimal representation. So, when OCI Services for Oracle Rdb describes an Oracle Rdb

TINYINT column as NUMBER(3,0), the column cannot hold the number 999, but it can hold the number 111.

4.1.10 Oracle Data Dictionary

When you prepare an Oracle Rdb database for OCI Services for Oracle Rdb, you install a number of database objects used to help OCI Services for Oracle Rdb emulate the Oracle data dictionary that can be used by OCI clients. These objects include:

- The Oracle metadata tables (or data dictionary) provided as views over the Oracle Rdb metadata tables. For OCI Services for Oracle Rdb releases prior to release 7.1.6, these metadata objects appeared to be user tables and were included in the list of tables displayed by a SHOW TABLES command. Beginning with release 7.1.6, these objects are created using the Oracle Rdb functionality HIDE_OBJECTS. They appear in a SHOW SYSTEM TABLES list but not in a SHOW TABLES list. If you upgrade from a prior version, some of the objects still appear as user objects and some appear as system objects. This does not impact any of the OCI Services for Oracle Rdb functionality.

See [Appendix A](#) for complete details.

- Four domains that OCI Services for Oracle Rdb adds as a convenience for programmers migrating Oracle applications. These four domains are:

```
NUMBER AS DOUBLE PRECISION
RDB1LONG AS VARCHAR (65238)
RDB2LONG AS VARCHAR (32634)
RDB4LONG AS VARCHAR (16317)
```

- TO_DATE (DATA, FORMAT, NLS_parameters)—Data is a string literal and uses either the provided format string or the default format string to convert the string to a DATE VMS data type. With Oracle Rdb, you cannot combine DATE VMS and ANSI date-time data types without using a CAST function. In the Oracle Level1 dialect the DATE VMS data type can be used with mathematical operators, so use it whenever possible.
- TO_NUMBER (DATA, FORMAT, NLS_parameters)—Data is a string literal and uses either the provided format string or the default format string to convert the string to a DOUBLE PRECISION data type. The TO_NUMBER function is restricted by the DOUBLE PRECISION data type, so integers with a precision 16 or greater cannot be represented precisely.
- TO_CHAR (DATA, FORMAT, NLS_parameters)—Data is a number or date literal and creates a formatted character string. When using TO_CHAR with an unscaled integer of precision 18, a format string must be provided. Use a format string when using TO_

CHAR with unscaled BIGINT data. TO_CHAR assumes an 18-digit number is a date if no format string is provided.

- USERENV—Given one input string, the function USERENV supplies details about the current session as a VARCHAR data type. Only the 'TERMINAL' and 'LANGUAGE' input values return any meaningful information. Input strings 'LABEL', 'SESSIONID', 'USERMODE', and 'ENTRYID' return valid fixed values.
- CHARTOROWID (DATA)—Data is a string literal that is converted to a rowid or dbkey.
- ROWIDTOCHAR (DATA)—Data is a rowid or dbkey that is converted to a string.

Refer to the Oracle documentation for more information about the referenced functions, domains, and data types.

4.1.11 Multischema Emulation

Because most Oracle Rdb databases are not multischema databases and because all Oracle databases are multischema, OCI Services for Oracle Rdb provides a form of multischema emulation. Multischema emulation uses the Oracle data dictionary and hooks into the SQL compiler.

Multischema emulation is enabled by default. If you do not require multischema emulation, OCI Services for Oracle Rdb provides the ALTER SESSION SET SCHEMA EMULATION RELAXED statement to allow you to disable it. See [Chapter 5](#) for additional information about this statement.

4.2 Initializing Your Server Environment

OCI Services for Oracle Rdb databases are served by Oracle SQL/Services. To initialize the execution environment, Oracle SQL/Services allows you to specify a SQL initialization file for the service. The initialization file executes SQL statements that set specific session parameters (for example, locking defaults or character set defaults).

You can execute most initialization statements:

- Directly in the SQL initialization file defined for the service
- Indirectly using the ORA_INIT stored procedure

The advantage to using the ORA_INIT stored procedure is that it allows you to conditionally enable data definition language (DDL) statements such as the ALTER SESSION statement in an IF block. Although Oracle Rdb SQL does not allow DDL statements in a compound statement such as an IF block, the ORA_INIT stored procedure allows you to store DDL and

other statements for subsequent execution by OCI Services for Oracle Rdb. Initialization statements that you stipulate with `ORA_INIT` are executed in the order you specify them after the SQL initialization file defined for the service has completed.

For example, you might want to enable full server logging with connections from the Oracle Net client application. Even though you can query the `ORA_SESSION` table to determine the client program name, you cannot form a query in the SQL initialization file that conditionally enables full server logging depending upon the client program name. This is because the `ALTER SESSION LOG FULL` statement is a DDL statement, and cannot occur in a compound statement such as an IF block. However, you can use the `ORA_INIT` stored procedure in your SQL initialization file to achieve the same effect, as follows:

1. Define a stored procedure, `SQLPLUS_LOG`, that you can use to determine if the client program is a SQL*Plus application.

The stored procedure might be similar to the following:

```
create module SQLPLUS_LOG_MODULE language sql
  procedure SQLPLUS_LOG;
begin
  declare :A integer;
  select count(*) into :A from ORA_SESSION
    where (INFO_TYPE = 'PROGRAM' and INFO containing 'SQLPLUS');
  if :A > 0 then call ORA_INIT ('ALTER SESSION LOG FULL'); end if;
end;
end module;
```

2. Call the `SQLPLUS_LOG` procedure from your service SQL initialization file, as follows:

```
call SQLPLUS_LOG();
```

3. If the client program is an Oracle Net application, call the `ORA_INIT` procedure to store an `ALTER SESSION LOG FULL` statement for subsequent execution by OCI Services for Oracle Rdb.

Note: Even though querying the client program name or client terminal name can be a convenient and powerful method of tailoring the server environment to the client, Oracle does not recommend that you use this method for security purposes. OCI Services for Oracle Rdb cannot guarantee that the client application has accurately reported the client program name or client terminal name.

See [Section A.2](#) for more information about the `ORA_SESSION` table.

4.3 Compatibility with Oracle Forms

If you use Developer/2000 prior to release 2.0 or prior to Oracle Forms release 5.0, you must set the following properties in your Oracle Forms module for compatibility with OCI Services for Oracle Rdb:

- Set the Module property Savepoint Mode to FALSE
- Set the Data Block property Locking Mode to DELAYED

You can modify the properties using the Property Palette or use the Property Class RDB_PROPERTY_CLASS of the file RDB_PROP.FMB provided in the Oracle SQL/Services examples directory on your server.

Beginning with Developer/2000 release 2.0 or Oracle Forms Release 5.0, these properties will be handled automatically during run time by Developer/2000.

4.4 Using Oracle SQL/Services with Universal Services

A universal service allows an Oracle client application to attach to any number of Oracle Rdb databases. It allows one or a small number of Oracle SQL/Services universal services to support a large number of infrequently used databases. With one or a small number of universal services serving any number of databases, it is reasonable to prestart the executor processes. This is an alternative to setting the minimum number of executors so that one is always running. Refer to [Section 2.6](#) for more information on universal services.

With the Oracle Network Manager release 3.1.0.5.1 and later, an Oracle Rdb database name can be supplied in the connect string for an Oracle Rdb service. The RDB_DATABASE field in the connect string is used to specify the database or databases to which a universal service should attach.

If the Oracle SQL/Services service to which the OCI connect strings map is a universal service, clients can specify the Oracle Rdb database name as one of the following:

- A full Oracle Rdb SQL ATTACH statement
- An @file_spec

The file you specify can contain SQL statements that tailor the SQL environment for a client connection. Also, the use of a file specification is a way for the the client application to execute multiple ATTACH statements. The first ATTACH statement cannot contain the ALIAS clause, as the first ATTACH statement must set up the default alias RDB\$DBHANDLE.

The following file attaches to three Oracle Rdb databases:

```
$ type multi.sql
```

```
attach 'filename mf_personnel';
attach 'alias a filename a_personnel';
attach 'alias b filename b_personnel';
```

OCI Services for Oracle Rdb uses the following syntax conventions when executing a command file:

- Leading and trailing space on a line is ignored.
 - Comments start with two consecutive hyphens (- -). The comment lines must start at the beginning of a line and continue to the next new line.
 - Each SQL statement must be able to be dynamically prepared, executed, and released by the SQL EXECUTE IMMEDIATE statement.
 - SQL statements cannot span multiple lines.
 - A trailing semicolon (;) at the end of the SQL statement is ignored to allow SQL files to be invoked and verified using interactive SQL.
- A simple file specification

An ATTACH statement is built around this file specification. The following example shows a TNSNAMES.ORA file with the RDB_DATABASE field specified in the connect string:

```
RARE_DB = (DESCRIPTION=
          (ADDRESS=
            (PROTOCOL=TCP)
            (HOST=SQS_NODE)
            (PORT=1234)
          )
          (CONNECT_DATA=(SERVICE=OCI_UNIV)
            (RDB_DATABASE=disk:[dev]db.rdb)
          )
        )
```

4.5 Using the RDB_NATCONN71.COM Command File

The RDB_NATCONN71.COM command file allows you to prepare or upgrade an Oracle Rdb database for use by OCI Services for Oracle Rdb, or to remove OCI-associated tables and functions from an Oracle Rdb database. You can also use the command file to add, modify, remove, or show users with encrypted passwords in a prepared Oracle Rdb database.

4.5.1 Preparing a Database

The PREPARE command in the RDB_NATCONN71.COM file is designed for the initial configuration of the emulated Oracle data dictionary. Use this operation for databases that have not been prepared for OCI Services for Oracle Rdb. The PREPARE operation must be run on each database that is to be accessed by OCI Services for Oracle Rdb.

```
$ @SYS$LIBRARY:RDB_NATCONN71 PREPARE database [60 | 6i]
```

You must specify 60 or 6i if you intend to use Oracle Developer as a client.

Note: The PREPARE operation supercedes the use of SQL prepare scripts that were used with prior versions of OCI Services for Oracle Rdb.

4.5.2 Upgrading a Database

Use the UPGRADE command in the RDB_NATCONN71.COM command file to upgrade the required metadata objects created with the PREPARE operation.

```
$ @SYS$LIBRARY:RDB_NATCONN71 UPGRADE database
```

This command must be used after you upgrade OCI Services for Oracle Rdb to a new release if the database was already prepared with a prior release.

Note: The UPGRADE operation supercedes the use of SQL upgrade scripts that were used with prior versions of OCI Services for Oracle Rdb.

4.5.3 Removing OCI Services for Oracle Rdb

If you need to remove the OCI Services for Oracle Rdb objects after you have completed a PREPARE or UPGRADE operation, use the following command to remove the definitions:

```
$ @SYS$LIBRARY:RDB_NATCONN71 DROP database
```

This command deletes all the tables, views, domains, stored procedures and external procedures installed in your database for OCI Services for Oracle Rdb. If you have user-defined references to any of these objects, the DROP command will fail.

4.5.4 Adding Users and Passwords

Most Oracle tools require that user names and their passwords be stored in the database (USER\$ table) in order for the tools to properly connect to the database. The ADD_USER command used in the RDB_NATCONN71.COM command file allows a DBA, or someone with SELECT privilege on the database, to add users and their associated passwords to the USER\$ table of a database.

```
@SYS$LIBRARY:RDB_NATCONN71 ADD_USER username new_password database
```

There are three parameters associated with the ADD_USER operation:

- The new user name
- The password associated with the new user name
- The database to which you want to attach

If a value is not specified for <new_password>, the user is added with an invalid password, allowing the user to subsequently modify the password. If you do not specify the remaining parameters, the command file prompts for them.

Note: The ADDUSER keyword is deprecated and will not be supported in a future release.

4.5.5 Modifying Passwords

Use the MODIFY_USER command in the RDB_NATCONN71.COM command file to modify the password for a given user in a list of databases specified in a previously created file.

```
$ @SYS$LIBRARY:RDB_NATCONN71 MODIFY_USER username new_password old_password
```

A data file containing a list of all Oracle Rdb databases on the system must be created prior to executing the MODIFY_USER operation in the RDB_NATCONN71.COM command file. The command file reads the file and updates the user's password in each database where the user has previously been entered. The default specification for the file is SYSS\$MANAGER:SQLSRV_NATCONN_DBS.DAT.

The logical SQLSRV_NATCONN_DBS can be defined to override the location and name of the data file. If the file does not exist in SYSS\$MANAGER, or the logical does not point to a valid file, the program looks in the current directory for a file named SQLSRV_NATCONN_DBS.DAT.

The data file is a text file with one line for each database name. It must contain the fully qualified database file specification. An exclamation point (!) can be used to designate a comment; the remainder of the line after the exclamation point will be ignored.

There are three parameters associated with the MODIFY_USER operation:

- The user name
- The new password to be associated with the existing user name
- The old password for the user name

There are four scenarios under which the MODIFY_USER operation can be invoked. The first two apply to the current user, who is executing the program, and require no privileges. The last two scenarios apply to the system manager or database administrator and require special privileges.

Scenario 1:

```
$ @SYS$LIBRARY:RDB_NATCONN71 MODIFY_USER " " new_password
```

Use this scenario when a user's OpenVMS password has been changed, but the new password has not yet been stored in the databases the user accesses. When the command file is called with only the new_password parameter, the user name for the current process is assumed. The new password is checked against the system User Authorization File (UAF). If it is valid, then the password is modified in each database listed in the SQLSRV_NATCONN_DBS data file where this user has already been entered.

Scenario 2:

```
$ @SYS$LIBRARY:RDB_NATCONN71 MODIFY_USER " " new_password old_password
```

Use this combination of parameters to update a user's password in both the OpenVMS system UAF and in the databases the user accesses. When you do not specify the user name, the user name for the current process is assumed. The old password is checked against the system UAF. If it is valid, then the OpenVMS account password is set to the new password. The password is then modified in each database listed in the SQLSRV_NATCONN_DBS data file where the user has already been entered.

Scenario 3:

```
$ @SYS$LIBRARY:RDB_NATCONN71 MODIFY_USER username new_password old_password
```

This scenario can be used by a system manager or database administrator to update another user's password both in the OpenVMS system UAF and in the databases that the user accesses. The manager must possess the appropriate privileges (for example, BYPASS or SYSPRV) in order to make this change.

If the old password is valid, the OpenVMS account password for the given user name is updated using the new password. The password is then modified in each database listed in the SQLSRV_NATCONN_DBS data file where this user has already been entered.

Scenario 4:

```
$ @SYS$LIBRARY:RDB_NATCONN71 MODIFY_USER username new_password ""
```

This scenario can be used by a system manager or database administrator to update another user's password both in the OpenVMS system UAF and in the databases that the user accesses, without verifying the old password. The OpenVMS account password for the given user name is updated using the new password. The manager must have the appropriate privileges for the password to be changed in this scenario (for example, BYPASS or SYSPRV). The password is then modified in each database listed in the SQLSRV_NATCONN_DBS data file where this user has already been entered.

4.5.6 Removing a User

Use the REMOVE_USER command in the RDB_NATCONN71.COM command file to remove a user name from the USER\$ table of the specified database.

```
$ @SYS$LIBRARY:RDB_NATCONN71 REMOVE_USER username database
```

4.5.7 Showing Users

Use the SHOW_USERS command in the RDB_NATCONN71.COM command file to display all user names in the USER\$ table of the specified database.

```
$ @SYS$LIBRARY:RDB_NATCONN71 SHOW_USERS database
```

4.6 Defining Character Sets

OCI Services for Oracle Rdb supports several character sets, including some multibyte character sets.

The character sets shown in [Table 4–1](#) are supported on your server system. To determine which character sets are supported on your client system, refer to your client-specific documentation.

Table 4–1 lists the character sets supported by OCI Services for Oracle Rdb.

Table 4–1 Supported Character Sets

Languages Supported	Oracle Character Set	Oracle Rdb Character Set
Brazilian Portuguese	.WE8DEC	DEC_MCS
Canadian French		
Czechoslovakian		
Danish		
Dutch		
Finnish		
French		
German		
Greek		
Hungarian		
Icelandic		
Italian		
Mexican Spanish		
Norwegian		
Polish		
Portuguese		
Russian		
Slovak		
Spanish		
Swedish		
Turkish		
DEC-Hanzi	ZHS16CGB2312-80	HANZI
Dec-Kanji	JA16VMS	DEC_KANJI
Korean	KO16KSC5601	KOREAN
Extended European	WE8ISO8859P1	ISOLATIN1
Extended European	WE8ISO8859P15	ISOLATIN9
Japanese Shift-JIS	JA16SJIS	SHIFT_JIS
Latin/Arabic (ISO)	AR8ISO8859P6	ISOLATINARABIC
Latin/Cyrillic (ISO)	CL8ISO8859P5	ISOLATINCYRILLIC
Latin/Greek (ISO)	EL8ISO8859P7	ISOLATINGREEK
Latin/Hebrew (ISO)	IW8ISO8859P8	ISOLATINHEBREW
Thai	TH8TISASCII	THAI
Traditional Chinese	ZHT16BIG5	BIG5
Unicode	UTF8	UTF8

Table 4–1 Supported Character Sets

Languages Supported	Oracle Character Set	Oracle Rdb Character Set
GB18030	ZHS32GBI1030	GB18030
Latin/Arabic (8-bit)	AR8MSWIN1256	WIN_ARABIC
Latin/Cyrillic (8-bit)	CL8MSWIN1251	WIN_CYRILLIC
Latin/Greek (8-bit)	EL8MSWIN1253	WIN_GREEK
Latin/Hebrew (8-bit)	IW8MSWIN1255	WIN_HEBREW
West European (8-bit)	WE8MSWIN1252	WIN_LATIN1

By default, OCI Services for Oracle Rdb supports the US7ASCII character set (defined as US7ASCII using the NLS_LANG logical name).

4.6.1 Defining Character Sets on Server Systems

On your server system, use the NLS_LANG logical name to define a character set other than the default US7ASCII character set. The format of the NLS_LANG logical name is as follows:

```
$ define NLS_LANG "[<language>][_<territory>].<character_set>"
```

Note that you cannot change the character set for the session once the session is started.

For example, to specify DEC_MCS, which provides all characters for the Western European languages, define the NLS_LANG logical name to be “.WE8DEC” in the process initialization file for the service.

Example 4–3 specifies only the Western European character set.

Example 4–3 Specifying the Western European Character Set

```
$ define NLS_LANG “.WE8DEC”
```

Example 4–4 specifies the French language and territory and the Western European character set.

Example 4–4 Specifying the French Language and Territory

```
$ define NLS_LANG “FRENCH_FRANCE.WE8DEC”
```

Note: You must specify a language or territory on the client system.

Alternatively, you can specify the language and territory using the Oracle SQL ALTER SESSION statement in your SQL initialization file, for example:

```
ALTER SESSION SET NLS_LANGUAGE=<language> NLS_TERRITORY=<territory>
```

4.6.2 Defining Character Sets on Client Systems

Because of the wide variety of client systems and operating system platforms available, this section cannot describe all the possibilities for defining character sets on client systems. For information about specifying character sets on your client system, see your platform-specific documentation.

4.6.3 Rules and Recommendations

Note the following rules and recommendations when you specify a character set:

- The character set is specific to the session.
 - You can specify only one character set for a given session.
- All character data that is sent to the database or requested from the database is assumed to be in the defined character set.
- If you use a character set other than US7ASCII, specify the desired character set on both the client and server systems.

If the character set on your client system is not compatible with the character set on the server, OCI Services for Oracle Rdb attempts to translate the character set. However, the results of the translation may not be as you expect.

4.7 Referencing an Oracle Rdb Database as a Database Link

Beginning with Oracle SQL/Services release 7.1.2, you can use Oracle SQL to establish a connection to a remote Oracle Rdb database. To define the database link, use the Oracle SQL CREATE DATABASE LINK statement. Once the database link is created, you can reference any table or tables in the Oracle Rdb database, including data dictionary tables. You can join tables from the Oracle Rdb database with each other and with tables in the

Oracle database. The following sections describe how to define a database link and list restrictions for using this feature.

4.7.1 CREATE DATABASE LINK Example

To define a database link to an Oracle Rdb database, connect to your Oracle database server and use the CREATE DATABASE LINK statement. The following line shows the syntax for the CREATE DATABASE LINK statement:

```
CREATE DATABASE LINK <link-name>
  [CONNECT TO username IDENTIFIED BY password]
  USING <connect-name>;
```

In the command line, supply the <link-name> and <connect-name> names as follows:

- The <link-name> parameter must be the service identifier of the Oracle SQL/Services service. The service name must consist of eight or fewer characters.
- The <connect-name> parameter must be the OCI Services for Oracle Rdb connect string from the TNSNAMES.ORA file. The connect string specifies the database or databases to which you want to attach.

For example, assume that your service name is my_serv and the connect string is my_conn as shown in the TNSNAMES.ORA file in [Example 4-5](#).

Example 4-5 TNSNAMES.ORA File

```
my_conn = (DESCRIPTION = (ADDRESS =
  (PROTOCOL = TCP) (HOST = sqs_node) (PORT = 1527) )
  (CONNECT_DATA = (SERVICE = my_serv) )
```

Note: In configurations using a release of the Oracle Network Manager prior to release 3.1.0.5.1, the SERVICE entry in the TNSNAMES.ORA file might be shown instead as SID (service identifier). For example:

```
(CONNECT_DATA = (SID = my_serv)
```

[Example 4-6](#) shows the CREATE DATABASE LINK statement for establishing a connection to the Oracle Rdb my_serv database service.

Example 4-6 Database Link Definition

```
CREATE DATABASE LINK my_serv
  [CONNECT TO username IDENTIFIED BY password]
```



```
USING 'my_conn' ;
```

If the service name you supply is incorrect, the following error message is returned when you attempt to use the database link:

ORA-2085: database link <link_name> connects to <other_name>

In the error message, the variable <other_name> is the service name of the database that you tried to connect to, but the variable does not match the name specified in <link_name>.

4.7.2 Database Link Restrictions

This section describes restrictions for the database links feature:

- The database link name must be the same as the service name to which it is connecting. It cannot exceed eight characters.
- Starting with release 7.1.6, OCI Services for Oracle Rdb provides the ability for any number of Oracle Rdb databases to participate in a two-phase commit transaction with any number of Oracle databases, using the XA Gateway on the OpenVMS system. To enable this functionality, you must still issue the command ALTER SESSION SET TX_MODE NOWARN_IPC. In addition, you must define the logical RDB\$DDTM_XG_INFO to be SYSTEM\$<nodename>, for example:

```
$ DEFINE RDB$DDTM_XG_INFO SYSTEM$MYNODE
```

You must also define the logical SQLNET_RECO_USER to identify the user name and password to be used by the recovery process when it logs on to the service to recover a failed transaction, for example:

```
$ DEFINE SQLNET_RECO_USER "SCOTT/TIGER"
```

- You cannot use standard DML and database links to update the Oracle Rdb database with data from an Oracle database. However, you can use PL/SQL statements in the Oracle database server to update the Oracle Rdb database.
- There is a restriction on Oracle SQL UPDATE and DELETE statements that contain subqueries. All tables referenced in an UPDATE or DELETE statement for the Oracle Rdb database must belong to the Oracle Rdb database.

[Example 4–7](#) shows a valid update statement.

Example 4–7 Valid Oracle SQL UPDATE Statement

```
UPDATE emp@rdb SET sal = sal * 1.1
WHERE deptno=(SELECT deptmp FROM emp@rdb WHERE dname = 'RESEARCH');
```

[Example 4–8](#) shows an update statement that will not work.

Example 4–8 Invalid Oracle SQL UPDATE Statement

```
UPDATE emp@rdb SET sal = sal * 1.1
      WHERE deptno=(SELECT deptno FROM depts WHERE dname = 'RESEARCH');
```

The statement fails because it attempts to update the emp table on the Oracle Rdb database by selecting from the Oracle database. When you try to execute the statement in [Example 4–8](#), Oracle server returns the following error:

ORA-2025: all tables in the SQL statement must be at the remote database

[Example 4–9](#) shows how you can work around this problem:

Example 4–9 Using PL/SQL When DML Does Not Work

```
SQL>
SQL> DECLARE CURSOR acur IS SELECT * FROM dept WHERE dname = 'RESEARCH';
 2   BEGIN
 3     FOR rec IN acur LOOP
 4       UPDATE emp@rdb SET sal = sal * 1.1 WHERE deptno = rec.deptno;
 5     END LOOP;
 6   END;
 7   /
```

PL/SQL procedure successfully completed.

SQL>

4.8 Using SQL EXPORT/IMPORT for OCI Services for Oracle Rdb Tables

Prior to release 7.1.0.4 of Oracle Rdb, OCI Services for Oracle Rdb entities were not preserved through an EXPORT/IMPORT cycle. In order to retain OCI Services for Oracle Rdb functionality after an EXPORT/IMPORT cycle in prior versions of Oracle Rdb, use the following steps:

1. Save a copy of your USER\$ table:
\$ RMU/UNLOAD database USER\$ USER\$.UNL
2. Run the DROP procedure on your database:

```
$ @SYS$LIBRARY:RDB_NATCONN71 DROP db_name
```

3. EXPORT, DROP, and IMPORT your database.

4. Run the PREPARE operation again:

```
$ @SYS$LIBRARY:RDB_NATCONN71 PREPARE db_name
```

5. Restore your old copy of the USER\$ table:

```
RMU/LOAD database USER$ USER$.UNL
```

Starting in Oracle Rdb release 7.1.0.4, the IMPORT command can create objects that reference functions, so the OCI services objects are preserved. There is no need to drop and re-create objects if you are running versions of Oracle Rdb release 7.1.0.4 and higher; however, you still must execute the full EXPORT/IMPORT cycle.

5

SQL ALTER SESSION Statement

This chapter explains how to use the Oracle SQL ALTER SESSION statement to control specific aspects of OCI Services for Oracle Rdb operations.

ALTER SESSION Statement

You can use the Oracle SQL ALTER SESSION statement with OCI Services for Oracle Rdb to:

- Change the values of National Language Support (NLS) parameters
- Change the server logging level
- Change the schema emulation mode

OCI Services for Oracle Rdb supports a subset of the ALTER SESSION SET NLS controls that are supported by the Oracle server. In addition, the ALTER SESSION statement supports controls that are unique to the OCI Services for Oracle Rdb environment.

Environment

You can use the ALTER SESSION statement:

- In an Oracle SQL/Services SQL initialization file
- On the command line if the OCI client has a SQL command line interface

Format

ALTER SESSION

```
SET ISOLATION LEVEL {READ COMMITTED | SERIALIZABLE}  
SET NLS LANGUAGE=nls_value  
SET NLS TERRITORY=nls_value  
SET NLS DATE FORMAT=nls_value  
SET NLS DATE LANGUAGE=nls_value  
SET NLS NUMERIC CHARACTERS=nls_value  
SET NLS ISO CURRENCY=nls_value  
SET NLS CURRENCY=nls_value  
SET NLS SORT=nls_value  
SET SCHEMA EMULATION {STRICT | RELAXED}  
SET SQLNET STRUCTURED DATE TYPES {ON | OFF}  
SET CONSTRAINTS {IMMEDIATE | DEFERRED}  
SET SQLNET TIMESTAMP DATE TYPE {ON | OFF}  
SET TX MODE NOWARN_1PC  
LOG {BRIEF | FULL | OFF}
```

Arguments

SET ISOLATION LEVEL READ COMMITTED

SET ISOLATION LEVEL SERIALIZABLE

Defines the degree to which the read operations of one transaction can be affected by the update operations of other concurrently executing transactions. Isolation levels affect only read/write transactions. Read-only transactions always read from the snapshot file if it is enabled.

For example, you implement the SET ISOLATION LEVEL control in the ALTER SESSION statement, as follows:

```
ALTER SESSION SET ISOLATION_LEVEL SERIALIZABLE
ALTER SESSION SET ISOLATION_LEVEL READ COMMITTED
```

The SET ISOLATION LEVEL argument is a synonym for the Oracle Rdb SQL DECLARE TRANSACTION ISOLATION LEVEL statement. Refer to the *Oracle Server SQL Language Reference Manual* for more information about isolation levels in Oracle and to the *Oracle Rdb Guide to SQL Programming* for more information about isolation levels in Oracle Rdb.

SET NLS keyword = nls_value

Changes the values of NLS parameters. All the SET NLS keywords are identical in syntax and meaning to Oracle SQL statements. Refer to the *Oracle Server SQL Language Reference Manual* for complete information.

SET SCHEMA EMULATION RELAXED

SET SCHEMA EMULATION STRICT

Allows you to choose between a relaxed or strict schema emulation layer. The schema emulation control is unique to OCI Services for Oracle Rdb.

The schema emulation layer is mostly transparent. However, because it is an emulation layer and not an exact implementation of the Oracle multischema model, you may encounter compatibility problems with some OCI clients. For this reason, OCI Services for Oracle Rdb provides two schema emulation modes, STRICT and RELAXED:

- **STRICT** schema emulation mode

This is the default mode in which tables and views that you create using an explicit schema that differs from the current schema are recorded in the ORA_OBJECTS table. Each row in the ORA_OBJECTS table defines a database object and the schema to which it belongs. In order to create a table or view outside your schema in STRICT schema emulation mode, you must have write access to the ORA_OBJECTS table,

because Oracle requires such privileges to create database objects outside your schema in an Oracle environment.

In addition, OCI Services for Oracle Rdb verifies references to tables and views that include a schema other than the current schema while in STRICT schema emulation mode. If the specified object does not belong to the specified schema, OCI Services for Oracle Rdb generates an error condition.

By default, tables are in the schema which is named after the user who created it.

- RELAXED schema emulation mode

OCI Services for Oracle Rdb does not record created tables and views in the ORA_OBJECTS table. All tables and views are created in your current schema and write access to ORA_OBJECTS is not required. OCI Services for Oracle Rdb does not verify references to tables and views that include a schema.

By default, tables are in the RDB_SCHEMA schema.

SET SQLNET_STRUCTURED_DATE_TYPES ON | YES

SET SQLNET_STRUCTURED_DATE_TYPES OFF | NO

Allows true data types to be returned to the OCI client. The following table shows the data types that are returned to the client.

Oracle Rdb Data Type	Data Types Returned to Client	
	SQLNET_STRUCTURED_DATE_TYPES = ON or YES	SQLNET_STRUCTURED_DATE_TYPES = OFF or NO
DATE	DATE	DATE
TIME	DATE	DATE
TIMESTAMP	TIMESTAMP	DATE
INTERVAL YEAR TO MONTH	INTERVAL YEAR TO MONTH	CHAR
INTERVAL DAY TO SECOND	INTERVAL DAY TO SECOND	CHAR

When this argument is set to ON or YES and the OCI client specifies structured data types, structured data types are returned to the client.

SET CONSTRAINTS IMMEDIATE

SET CONSTRAINTS DEFERRED

Sets the constraint setting.

If **IMMEDIATE** is specified, during this transaction all constraints defined as **DEFERRABLE INITIALLY DEFERRED** are evaluated as though defined as **DEFERRABLE INITIALLY IMMEDIATE**.

If **DEFERRED** is specified, all constraints defined as **DEFERRABLE INITIALLY DEFERRED** are evaluated as originally specified in the constraint definition.

SET SQLNET_TIMESTAMP_DATE_TYPE YES | ON

SET SQLNET_TIMESTAMP_DATE_TYPE NO | OFF

Allows you to use a **TIMESTAMP** data type on OpenVMS. A value of **YES** or **ON** means that an Oracle Rdb data type of **TIME** or **TIMESTAMP** is returned as a **TIMESTAMP** value. If the default of **OFF** is retained, or if the argument has been set to **OFF** or **NO**, a **DATE** data type is returned.

SET TX_MODE NOWARN_1PC

Allows two-phase commit from a dblink in an Oracle database to a single Oracle Rdb database. This argument instructs OCI Services for Oracle Rdb to join the distributed transaction and to allow the Oracle database to coordinate the transaction. This argument is also required if you are using XA to allow two-phase commit with more than one Oracle Rdb database.

LOG BRIEF

LOG FULL

LOG OFF

Enables or disables logging of information in the Oracle SQL/Services log file. You can use OCI Services for Oracle Rdb logging to see which Oracle SQL statements are being sent from the client to the server. This **ALTER SESSION** argument is unique to OCI Services for Oracle Rdb.

The default mode is to perform **FULL** logging during logon processing. Logging is turned **OFF** by default for the remainder of the session. When logging is **OFF**, OCI Services for Oracle Rdb does not record processing information in the server log.

The following list describes the **BRIEF** and **FULL** logging modes:

- **BRIEF** logging mode provides only the most critical information needed to diagnose problems and understand how OCI Services for Oracle Rdb interacts with the client. Brief logging records the following information in the server log:
 - All SQL statements requested by the client
 - Client SQL statements after OCI Services for Oracle Rdb has performed modifications
 - Server error messages

- Row fetch count

BRIEF logging is usually sufficient to diagnose user problems or to better understand which Oracle SQL statements are generated by client applications.

- FULL logging provides a great quantity of information in the OCI Services for Oracle Rdb log file. FULL logging includes all the information included with BRIEF logging and in addition:
 - All internal SQL statements generated and executed by OCI Services for Oracle Rdb
 - OCI protocol events, such as parse, describe ,and execute
 - Reloading of OCI Services for Oracle Rdb internal cache
 - Synchronization of the ORA_OBJECTS table with the RDB\$RELATIONS table
 - Schema emulation information, including schema_name.object_name references
 - SQLDA information used to communicate between OCI Services for Oracle Rdb and Oracle Rdb SQL

FULL logging is the best source of information when you need to diagnose a client-side problem occurring with OCI Services for Oracle Rdb. Also, if you need to submit a problem report to Oracle, you should include a full session log file with your problem report.

Note: The form and content of the server log file is subject to change.

Usage Notes

- Use of OCI Services for Oracle Rdb with multischema Oracle Rdb databases is not supported at this time.
- Debug flags can be used to log additional information in the SQL/Services executor log file. Enable debug flags by defining the logical SQLNET_DEBUG_FLAGS in your SQL Services service process initialization file.
- While Oracle Rdb supports the ANSI multischema database model, the majority of Oracle Rdb databases that might be accessed with OCI applications through OCI Services for Oracle Rdb exist in single schema form. Moreover, the minority of Oracle Rdb databases that do exist in multischema form are unlikely to contain a schema configuration that is compatible with the typical Oracle environment. Therefore, OCI Services for Oracle Rdb provides a strict or relaxed schema emulation layer.

- The schema emulation layer allows OCI client applications to operate with single schema Oracle Rdb databases as though the Oracle Rdb database contained a schema configuration typical of that found in an Oracle database. The schema emulation layer provides a virtual schema environment similar to that of Oracle.
- In addition, the OCI Services for Oracle Rdb data dictionary provides views and tables that emulate the predefined Oracle schemas and schema objects. However, you cannot use the same name for two different database objects in different schemas as you can with Oracle.

To present this restriction to OCI clients in a way that is most like an Oracle environment, the schema emulation layer implicitly defines a private synonym within the current schema to each object in the database. If you were to define private synonyms in the current schema in an Oracle environment, you would encounter the same unique name requirement as with OCI Services for Oracle Rdb.

- If you choose the STRICT schema emulation mode, when you log on or delete a table or view, OCI Services for Oracle Rdb attempts to resynchronize the ORA_OBJECTS table with RDB\$RELATIONS, as follows:
 - If ORA_OBJECTS references a database object that no longer exists in RDB\$RELATIONS, OCI Services for Oracle Rdb attempts to correct ORA_OBJECTS by deleting the obsolete records from ORA_OBJECTS.
 - If you do not have write access to ORA_OBJECTS (similar to not having the Oracle privileges to delete objects outside your schema), OCI Services for Oracle Rdb generates an error condition and rolls back the current transaction.
 - If you were attempting to delete a table or view, the effect of the rollback is to restore the table or view because you do not have sufficient privilege to delete it.
 - If you were attempting to log on, the logon attempt fails.

This condition occurs only if the ORA_OBJECTS table or database objects are manipulated outside OCI Services for Oracle Rdb. The condition is corrected automatically when a user who has write access to the ORA_OBJECTS table logs on to the system.

Management Commands

This chapter describes the syntax and semantics of the `SQLSRV_MANAGE` utility of Oracle SQL/Services. This utility is used to manage the Oracle SQL/Services server and its components. See [Section 6.1](#) for a description of syntax conventions.

See the Oracle SQL/Services Manager GUI help for information on using this utility.

The `SQLSRV_MANAGE` commands include management commands, environment commands, and switches. [Section 6.2](#) describes how the `SQLSRV_MANAGE` management commands work.

6.1 Syntax Conventions

The `SQLSRV_MANAGE` utility uses the following syntax conventions and semantics for both its environment and management commands:

[]	Brackets enclose optional clauses from which you can choose none, one, or more of the enclosed options. Do not include brackets in your option.
{ }	Braces indicate that you must choose at least one of the enclosed options. Do not include braces in your option.
	The vertical bar means that you can select only one of the options shown.
,	The comma means that you can choose as many of the options shown as you like, separating your choices with commas to be typed as part of the command.
< >	Angle brackets enclose user-supplied names.
::=	An argument followed by a double colon and equal sign represents the definition of the argument.
White space and new lines	White space and new lines (carriage returns) are not significant in the syntax diagram.

Keywords	Keywords are not case sensitive. Keywords are presented in uppercase characters and are underlined.
...	Horizontal ellipsis points in commands mean that parts of the command not directly related to the example have been omitted.
;	All statements must be terminated with a semicolon (;) with the exception of the EXIT and HELP commands, in which the semicolon is optional.

The following syntax and semantics are also used.

<identifier>

An <identifier> is a string starting with a letter and composed of letters (a to z, A to Z), numbers (0 to 9), hyphens (-), and underscores (_). For example:

```
AARDVARK_1-1101
```

<quoted-string>

A <quoted-string> can use either single or double quotation marks containing any characters within it except a new line character. For example:

```
'user1'  
"Today is 5/6/94"
```

Single-quoted strings can contain embedded double quotation marks, and double quoted strings can contain embedded single quotation marks. For example:

```
'Contestant number three said, "My name is Data"  
"Today's beach report is 'sunny and warm'"
```

A new line character inside a string is assumed to be a syntax error; that is, an unterminated quoted string.

Quoted strings are also useful for representing strings that start with a number. For example:

```
'71_user'
```

<number>

A <number> is an integer. It can start with a plus or minus sign and can consist of one or more numbers from 0 to 9. Numbers can be represented either in decimal or hexadecimal format. To represent a number in hexadecimal format, precede the numeric value with the value '0x' or '0X'. For example:

```
0x0000088a
```

<version-data-type>

A <version-data-type> is a software version number with a major and minor version number consisting of one or more numbers from 0 to 9, separated by a decimal point. The major version number is to the left of the decimal point and the minor version number is to the right of the decimal point. The syntax is as follows:

```
n[nnn...].n[nnn...]
```

For example:

```
7.1  
6.10
```

Comments

Comments start with two consecutive hyphens (--) and continue to the next new line. For example:

```
-- This is a comment line.
```

Order of Command Arguments

The order of the command arguments of the management commands is not important. If you enter a command that contains two or more arguments, the arguments do not need to be in the order presented in the format description of that command.

Use of Underscores Between Keywords in Arguments

On the command-line interface, a space can replace the underscore between any keywords in arguments. For example, rather than enter the two keywords NETWORK_PORT (with the underscore separator), you can enter NETWORK PORT (with a space separator) on the command line, and the SQLSRV_MANAGE utility correctly parses these two keywords without returning an error.

SQL Initialization Files

SQL initialization files use the following syntax conventions:

- Leading and trailing white space on a line is ignored.
- Comments start with two consecutive hyphens (--), must start at the beginning of a line, and continue to the next new line.
- Each SQL initialization statement must be able to be dynamically prepared, executed, and released by the SQL EXECUTE IMMEDIATE statement. Refer to the EXECUTE IMMEDIATE statement in the *Oracle Rdb SQL Reference Manual* for more information.

- SQL statements cannot span multiple lines.
- A trailing semicolon (;) at the end of a SQL statement is ignored to allow SQL initialization files to be invoked and verified using interactive SQL.

The following example illustrates a sample SQL initialization file:

```
--  
-- This SQL initialization file sets the SQL dialect and default  
-- character set for an executor process.  
--  
SET DIALECT 'SQL89';  
SET DEFAULT CHARACTER SET 'KANJI';
```

6.2 How SQLSRV_MANAGE Commands Work

This section describes how SQLSRV_MANAGE commands work.

Server Configuration Commands

The following commands operate on the server, dispatcher, and service objects in a server configuration:

- ALTER SERVER, CONNECT TO SERVER, CREATE SERVER, DISCONNECT SERVER, DROP SERVER, EXTRACT, RESTART SERVER, SET CONFIG_FILE, SET CONNECTION, SHOW SERVER, SHOW SETTINGS, SHOW VERSION, SHUTDOWN SERVER, START SERVER
- ALTER DISPATCHER, CREATE DISPATCHER, DROP DISPATCHER, SHOW DISPATCHER, SHUTDOWN DISPATCHER, START DISPATCHER
- ALTER SERVICE, CREATE SERVICE, DROP SERVICE, GRANT USE ON SERVICE, KILL EXECUTOR, REVOKE USE ON SERVICE, SHOW CLIENTS FOR SERVICE, SHOW SERVICE, SHUTDOWN SERVICE, START SERVICE

Environment Use Commands and Switches

The following commands operate on the SQLSRV_MANAGE system management environment:

- -input and -output switches
- SHOW CONNECT[ION], SHOW SETTINGS
- CONNECT [TO] SERVER, DISCONNECT SERVER, SET CONFIG_FILE, SET CONNECTION
- @, CLOSE, EXIT, HELP, OPEN, SET CONFIRM, SET OUTPUT, SET VERIFY

Table 6–1 describes the three different groups of Oracle SQL/Services objects and shows how each object is acted upon by a set of command verbs.

Table 6–1 Oracle SQL/Services Objects and How Each Object Is Acted Upon by a Command

Object	Command	Description
Dispatcher	ALTER	Change a dispatcher object definition in the configuration file and dynamically change selected attributes for a running server.
	CREATE	Create a dispatcher object for the current server and add the definition to the configuration file.
	DROP	Delete a dispatcher object definition for an inactive dispatcher for the current server from the configuration file.
	SHOW	Show a dispatcher object definition.
	SHUT[DOWN]	Shut down the specified dispatcher object.
	START	Start a dispatcher process for the defined dispatcher object for the current server.
Server	ALTER	Change a server object definition in the configuration file and dynamically change selected attributes for a running server.
	CONNECT	Connect to a running server.
	CREATE	Create a configuration file and a server object.
	DISCONNECT	Disconnect from a running server.
	DROP	Delete a server object definition and delete the configuration file for an inactive server.
	EXTRACT	Extract the definitions for servers, dispatchers, and services, and write them to a SQL/Services command script.
	RESTART	Restart the server including all automatically started dispatchers and services for the current server object.

Table 6–1 Oracle SQL/Services Objects and How Each Object Is Acted Upon by a Command

Object	Command	Description
	SET CONFIG_FILE	Set the current configuration so subsequent commands can modify a server's configuration file.
	SET CONNECTION	Set the connection to the server object with the specified connection name.
	SHOW	Show the server object definition.
	SHOW SETTINGS	Show the current configuration file.
	SHOW VERSION	Show the version of the SQLSRV_MANAGE management client.
	SHUT[DOWN]	Shut down the current server object.
	START	Start the server, including all automatically started dispatcher and executor processes for the current server object.
Service	ALTER	Change a service object definition in the configuration file and dynamically change selected attributes for a running service.
	CREATE	Create a service object and add the definition to the configuration file.
	DROP	Delete a service object definition from the configuration file for an inactive service.
	GRANT USE ON	Grant the USE privilege descriptor for a service object to a user name or identifier.
	KILL EXECUTOR	Kill an executor process.
	REVOKE USE ON	Revoke the USE privilege descriptor for a service object from a user name or identifier.
	SHOW CLIENTS	Show the active users of a service.
	SHOW	Show a service object definition including the USE privilege descriptor for a service object for all user names and identifiers.
	SHUT[DOWN]	Shut down the specified service object.
	START	Start the specified service object.

Table 6–2 describes the SQLSRV_MANAGE environment commands and switches.

Table 6–2 SQLSRV_MANAGE Environment Commands and Switches

Command or Switch	Description
-input switch	Specify the name of an input file from which the SQLSRV_MANAGE utility reads input.
-output switch	Specify the name of an output file to which the SQLSRV_MANAGE utility writes output.
@	Run an indirect command file.
CLOSE	Close an output file.
CONNECT [TO] SERVER	Connect to a running server.
DISCONNECT SERVER	Disconnect from a running server.
EXIT	Exit the SQLSRV_MANAGE utility.
HELP	Get help on a topic.
OPEN	Open an output file.
SET CONFIG_FILE	Set the current configuration so that subsequent commands can modify a server's configuration file.
SET CONFIRM	Require confirmation for certain management operations.
SET CONNECTION	Change the current connection to a server to another connection from among a group of established connections.
SET OUTPUT	Direct output to the default device when enabled.
SET VERIFY	Display command file input on the default output device as it is read.
SHOW CONNECT[ION]	Show information about the current server object and all of the active connections that SQLSRV_MANAGE has to servers.
SHOW SETTINGS	Show information about the verify and output settings.
SHOW VERSION	Show the version of the SQLSRV_MANAGE management client.

–input Switch

Specifies the name of the input file from which the SQLSRV_MANAGE utility reads input.

Format

```
–i[n[put]]          <file-spec>

                    <file-spec> ::= <identifier> or <quoted-string>
```

Arguments

<file-spec>

The input file name. The file name is expressed either as an identifier or as a quoted string.

Usage Notes

- –i and –in are synonyms for the –input command.
- The SQLSRV_MANAGE utility does not prompt for input, and exits when the specified file is completely read.
- You cannot enter the –input switch at the SQLSRV prompt.

Examples

Example 1: Specify an input file from which the SQLSRV_MANAGE utility reads input.

```
$ sqlsrv_manage ::= $SYS$SYSTEM:sqlsrv_manage71
$ sqlsrv_manage -input sqlsrv_create.sqs
```

-output Switch

Specifies the name of the output file to which the SQLSRV_MANAGE utility writes output.

Format

`-o[ut[put]]` <file-spec>

 <file-spec> ::= <identifier> or <quoted-string>

Arguments

<file-spec>

The output file name. The file name is expressed either as an identifier or as a quoted string.

Usage Notes

- `-o` and `-out` are synonyms for the `-output` switch.
- The SQLSRV_MANAGE utility writes all output to the specified file until a CLOSE or OPEN command is executed. If a CLOSE command is issued, subsequent output is sent to standard output. If an OPEN command is issued, output is sent to the new output file.
- You cannot enter the `-output` switch at the SQLSRV prompt.

Examples

Example 1: Specify an output file to which the SQLSRV_MANAGE utility writes output.

```
$ sqlsrv_manage -output out_testfile
```

@ Command

Runs an indirect command file in the SQLSRV_MANAGE environment.

Format

@ <file-spec>;

<file-spec> ::= <identifier> or <quoted-string>

Arguments

<file-spec>

The indirect command file name. The file name is expressed as either an identifier or as a quoted string.

Usage Notes

When executed, the indirect command file is opened and input is taken from that file until either a syntax error occurs or there are no more characters in the file.

Examples

Example 1: Run an indirect script named test_file.sqs. Use a quoted string if it is important to preserve case.

```
SQLSRV> @ 'test_file.sqs';
```

ALTER DISPATCHER Command

Changes a dispatcher object definition for the current server only. Changes to a dispatcher definition are stored in the configuration file. Offline dispatcher changes do not affect a running server. Online dispatcher changes affect the running server if the change is to a dynamic attribute; otherwise, the dispatcher must be shut down and started again or the server restarted for dispatcher changes to take effect.

Format

```
ALTER DISPATCHER <disp-name>
->[ AUTOSTART { ON | OFF } ]
->[ MAX_CONNECTIONS <number> ]
->[ IDLE_USER_TIMEOUT <number-in-seconds> ]
->[ MAX_CLIENT_BUFFER_SIZE <number> ]
->[ DUMP_PATH <directory-specification> ]
->[ LOG_PATH <directory-specification> ]
->[ <network-port-spec> ] ...;

<disp-name> ::= <identifier>
<network-port-spec> ::= NETWORK_PORT <transport-spec>
    PROTOCOL { NATIVE | OCI | SQLSERVICES | JDBC }
<transport-spec> ::= { <tcp-spec> | <decnet-spec> | <ipxspx-spec>
    | sqlnet-spec }
<tcp-spec> ::= TCPIP [ PORT_ID <number> ]
<decnet-spec> ::= DECNET [ OBJECT { <number> | <identifier>
    | <quoted-string> } ]
<ipxspx-spec> ::= IPXSPX [ <PORT_ID <number> ]
<sqlnet-spec> ::= SQLNET_LISTENER_NAME { <identifier>
    | <quoted-string> }
```

Arguments

<disp-name>

The dispatcher name. The dispatcher name is expressed as an identifier.

AUTOSTART {ON | OFF}

Determines whether or not the dispatcher object automatically starts up when you issue a START SERVER or RESTART SERVER command. If the argument is specified as ON, the dispatcher object automatically starts when you issue a START SERVER or RESTART SERVER command. The default is ON.

MAX_CONNECTIONS <number>

Specifies the maximum number of network connections from clients that the dispatcher accepts. The maximum number of connections is expressed as an integer. The default is 100. There is no upper limit other than the operating system configuration, the network configuration, and shared server memory.

IDLE_USER_TIMEOUT <number-in-seconds>

Specifies the amount of time in seconds that a client (user) can remain idle before the dispatcher disconnects the client. The <number-in-seconds> value is expressed as an integer. The default value is 0, which displays as "<none>" in a SHOW DISPATCHER command and means that the idle timeout value is infinite. A value specified other than 0 is rounded to the next higher multiple of 90 seconds. This is a dynamic attribute that, when changed, takes effect immediately.

MAX_CLIENT_BUFFER_SIZE <number>

Specifies the maximum client buffer size permitted. The maximum allowed client buffer size is 32,000 bytes. If a client application specifies a buffer size larger than the maximum, then the Oracle SQL/Services client API adjusts the buffer size to the maximum size specified for the dispatcher. The default and minimum value allowed for the MAX_CLIENT_BUFFER_SIZE attribute is 5000 bytes.

DUMP_PATH <directory-specification>

Specifies a directory name for bugcheck dump files.

LOG_PATH <directory-specification>

Specifies a directory name for log files.

If you specify NOLOG instead of a directory name for the LOG PATH argument, no log file is written.

<network-port-spec>

Lists network ports that the dispatcher should use for communications with clients. The network port specification is any one or any combination of the following: TCP/IP, DECnet, IPX/SPX, and Oracle Net. The default port ID for TCP/IP is 118, the default DECnet object is 81, and the default port ID for IPX/SPX is 33969 (0x84b1). If the network port is not specified, the dispatcher will use the default ports. The <network-port-spec> argument can be repeated to include multiple OCI Services for Oracle Rdb listener names. The maximum number of times that the <network-port-spec> argument can be specified in the ALTER DISPATCHER command is five.

This argument also determines the message protocol that each dispatcher network port can support. A dispatcher network port can support only one message protocol. Specify a message protocol that matches the type of client you want a dispatcher network port to support:

- NATIVE
Oracle RMU clients (Oracle Rdb Performance Monitor, Oracle RMUwin, Parallel Backup Monitor)
- OCI
Oracle clients using the Oracle Call Interface (OCI) or Oracle Enterprise Manager clients
- SQLSERVICES
Oracle SQL/Services clients (Query Performance Tuner, Oracle ODBC Driver for Rdb, Oracle SQL/Services, Oracle Rdb Schema Manager)
- JDBC
JDBC for Oracle Rdb clients

Note: Ensure that you have a dispatcher network port defined with a dispatcher message protocol that supports each service API you want to use. For example, if you define a service that supports the OCI API and another service that supports the SQLSERVICES API, you must define at least one dispatcher network port that supports the OCI dispatcher message protocol and the Oracle Net transport and another dispatcher network port that supports the SQLSERVICES dispatcher message protocol and any available transport, respectively.

Usage Notes

- In general, any specified clauses in the ALTER DISPATCHER definition replace the specification of items in the previous dispatcher definition. That is, if a clause is specified in the ALTER DISPATCHER command, then the specification of items for that clause is changed in the definition. If no clause is specified, the specification of items remains unchanged for that clause.
- If a network port is altered, the entire network port specification is replaced. Thus, you can add a network port to the existing list with the ALTER DISPATCHER command, but you must respecify all other network port specifications to retain them in the configuration file.
- To use the Oracle Net transport option, specify the Oracle Net transport option as `<sqlnet-spec>` in the `<transport-spec>` argument and specify the OCI Services for Oracle Rdb listener name as its `<identifier>` argument.
- The word LISTENER is a synonym for the keyword LISTENER_NAME.
- SQLSRV_MANAGE lets you create two or more dispatchers listening on the same port ID or object, but only the first dispatcher with a unique port ID or object is allowed to start. If you attempt to start a second dispatcher listening on the same port ID or object, it fails to start if it cannot listen on any of the specified network ports.
- Oracle recommends that you create a log file for troubleshooting purposes unless you have a problem with excessive I/O entries in the log file. The default directory for the LOG PATH and DUMP PATH arguments is SYSS\$MANAGER.
- The LOG PATH and DUMP PATH arguments are available when you use the SQLSRV_MANAGE command utility, but not when you use the Oracle SQL/Services Manager GUI. You are able to view the log files and bugcheck dump files when you use the Oracle SQL/Services Manager GUI if the sqlsrv_manage service is running.
- If values are assigned to existing logicals SQLSRV_DISP_LOGPATH and SQLSRV_DISP_DUMPPATH, they override log path and dump path values specified by the LOG PATH and DUMP PATH arguments.

Examples

Example 1: Dynamically alter the idle user timeout value.

```
SQLSRV> ALTER DISPATCHER tcpip_disp IDLE_USER_TIMEOUT 180;
```

Example 2: Alter a dispatcher to use the OCI Services for Oracle Rdb protocol. This command removes all other ports for this dispatcher. You must respecify all existing network ports to prevent the loss of previously defined network ports for this dispatcher.

```
SQLSRV> ALTER DISPATCHER OCI_disp NETWORK_PORT SQLNET LISTENER_NAME
"OCI-LISTENER";
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect
SQLSRV> SHUTDOWN DISPATCHER OCI_disp;
SQLSRV> START DISPATCHER OCI_disp;
```

Example 3: Specifying log path and dump path.

```
SQLSRV> ALTER DISPATCHER SQLSRV_DISP1 LOG PATH 'USER1:[SQLSRV_TEST1.AAA]'
_SQLSRV> DUMP PATH 'USER1:[SQLSRV_TEST2.BBB]';
%SQLSRV-S-ALTER_RESTART, Restart object to have altered settings take effect
```

ALTER SERVER Command

Changes a server object definition. Changes to a server definition are stored in the configuration file. Offline server changes do not affect a running server. Online server changes affect the running server if the change is to a dynamic attribute; otherwise, the server must be shut down and started again or restarted for changes to take effect.

Format

ALTER SERVER

```

->[ MAX_SHARED_MEMORY_SIZE <number> ]
-->[DUMP_PATH <directory-specification>]
-->[LOG_PATH <directory-specification>]
->[ PROCESS_STARTUP_TIMEOUT <number-in-seconds> ]
->[ PROCESS_SHUTDOWN_TIMEOUT <number-in-seconds> ]
->[ <network-port-spec> ]...;

<network-port-spec> ::= NETWORK_PORT <transport-spec>
<transport-spec> ::= { <tcp-spec> | <decnet-spec> }
<tcp-spec> ::= TCPIP [ PORT_ID <number> ]
<decnet-spec> ::= DECNET [ OBJECT {<number> | <identifier>
| <quoted-string> } ]

```

Arguments

MAX_SHARED_MEMORY_SIZE <number>

Specifies the size in kilobytes of the maximum shared memory the server should use. If the value is changed, that value becomes the maximum shared memory size when the monitor starts up. The default value is 2000 kilobytes or 2 megabytes. Oracle SQL/Services allocates the maximum shared memory size when the monitor starts up.

DUMP_PATH <directory-specification>

Specifies a directory name for bugcheck dump files.

LOG PATH <directory-specification>

Specifies a directory name for log files.

If you specify NOLOG instead of a directory name for the LOG PATH argument, no log file is written.

PROCESS_STARTUP_TIMEOUT <number-in-seconds>

Specifies the length of time in seconds to wait before deciding that a dispatcher or executor process is not going to start up before the monitor takes action and terminates the process. The default value is 0 seconds, which means that no process startup timer value is set. This is a dynamic attribute that, when changed, takes effect immediately. See the Usage Notes for more information.

PROCESS_SHUTDOWN_TIMEOUT <number-in-seconds>

Specifies the length of time to wait in seconds before deciding that a dispatcher or executor process is not going to shut down before the monitor takes action and terminates the process. The default value is 0 seconds, which means that no process shutdown timer value is set; the process shutdown timer value is infinite. This is a dynamic attribute that, when changed, takes effect immediately. See the Usage Notes for more information.

<network-port-spec>

Lists network ports that the monitor should use for communications with Oracle SQL/Services management clients (SQLSRV_MANAGE and the Oracle SQL/Services Manager GUI). The network port specification is TCP/IP or DECnet. The default port ID for TCP/IP is 2199 and the default DECnet object name is SQLSRV_SERVER. If no network ports are specified, the monitor of the server uses the default ports. The maximum number of times that the <network-port-spec> argument can be specified in the ALTER SERVER command is five. If a network port is altered, the entire network port specification is replaced.

DECnet or TCP/IP must be available on the node for which the ALTER SERVER definition is used. If none of these are available, then the server will not run.

Usage Notes

- The server definition can be altered online using the CONNECT [TO] SERVER command or offline if you select its configuration file using the SET CONFIG_FILE command. Online changes for dynamic attributes take effect immediately. When you make an online change of a nondynamic attribute, a status message is returned indicating that you must restart the server to have altered settings take effect. Oracle recommends that you immediately restart the running server after you complete your management session to ensure the overall consistency of the Oracle SQL/Services server. (To restart the running server, issue the RESTART SERVER command.)

- In general, any specified clauses in the ALTER SERVER definition replace the specification of items in the previous server definition. That is, if a clause is specified in the ALTER SERVER command, then the specification of items for that clause is changed in the definition. If no clause is specified, the specification of items remains unchanged for that clause.
- If you want to set process startup and shutdown timers, follow these guidelines:
 - Usually dispatcher and executor processes start up and shut down in a reasonable period of time. Only during an unusual situation would you need to specify nonzero values for the PROCESS_STARTUP_TIMEOUT and PROCESS_SHUTDOWN_TIMEOUT arguments.
 - In heavily loaded systems, it often takes longer for a particular operation to complete. If either process startup or process shutdown is set to a value other than zero and fails for no apparent reason (you have checked other possible causes and have not isolated the problem), set a higher value for the PROCESS_STARTUP_TIMEOUT argument or the PROCESS_SHUTDOWN_TIMEOUT argument to see if that solves the problem.
- The SQLSRV_MANAGE utility attempts to connect to the monitor of the server using the default TCP/IP or DECnet ports. If you change the network port of the server, you must also specify that port explicitly when connecting from the SQLSRV_MANAGE utility.
- If a network port is altered, the entire network port specification is replaced. Thus, you can add a network port to the existing list with the ALTER SERVER command, but you must respecify all other network port specifications to retain them in the configuration file.
- If the same port ID is specified more than once, an error is returned.

Examples

Example 1: Alter a server definition online.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> ALTER SERVER MAX_SHARED_MEMORY_SIZE 4000;  
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect  
SQLSRV> RESTART SERVER;  
Disconnected from Server  
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected
```

Example 2: Alter a server definition offline.

```
SQLSRV> SET CONFIG_FILE 'my_config_file';  
SQLSRV> ALTER SERVER MAX_SHARED_MEMORY_SIZE 4000;  
SQLSRV> RESTART SERVER;  
Connecting to server ...  
Connected
```

ALTER SERVICE Command

Changes a service object definition for the current server only. Changes to a service definition are stored in the configuration file. Offline service changes do not affect a running server. Online service changes affect the running server if the change is to a dynamic attribute; otherwise, the service must be shut down and started again or the server restarted for service changes to take effect.

Format

```

ALTER SERVICE <service-name>
-> [ PROTOCOL { OCI | RMU | SQLSERVICES } ]
-> [ AUTOSTART { ON | OFF } ]
-> [ DEFAULT_CONNECT_USERNAME { <quoted-string> | <identifier> } ]
-> [ DEFAULT_CONNECT_PASSWORD <quoted-string> ]
-> [ REUSE [ SCOPE] [IS] { SESSION | TRANSACTION } ]
-> [ SQL_VERSION { <version-number> | S[TANDARD] } ]
-> [ PROCESS_INITIALIZATION { <quoted-string> | LOGIN } ]
-> [ ATTACH <quoted-string> ]
-> OWNER { <quoted-string> | <identifier> }
-> [ OWNER_PASSWORD <quoted-string> ]
-> [ SCHEMA <quoted-string> ]
-> [ SQL_INIT_FILE <quoted-string> ]
-> [ DATABASE_AUTHORIZATION { [ SERVICE] OWNER
    | [ CONNECT ] USERNAME } ]
-> [ APPLICATION_TRANSACTION_USAGE { SERIAL | CONCURRENT } ]
-> [ IDLE_USER_TIMEOUT <number-in-seconds> ]
-> [ IDLE_EXECUTOR_TIMEOUT <number-in-seconds> ]
-> [ MIN_EXECUTORS <number> ]
-> [ MAX_EXECUTORS <number> ]

```

```
->[ CLIENTS_PER_EXECUTOR <number> ] ;
```

```
<service-name> ::= <identifier>
```

Arguments

<service-name>

The service name. The service name is expressed as an identifier.

PROTOCOL {OCI | RMU | SQLSERVICES}

Determines the application programming interface (API) that each service can support. A service can support only one API. Specify an API that matches the type of client you want a service to support:

- OCI
Oracle or third-party clients using the Oracle Call Interface (OCI)
- RMU
Oracle RMU clients (Oracle Rdb Performance Monitor, Oracle RMUwin, Parallel Backup Monitor)
- SQLSERVICES
Oracle SQL/Services clients (Query Performance Tuner, Oracle ODBC Driver for Rdb, Oracle SQL/Services, Oracle Rdb Schema Manager). This is the default.

Note: Ensure that you have a dispatcher network port defined with a dispatcher message protocol that supports each service API you want to use. For example, if you define a service that supports the OCI API and another service that supports the SQLSERVICES API, you must define at least one dispatcher network port that supports the OCI dispatcher message protocol and the Oracle Net transport and another dispatcher network port that supports the SQLSERVICES dispatcher message protocol and any available transport, respectively.

AUTOSTART {ON | OFF}

Determines whether or not the service object automatically starts up when you issue a START SERVER or RESTART SERVER command. If the argument is specified as ON, the service object automatically starts when you issue a START SERVER or RESTART SERVER command. The default is ON.

DEFAULT_CONNECT_USERNAME {<quoted-string> | <identifier>}

Specifies the user name as a quoted-string or identifier under which unknown users will be allowed to connect to the service. This argument can be applied only to database services that support the SQLSERVICES protocol. See [Section 2.7.1](#) and [Section 2.8](#) for more information about using this argument. This is a dynamic attribute that, when changed, takes effect immediately.

DEFAULT_CONNECT_PASSWORD <quoted-string>

Specifies the password associated with the connect user name as a quoted string.

REUSE SCOPE IS {SESSION | TRANSACTION}**■ SESSION**

An executor for a session reusable service processes requests for one client session at a time. A session begins when a client connects to the service and the connection is bound to an executor process. A session ends when a client disconnects from the service and the connection is unbound from the executor process. This is the default.

■ TRANSACTION

An executor for a transaction reusable service processes requests for one transaction at a time; however, it supports many concurrent client sessions. A transaction begins when a client issues a SQL statement that either implicitly or explicitly starts a transaction. A transaction ends when a client issues a successful SQL COMMIT or ROLLBACK statement. The REUSE SCOPE IS TRANSACTION argument may be applied only to database services that use the SQLSERVICES protocol.

See [Section 2.6](#) for more information.

SQL_VERSION {<version-number>| STANDARD}

Specifies the version of SQL to use for the service. It is expressed as either a version number data type (for example, 7.1) for selecting a version of SQL in an Oracle Rdb multiversion environment or by the keyword STANDARD (or S) for running a standard version of SQL in an Oracle Rdb single version environment. Either value is used as the first parameter argument for the Oracle Rdb RDB\$SETVER command procedure when it runs, as described in the installation information. The version number resolves to an "n.n" parameter argument and the word STANDARD or S resolves to an S parameter argument. When no value is specified, the default is the keyword STANDARD.

PROCESS_INITIALIZATION {<quoted-string> | LOGIN}

The process initialization file can be either a special process initialization file specified as a <quoted-string> or the keyword LOGIN. When LOGIN is specified for the process initialization file, Oracle SQL/Services uses the file specified by the LGICMD qualifier in AUTHORIZE as returned by the OpenVMS SYS\$GETUAI system service. If this file

specification is not fully qualified, the file will not be found and the executor will fail. If you specify process initialization as LOGIN, make sure LGICMD qualifier is defined for the service owner account. The process initialization or login file is used to help define some of the attributes of the executor process for this service. If no argument is specified, the default is not to run any initialization file. This file is executed once for each executor, during executor startup.

ATTACH <quoted-string>

The SQL ATTACH statement.

If you do not specify a SQL ATTACH statement, you create a universal service that is not preattached to a specific database.

If you specify a SQL ATTACH statement, you create a database service that is preattached to the specified database.

This argument is a single-quoted string and is exactly the same format as the attach-string-literal used in dynamic SQL. The FILENAME keyword in this string cannot be abbreviated.

See the *Oracle Rdb SQL Reference Manual* for more information on the ATTACH statement.

OWNER {<quoted-string | <identifier>}

Specifies the user name of the owner of the service. Every service has an owner name. The owner name must be specified as a quoted string or an identifier; otherwise, an error message is returned.

If the service is a database service, then the service owner's privileges are used for access checks when an executor attaches to the specified database. See [Section 2.6](#) for more information on database services.

If the database access authorization is by service owner, then the service owner's privileges are used for all database access operations. See the DATABASE_AUTHORIZATION argument, later in this argument list, for more information on database access authorization.

Executors are created with the privileges and quotas from the account of the service owner. See [Section 2.10.1](#) for more information.

The OWNER argument can be applied only to database services that support the SQLSERVICES protocol.

OWNER PASSWORD <quoted-string>

Specifies the password for the owner of the service.

SCHEMA <quoted-string>

Provides a way to specify the default schema that you want to use when an executor attaches to a multischema database.

If a schema name is not specified in the service definition, the schema name defaults to the service owner account name if the database access authorization is service owner, or to the connect user name if the database access authorization is connect user name (see [Section 2.9](#)).

The schema argument allows the default to be overridden. This argument is ignored if it is supplied on a service that supports OCI connections.

SQL_INIT_FILE <quoted-string>

Specifies a file containing SQL statements that tailor the SQL environment for a client connection. For example, you can set the SQL dialect and default character set by using a SQL initialization file. The statements in a SQL initialization file are executed every time a client connects to a service.

See [Section 6.1](#) for more information about using a SQL initialization file.

DATABASE_AUTHORIZATION {[SERVICE] OWNER | [CONNECT] USERNAME}

Determines the user name under which access to the database is made. The default is CONNECT USERNAME.

- SERVICE OWNER

For a database service, all access to the database is made by using the service owner user name. This option is not supported by OCI Services for Oracle Rdb.

- CONNECT USERNAME

Access to the database is made by using the client-specified user name, the DECnet proxy user name, or the user name specified in the DEFAULT_CONNECT_USERNAME argument.

For more information on database access authorization, see [Section 2.7](#) and [Section 2.8](#).

APPLICATION_TRANSACTION_USAGE {SERIAL | CONCURRENT}

Applies only to transaction reusable database services. Some applications make only a single connection to a service to perform their work, while other applications make multiple connections to the same service. Connections created to transaction reusable database services are tied to the same executor for the life of the session.

If a client application makes multiple connections to a service and these are assigned to the same executor, a deadlock occurs if the client application attempts to start a new transaction on one connection before ending an existing transaction on another connection. When you specify the CONCURRENT keyword, Oracle SQL/Services ensures that multiple

connections from the same client application on the same node are never assigned to the same executor process.

When you specify the SERIAL keyword, Oracle SQL/Services assumes that client applications do not start concurrent transactions on multiple connections. Oracle SQL/Services assigns connections to executor processes on a least busy basis (the executor process with the fewest client connections already assigned). Thus, if a client application made more than one connection to the same service and the keyword SERIAL was specified, the second connection may or may not have gone to the same executor process as the first connection, depending on how many connections were assigned to that executor process versus how many connections were assigned to the other executor processes for that service.

The default for the APPLICATION_TRANSACTION_USAGE argument is SERIAL. This is a dynamic attribute that, when changed, takes effect immediately.

Some applications, such as Microsoft Access, make multiple connections to the same service to perform their work and require that you specify the CONCURRENT keyword. If set to CONCURRENT, Oracle SQL/Services considers the node, user name, and application name of the client when choosing an executor to which to tie the connection and ensures that multiple connections from the same client application are never assigned to the same executor process.

This argument is used only by Oracle SQLSERVICES services.

IDLE_USER_TIMEOUT <number-in-seconds>

Specifies the amount of time in seconds that a client (user) can remain idle before the server disconnects the client. This value is expressed as an integer. The default value is 0, which displays as "<none>" in a SHOW SERVICE command and means that the idle timeout value is infinite. A specified value other than 0 is rounded to the next higher multiple of 90 seconds. This is a dynamic attribute that, when changed, takes effect immediately.

IDLE_EXECUTOR_TIMEOUT <number-in-seconds>

Specifies the amount of time in seconds that an executor process for a session reusable service can remain inactive (not bound to a client connection) before being deleted. The value is expressed as an integer. The default timeout value is 1800 seconds (30 minutes). This is a dynamic attribute that, when changed, takes effect immediately.

MIN_EXECUTORS <number>

Sets the minimum value to which the number of executor processes is allowed to decrease. This is also the number of executor processes started at startup using a START SERVICE or START SERVER command. The value is expressed as an integer. The default minimum number of executors for a session reusable service is 0. A service with MIN_EXECUTORS set to 0 will never show the Starting state when the service starts up. The state will either

display as Running or Failed. This is a dynamic attribute that, when changed, takes effect immediately.

If you use transaction reusable executors, you must set the value for the minimum number of executors so that it is equal to the value for the maximum number of executors. The default value is 1 for a transaction reusable service.

MAX_EXECUTORS <number>

Sets the maximum value to which the number of executor processes is allowed to increase. The value is expressed as an integer. The default maximum number of executors is 1. This is a dynamic attribute that, when changed, takes effect immediately.

If you use transaction reusable executors, you must set the value for the minimum number of executors so that it is equal to the value for the maximum number of executors. The default value is 1 for a transaction reusable service.

CLIENTS_PER_EXECUTOR <number>

Specifies the number of clients allowed per executor. The number of clients allowed is dependent upon whether the service is session reusable or transaction reusable. The default number of clients per executor for session reusability is 1 and cannot be greater than 1. The default number of clients per executor for transaction reusability is 1 but can be greater than 1. The CLIENTS_PER_EXECUTOR value is expressed as an integer. This is a dynamic attribute that, when changed, takes effect immediately.

Usage Notes

- When a service other than an OCI service is created, only a privileged user with SYSPRV privilege is authorized to use the service. You must grant privileges to any other users.
- Values specified for parameters in an ALTER SERVICE command replace values defined in the configuration file and in the running server. However, changes to the running server are not immediate and are as described in the following items:
 - If the value for the minimum number of executors for a session reusable service is decreased, the actual number of executor processes does not decrease until individual executors time out using their current timeout settings.
 - If the value for the minimum number of executors for a transaction reusable service is decreased, the actual number of executor processes does not decrease until the service is shut down and started again.
 - If the value for the maximum number of executors is increased, newly created executor processes succeed where they might have previously reached the limit.

- If the value for the minimum number of executors is increased, new executors are created until the new minimum number of executors is active.
- If the value for the idle executor timeout parameter is changed, the new idle timeout value is used beginning with the next timeout cycle of a given executor.

Examples

Example 1: Alter a transaction reusable database service online to increase the number of clients per executor to 20 and raise the minimum and maximum number of executors to 10. Because these attributes are dynamic attributes, the service need not be shut down and started up again.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> ALTER SERVICE database4  
_SQLSRV> MIN_EXECUTORS 10  
_SQLSRV> MAX_EXECUTORS 10  
_SQLSRV> CLIENTS_PER_EXECUTOR 20;
```

Example 2: Alter a service online to change the SQL_INIT_FILE attribute. Because this attribute is not a dynamic attribute, the service must be restarted for the change to take effect.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> ALTER SERVICE database4  
_SQLSRV> SQL_INIT_FILE 'sql710';  
%DBS-S-ALTER_RESTART, Restart object to have altered settings take affect  
SQLSRV> SHUTDOWN SERVICE database4;  
SQLSRV> START SERVICE database4;
```

CLOSE Command

Closes an output file in the SQLSRV_MANAGE environment.

Format

CLOSE;

Usage Notes

Upon closing an output file, output is directed to standard output.

Examples

Example 1: Close an output file.

```
SQLSRV> CLOSE;
```

CONNECT TO SERVER Command

Connects to a server online so that you can begin managing it.

Format

```

CONNECT [TO] SERVER      [ AS <connect-name> ]
                        ->[ USER { <user-name> USING { <password> }
                        ->[ NODE { <quoted-string> | <identifier> } ]
                        ->[ <network-port-spec> ];

<connect-name> ::= <identifier>
<user-name> ::= { <quoted-string> | <identifier> }
<password> ::= <quoted-string>
<network-port-spec> ::= NETWORK PORT <transport-spec>
<transport-spec> ::= { <tcp-spec> | <decnet-spec> }
<tcp-spec> ::= TCPIP [ PORT_ID <number> ]
<decnet-spec> ::= DECNET [OBJECT { <number> | <identifier>
                        | <quoted-string> } ]

```

Arguments

<connect-name>

The connection name. The identifier that uniquely identifies the connection to a server on a particular node. The connection name is most useful when connecting to more than one server at a time. If you are going to manage only one server, a connection name is not needed. Whenever you create a new connection, it becomes the current connection. To switch to a server that you want to manage among those that you are connected to, use the SET CONNECTION command and specify the connection name of the server.

The connection name is expressed as an identifier.

USER <user-name> USING <password>

Specifies the user name and password of an account that is authorized to manage the server. The user name and password are expressed as either quoted strings or identifiers.

If you are using DECnet or TCP/IP with sufficient privileges to manage a server on the local node, you do not need to enter a user name and password when connecting to the server on a local node. See the Usage Notes for more information on connecting to a server on a local node without specifying a user name and password.

NODE <quoted-string | identifier>

The node where the server is located. By default the node name is the local host name. The node-name is expressed as a quoted string or identifier. It can be used to connect to a remote server.

<network-port-spec>

Lists network ports that the monitor should use for communications with Oracle SQL/Services management clients (SQLSRV_MANAGE and the Oracle SQL/Services Manager GUI). The <network-port-spec> argument is TCPIP or DECNET. The network port specification defaults to TCP/IP with a default port ID of 2199. The default DECnet object is named SQLSRV_SERVER.

Usage Notes

- You must either connect to a server before you can begin managing it online or select the configuration file of the server (SET CONFIG_FILE command) to manage it offline.
- When you establish a new connection to a server using the CONNECT TO SERVER command, the new connection becomes the current connection. All subsequent online system management commands operate on the current connection. Use the SET CONNECT command to switch between connections to multiple servers. Use the DISCONNECT command to disconnect from a server.
- A local user can connect to a server using DECnet without specifying a user name or password. You must have either the SYSPRV or BYPASS privilege to omit the user name and password when connecting to a server using TCP/IP.
- If you are connecting to a local server using the configuration file you currently have open, SQLSRV_MANAGE attempts to connect to any network port defined for the server. It tries each network port in a round-robin fashion up to three times each to establish a management connection. The network port used for the management connection is the first one that is successful.

Examples

Example 1: Connect to a server on the local node as a privileged local user using TCP/IP.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...
```

Connected

Example 2: Connect to a server (user name and password are quoted strings).

```
SQLSRV> CONNECT SERVER USER 'system' USING 'password';  
Connecting to server ...  
Connected
```

CREATE DISPATCHER Command

Creates a dispatcher object definition for the current server. The definition is stored in the configuration file. New dispatcher objects must be started online to be part of a running server. Each dispatcher defined must be listening on a unique set of network ports or objects.

Format

```

CREATE DISPATCHER <disp-name>
->[ AUTOSTART { ON | OFF } ]
->[ MAX CONNECTIONS <number> ]
->[ IDLE USER TIMEOUT <number-in-seconds> ]
->[ MAX CLIENT BUFFER SIZE <number> ]
->[ DUMP PATH <directory-specification> ]
->[ LOG PATH <directory-specification> ]
->[ <network-port-spec> ] ... ;

<disp-name> ::= <identifier>
<network-port-spec> ::= NETWORK PORT <transport-spec>
    PROTOCOL { NATIVE | OCI | SQLSERVICES | JDBC }
<transport-spec> ::= { <tcp-spec> | <decnet-spec>
    | <ipxspx-spec> | <sqlnet-spec> }
<tcp-spec> ::= TCPIP [ PORT ID <number> ]
<decnet-spec> ::= DECNET [ OBJECT { <number> | <identifier>
    | <quoted-string> } ]
<ipxspx-spec> ::= IPXSPX [ PORT ID <number> ]
<sqlnet-spec> ::= SQLNET LISTENER NAME { <identifier>
    | <quoted-string> }

```

Arguments

<disp-name>

The dispatcher name. The dispatcher name is expressed as an identifier. The dispatcher name must be unique.

AUTOSTART {ON | OFF}

Determines whether or not the dispatcher object automatically starts up when you issue a START SERVER or RESTART SERVER command. If the argument is specified as ON, the dispatcher object automatically starts when you issue a START SERVER or RESTART SERVER command. The default is ON.

MAX_CONNECTIONS <number>

Specifies the maximum number of network connections from clients that the dispatcher will accept. The maximum number of connections is expressed as an integer. The default is 100. There is no upper limit other than the operating system configuration, the network configuration, and shared server memory.

IDLE_USER_TIMEOUT <number-in-seconds>

Specifies the amount of time in seconds that a client (user) can remain idle before the dispatcher disconnects the client. This value is expressed as an integer. The default value is 0, which displays as "<none>" in a SHOW DISPATCHER command and means that the idle timeout value is infinite. A specified value other than 0 is rounded to the next higher multiple of 90 seconds.

MAX_CLIENT_BUFFER_SIZE <number>

Specifies the maximum client buffer size permitted. The maximum allowed client buffer size is 32,000 bytes. If a client application specifies a buffer size larger than the maximum, then the Oracle SQL/Services client API adjusts the buffer size to the maximum size specified for the dispatcher. The default and minimum value allowed for the MAX_CLIENT_BUFFER_SIZE attribute is 5000 bytes.

DUMP_PATH <directory-specification>

Specifies a directory name for bugcheck dump files.

LOG_PATH <directory-specification>

Specifies a directory name for log files.

If you specify NOLOG instead of a directory name for the LOG_PATH argument, no log file is written.

<network-port-spec>

Lists network ports that the dispatcher should use for communications with clients. The network port specification is any one or any combination of the following: TCP/IP, DECnet, IPX/SPX, and Oracle Net. The default port ID for TCP/IP is 118, the default DECnet object is 81, and the default port ID for IPX/SPX is 33969 (0x84b1). If no network port is specified, the dispatcher uses the default ports. The <network-port-spec> argument can be repeated to include multiple OCI Services for Oracle Rdb listener names. The maximum number of times that the <network-port-spec> argument can be specified in the CREATE DISPATCHER command is five.

This argument also determines the message protocol that each dispatcher network port can support. A dispatcher network port can support only one message protocol. Specify a message protocol that matches the type of client you want a dispatcher network port to support:

- NATIVE
Oracle RMU clients, for example, Oracle Rdb Performance Monitor, Oracle RMUwin, Parallel Backup Monitor
- OCI
Oracle clients using the Oracle Call Interface (OCI) or Oracle server clients
- SQLSERVICES
Oracle SQL/Services clients (Query Performance Tuner, Oracle ODBC Driver for Rdb, Oracle SQL/Services API, Schema Manager)
- JDBC
JDBC for Oracle Rdb clients.

Note: Ensure that you have a dispatcher network port defined with a dispatcher message protocol that supports each service API you want to use. For example, if you define a service that supports the OCI API and another service that supports the SQLSERVICES API, you must define at least one dispatcher network port that supports the OCI dispatcher message protocol and the Oracle Net transport and another dispatcher network port that supports the SQLSERVICES dispatcher message protocol and any available transport, respectively.

Usage Notes

- To use the Oracle Net transport option, specify the Oracle Net transport option as <sqlnet-spec> in the <transport-spec> argument and specify the OCI Services for Oracle Rdb listener name as its <identifier> argument.
- The word LISTENER is a synonym for the keyword LISTENER_NAME.
- SQLSRV_MANAGE lets you create two or more dispatchers listening on the same port ID or object, but only the first dispatcher with a unique port ID or object is allowed to start. If you attempt to start a second dispatcher listening on the same port ID or object, it fails to start if it cannot listen on any of the specified network ports.
- Oracle recommends that you create a log file for troubleshooting purposes unless you have a problem with excessive I/O entries in the log file. The default directory for the LOG PATH and DUMP PATH arguments is SYS\$MANAGER.
- The LOG PATH and DUMP PATH arguments are available when you use the SQLSRV_MANAGE command utility, but not when you use the Oracle SQL/Services Manager GUI.
- If values are assigned to existing logicals SQLSRV_DISP_LOGPATH and SQLSRV_DISP_DUMPPATH, they override log path and dump path values specified by the LOG PATH and DUMP PATH arguments.

Examples

Example 1: Create a dispatcher that uses the TCP/IP protocol.

```
SQLSRV> CREATE DISPATCHER tcpip_disp NETWORK_PORT TCP/IP;  
SQLSRV> START DISPATCHER tcpip_disp;
```

Example 2: Create a dispatcher that uses the SQL*Net protocol.

```
SQLSRV> CREATE DISPATCHER sqlnet_disp  
_SQLSRV> NETWORK_PORT SQLNET LISTENER_NAME LISTENER;  
SQLSRV> START DISPATCHER sqlnet_disp;
```

Example 3: Specifying the NOLOG argument. If you specify NOLOG instead of a directory name for the LOG PATH argument, no log file is written, for example:

```
SQLSRV> CREATE DISPATCHER SQLSRV_DISP1 LOG PATH 'NOLOG'  
_SQLSRV> DUMP PATH 'SYS$MANAGER';
```

CREATE SERVER Command

Creates the server object definition and the configuration file. The definition is stored in the configuration file. The new server must be started offline.

Format

CREATE SERVER

```
->[ MAX_SHARED_MEMORY_SIZE <number> ]  
-->[ LOG_PATH <directory-specification> ]  
-->[ DUMP_PATH <directory-specification> ]  
->[ PROCESS_STARTUP_TIMEOUT <number-in-seconds> ]  
->[ PROCESS_SHUTDOWN_TIMEOUT <number-in-seconds> ]  
->[ <network-port-spec> ] ... ;
```

```
<network-port-spec> ::= NETWORK_PORT <transport-spec>  
<transport-spec> ::= { <tcp-spec> | <decnet-spec> }  
<tcp-spec> ::= TCPIP [ PORT_ID <number> ]  
<decnet-spec> ::= DECNET [ OBJECT { <number> | <identifier>  
| <quoted-string> } ]
```

Arguments

MAX_SHARED_MEMORY_SIZE <number>

Sets the size in kilobytes of the maximum shared memory that the server should use. The default is 2000 kilobytes (2 megabytes). Starting with Oracle SQL/Services release 7.0 and later, the server allocates the maximum shared memory size when the monitor starts up.

DUMP_PATH <directory-specification>

Specifies a directory name for bugcheck dump files. The default directory is SYSS\$MANAGER.

LOG_PATH <directory-specification>

Specifies a directory name for log files. The default directory is SYSS\$MANAGER.

If you specify NOLOG instead of a directory name for the LOG PATH argument, no log file is written.

PROCESS_STARTUP_TIMEOUT <number-in-seconds>

Specifies the length of time to wait before deciding that a dispatcher or executor process is not going to start up before the monitor takes action and terminates the process. This argument is an integer expressed in seconds. The default value is 0 seconds, which means that no process startup timer value is set. See the Usage Notes for more information.

PROCESS_SHUTDOWN_TIMEOUT <number-in-seconds>

Specifies the length of time to wait before deciding that a dispatcher or executor process is not going to shut down before the monitor takes action and terminates the process. This argument is an integer expressed in seconds. The default value is 0 seconds, which means that no process shutdown timer value is set; the process shutdown timer value is infinite. See the Usage Notes for more information.

<network-port-spec>

Lists network ports that the monitor should use for communications with Oracle SQL/Services management clients (SQLSRV_MANAGE and the Oracle SQL/Services Manager GUI). The network port specification is TCP/IP or DECnet. The default port ID for TCP/IP is 2199 and the default DECnet object name is SQLSRV_SERVER. If no network ports are specified, the monitor of the server uses the default ports. The maximum number of times that the <network-port-spec> argument can be specified in the CREATE SERVER command is five.

DECnet or TCP/IP must be available on the node for which the create server definition is defined. If none of these are available, the server will not start.

Usage Notes

- The CREATE SERVER command is typically used only during an Oracle SQL/Services installation. The installation procedure uses the SQLSRV_CREATE71.COM procedure to create a configuration file containing a server and a default set of dispatchers and services, and to start the server.

If you accidentally delete the configuration file or if the file becomes corrupted, you need to re-create the server if you do not have a backup. First, delete the original configuration file if it still exists. However, be sure to retain a copy of the file if it was corrupted by an Oracle SQL/Services component, so you can submit it with a software problem report. See [Section 7.1](#) for information on how to report a software problem. There are two ways to re-create the server.

- Run the SQLSRV_CREATE71.COM procedure.

Execute the SYSSMANAGER:SQLSRV_CREATE71.COM command procedure, which re-creates the server using the SYSSMANAGER:SQLSRV_CREATE71.SQS SQLSRV_MANAGE script.

Note: This is the recommended method of re-creating a server. Execute the SQLSRV_CREATE71.SQS file to re-create just the Oracle RMU dispatcher and Oracle RMU service objects.

- Issue the SET CONFIG_FILE command and specify a configuration file specification that does not exist. When you do this, you are prompted if you want to create one now; answer YES. The default is NO. If the SET CONFIRM command is set to OFF, then you are not prompted. A SHOW SETTINGS command displays the current settings and the file specification for this new configuration file. Issue a CREATE SERVER command to create a server using this configuration file.
- If the configuration file already exists and you issue a CREATE SERVER command, an error message displays and the CREATE SERVER command fails.
- The SQLSRV_MANAGE utility attempts to connect to the monitor of the server using the default TCP/IP or DECnet ports. If you change the network port of the server, you must also specify that port explicitly when connecting from the SQLSRV_MANAGE utility.
- If you want to set process startup and shutdown timers, follow these guidelines:
 - Usually dispatcher and executor processes start up and shut down in a reasonable period of time. Only during an unusual situation would you need to specify nonzero values for the PROCESS_STARTUP_TIMEOUT and PROCESS_SHUTDOWN_TIMEOUT arguments.
 - In heavily loaded systems, it often takes longer for a particular operation to complete. If either process startup or process shutdown is set to a value other than zero and fails for no apparent reason (you have checked other possible causes and have not isolated the problem), set a higher value for the PROCESS_STARTUP_TIMEOUT argument or the PROCESS_SHUTDOWN_TIMEOUT argument to see if that solves the problem.

Examples

Example 1: Create a server definition for a local node on which there is currently no Oracle SQL/Services server.

```
SQLSRV> SET CONFIG_FILE 'my_config_file';  
SQLSRV> CREATE SERVER MAX_SHARED_MEMORY_SIZE 3000;
```

```
SQLSRV> START SERVER;  
Server started  
Connecting to server ...  
Connected
```

CREATE SERVICE Command

Creates a service object definition for the current server only. The definition is stored in the configuration file. New service objects must be started online to be part of a running server.

Format

```

CREATE SERVICE <service-name>
->[ PROTOCOL { OCI | RMU | SQLSERVICES } ]
->[ AUTOSTART { ON | OFF } ]
->[ DEFAULT_CONNECT_USERNAME { <quoted-string> | <identifier> } ]
-->[ DEFAULT_CONNECT_PASSWORD <quoted-string> ]
->[ REUSE [ SCOPE ] [ IS ] { SESSION | TRANSACTION } ]
->[ SQL_VERSION { <version-number> | S[TANDARD] } ]
->[ PROCESS_INITIALIZATION { <quoted-string> | LOGIN } ]
->[ ATTACH <quoted-string> ]
-> OWNER { <quoted-string> | <identifier> }
->[OWNER PASSWORD <quoted-string> ]
->[ SCHEMA <quoted-string> ]
->[ SQL_INIT_FILE <quoted-string> ]
->[ DATABASE_AUTHORIZATION { [ SERVICE ] OWNER
| [ CONNECT ] USERNAME } ]
->[ APPLICATION_TRANSACTION_USAGE
{ SERIAL | CONCURRENT } ]
->[ IDLE_USER_TIMEOUT <number-in-seconds> ]
->[ IDLE_EXECUTOR_TIMEOUT <number-in-seconds> ]
->[ MIN_EXECUTORS <number> ]
->[ MAX_EXECUTORS <number> ]
->[ CLIENTS_PER_EXECUTOR <number> ] ;

```

<service-name> ::= <identifier>

Arguments

<service-name>

The service name. The service name is expressed as an identifier. The service name must be unique.

PROTOCOL {OCI | RMU | SQLSERVICES}

Determines the application programming interface (API) that each service can support. A service can support only one API. Specify an API that matches the type of client you want a service to support:

- OCI
Oracle or third-party clients using the Oracle Call Interface (OCI)
- RMU
Oracle RMU clients, for example, Rdb Performance Monitor, RMUwin, Parallel Backup Monitor
- SQLSERVICES
Oracle SQL/Services clients (Query Performance Tuner, Oracle ODBC Driver for Rdb, Oracle SQL/Services API, Schema Manager). This is the default.

Note: Ensure that you have a dispatcher network port defined with a dispatcher message protocol that supports each service API you want to use. For example, if you define a service that supports the OCI API and another service that supports the SQLSERVICES API, you must define at least one dispatcher network port that supports the OCI dispatcher message protocol and the SQL*Net transport and another dispatcher network port that supports the SQLSERVICES dispatcher message protocol and any available transport, respectively.

AUTOSTART {ON | OFF}

Determines whether or not the service object automatically starts up when you issue a START SERVER or RESTART SERVER command. If the argument is specified as ON, the service object automatically starts when you issue a START SERVER or RESTART SERVER command. The default is ON.

DEFAULT_CONNECT_USERNAME {<quoted-string> | <identifier>}

Specifies the user name as either a quoted string or an identifier under which unknown users are allowed to connect to the service. See [Section 2.7.1](#) and [Section 2.8](#) for more information about using this argument. The DEFAULT_CONNECT_USERNAME argument can be applied only to database services that support the SQLSERVICES protocol.

DEFAULT_CONNECT_PASSWORD <quoted-string>

Specifies the password associated with the connect user name as a quoted string.

REUSE SCOPE IS {SESSION | TRANSACTION}**■ SESSION**

An executor for a session reusable service processes requests for one client session at a time. A session begins when a client connects to the service and the connection is bound to an executor process. A session ends when a client disconnects from the service and the connection is unbound from the executor process. SESSION is the default.

■ TRANSACTION

An executor for a transaction reusable service processes requests for one transaction at a time; however, it supports many concurrent client sessions. A transaction begins when a client issues a SQL statement that either implicitly or explicitly starts a transaction. A transaction ends when a client issues a successful SQL COMMIT or ROLLBACK statement. The REUSE SCOPE IS TRANSACTION argument can be applied only to database services that support the SQLSERVICES protocol.

See [Section 2.6](#) for more information.

SQL_VERSION {<version-number> | STANDARD}

Specifies the version of SQL to use for the service. It is expressed as either a version number data type (for example, 7.1) for selecting a version of SQL in an Oracle Rdb multiversion environment or by the keyword STANDARD (or S) for running a standard version of SQL in an Oracle Rdb single version environment. Either value is used as the first parameter argument for the Oracle Rdb RDB\$SETVER command procedure when it runs, as described in the installation information. The version number resolves to an "n.n" parameter argument and the word STANDARD or S resolves to an S parameter argument. When no value is specified, the default is to use the keyword STANDARD.

PROCESS_INITIALIZATION {<quoted-string> | LOGIN}

The process initialization file can be either a special process initialization file specified as a <quoted-string> or the keyword LOGIN. When LOGIN is specified for the process initialization file, Oracle SQL/Services uses the file specified by the LGICMD qualifier in AUTHORIZE as returned by the OpenVMS SYS\$GETUAI system service. If this file specification is not fully qualified, the file will not be found and the executor will fail. If you

specify process initialization as LOGIN, make sure LGICMD qualifier is defined for the service owner account. The process initialization or login file is used to help define some of the attributes of the executor process for this service. If no argument is specified, the default is not to run any initialization file. This file is executed once for each executor, during executor startup.

ATTACH <quoted-string>

The SQL ATTACH statement.

If you do not specify a SQL ATTACH statement, you create a universal service that is not preattached to a specific database.

If you specify a SQL ATTACH statement, you create a database service that is preattached to the specified database.

This argument is a single-quoted string and is exactly the same format as the attach-string-literal used in dynamic SQL. The FILENAME keyword in this string cannot be abbreviated.

See the *Oracle Rdb SQL Reference Manual* for more information on the ATTACH statement.

OWNER {<quoted-string | <identifier>}

Specifies the user name of the owner of the service. Every service has an owner name. The owner name must be specified as a quoted-string or identifier; otherwise, an error message is returned.

If the service is a database service, then the service owner's privileges are used for access checks when an executor attaches to the specified database. See [Section 2.6](#) for more information on database services.

If database access authorization is by service owner, then the service owner's privileges are used for all database access operations. See the DATABASE_AUTHORIZATION argument for more information on database access authorization.

Executors are created with the privileges and quotas from the service owner's account. See [Section 2.10.1](#) for more information.

The OWNER argument can be applied only to database services that support the SQLSERVICES protocol.

OWNER PASSWORD <quoted-string>

Specifies the password for the owner of the service.

SCHEMA <quoted-string>

Provides a way to specify the default schema that you want to use when an executor attaches to a multischema database.

If a schema name is not specified in the service definition, the schema name defaults to the service owner account name if the database access authorization is service owner, or to the connect user name if the database access authorization is connect user name (see [Section 2.9](#)).

The schema argument allows the default to be overridden. If this argument is supplied to OCI Services for Oracle Rdb, it is ignored.

SQL_INIT_FILE <quoted-string>

Specifies a file containing SQL statements that tailor the SQL environment for a client connection. For example, you can set the SQL dialect and default character set by using a SQL initialization file. The statements in a SQL initialization file are executed every time a client connects to a service.

See [Section 6.1](#) for more information about using a SQL initialization file.

DATABASE_AUTHORIZATION {[SERVICE] OWNER | [CONNECT] USERNAME}

Determines the user name under which access to the database is made. The default is CONNECT USERNAME.

- SERVICE OWNER

For a database service, all access to the database is made by using the service owner user name. This option is not supported by OCI Services for Oracle Rdb.

- CONNECT USERNAME

Access to the database is made by using the client-specified user name, the DECnet proxy user name, or the user name specified in the DEFAULT_CONNECT_USERNAME argument.

For more information on database access authorization, see [Section 2.7](#) and [Section 2.8](#).

APPLICATION_TRANSACTION_USAGE {SERIAL | CONCURRENT}

The APPLICATION_TRANSACTION_USAGE argument is applicable only to transaction reusable database services. Some applications make only a single connection to a service to perform their work, while other applications make multiple connections to the same service. Connections created to transaction reusable database services are tied to the same executor for the life of the session. Refer to [Section 2.6.3](#), "Transaction Reusable Database Services", for more information.

If a client application makes multiple connections to a service and these are assigned to the same executor, a deadlock occurs if the client application attempts to start a new transaction

on one connection before ending an existing transaction on another connection. When you specify the **CONCURRENT** keyword, Oracle SQL/Services ensures that multiple connections from the same client application on the same node are never assigned to the same executor process.

When you specify the **SERIAL** keyword, Oracle SQL/Services assumes that client applications do not start concurrent transactions on multiple connections. Oracle SQL/Services assigns connections to executor processes on a least busy basis (the executor process with the fewest client connections already assigned). Thus, if a client application made more than one connection to the same service and the keyword **SERIAL** was specified, the second connection may or may not have gone to the same executor process as the first connection, depending on how many connections were already assigned to that executor process versus how many connections were assigned to the other executor processes for that service.

The default for the **APPLICATION_TRANSACTION_USAGE** argument is **SERIAL**.

Some applications, such as Microsoft Access, make multiple connections to the same service to perform their work and require that you specify the **CONCURRENT** keyword. If set to **CONCURRENT**, Oracle SQL/Services considers the node, user name, and application name of the client when choosing an executor to which to tie the connection and ensures that multiple connections from the same client application are never assigned to the same executor process.

This argument is used only by Oracle SQLSERVICES services.

IDLE_USER_TIMEOUT <number-in-seconds>

Specifies the amount of time in seconds that a client (user) can remain idle before the server disconnects the client. This value is expressed as an integer. The default value is 0, which displays as "<none>" in a **SHOW SERVICE** command and means that the idle timeout value is infinite. A specified value other than 0 is rounded to the next higher multiple of 90 seconds.

IDLE_EXECUTOR_TIMEOUT <number-in-seconds>

Specifies the amount of time in seconds that an executor process for a session reusable service can remain inactive (not bound to a client connection) before being deleted. The value is expressed as an integer. The default timeout value is 1800 seconds (30 minutes).

MIN_EXECUTORS <number>

Sets the minimum value to which the number of executor processes is allowed to decrease. This is also the number of executor processes started at startup using a **START SERVICE** or **START SERVER** command. The value is expressed as an integer. The default minimum number of executors for a session reusable service is 0. A service with **MIN_EXECUTORS**

set to 0 never shows the Starting state when the service starts up. The state displays as either Running or Failed.

If you use transaction reusable executors, you must set the value for the minimum number of executors so that it is equal to the value for the maximum number of executors. The default value is 1 for a transaction reusable service.

MAX_EXECUTORS <number>

Sets the maximum value to which the number of executor processes is allowed to increase. The value is expressed as an integer. The default maximum number of executors is 1.

If you use transaction reusable executors, you must set the value for the minimum number of executors so that it is equal to the value for the maximum number of executors. The default value is 1 for a transaction reusable service.

CLIENTS_PER_EXECUTOR <number>

Specifies the number of clients allowed per executor. The number of clients allowed is dependent upon whether the service is session reusable or transaction reusable. The default number of clients per executor for session reusability is 1 and cannot be greater than 1. The default number of clients per executor for transaction reusability is 1 but can be greater than 1. The CLIENTS_PER_EXECUTOR value is expressed as an integer.

Usage Notes

- When a service other than an OCI service is created, only a privileged user with SYSPRV privilege is authorized to use the service. Use the GRANT command to enable other users.
- If you use the default minimum number of 0 executors, the default maximum number of executors is 1. If the minimum number of executors defined is greater than 0, the default maximum number of executors *equals* the defined minimum value. For example, if the defined minimum number of executors is 5, the default maximum number of executors is also 5.

Examples

Example 1: Create a universal service named V71.

```
SQLSRV> CREATE SERVICE V71 OWNER 'SQLSRV$DEFLT' SQL VERSION 7.1
_SQLSRV> MIN_EXECUTORS 5
_SQLSRV> MAX_EXECUTORS 10;
SQLSRV> START SERVICE V71;
```

DISCONNECT SERVER Command

Disconnects a connection to a server.

Format

```
DISCONNECT SERVER      [ <connect-name> ];  
  
                        <connect-name> ::= <identifier>
```

Arguments

<connect-name>

The connection name. This identifier uniquely identifies the connection to a server on a particular node. The connection name is expressed as an identifier.

Usage Notes

The DISCONNECT SERVER command works in the opposite way as the CONNECT TO SERVER command. It disconnects the named connection if a connection name is specified or disconnects the current connection if no connection name is specified.

Examples

Example 1: Disconnect from the server whose connection name is eagle.

```
SQLSRV> CONNECT TO SERVER AS eagle;  
Connecting to server ...  
Connected  
SQLSRV> DISCONNECT SERVER eagle;
```

DROP Command

Deletes the specified object for the current server.

Format

```
DROP                <obj-type> <obj-name>;

                        <obj-type> ::= DISPATCHER | SERVICE
                        <obj-name> ::= <identifier>
```

Arguments

<obj-type>

Specifies dispatcher or service using the keyword DISPATCHER or SERVICE object type, respectively.

<obj-name>

The name of the object to be deleted. The object name is expressed as an identifier.

Usage Notes

- For online deletions, the object to be deleted cannot be currently active or running; that is, the object must first be shut down online. You may want to issue a SHOW CLIENTS command to determine if there are any client applications using the service you are going to shut down and delete and to ensure that no clients are connected to that service.

The SQLSRV_MANAGE utility does not prevent you from deleting a dispatcher or service object online while the dispatcher or service is running on a different node in an environment where two or more nodes share the same configuration file. If this happens, the SQLSRV_MANAGE utility displays a warning message if you show the dispatcher or service that has been deleted but is still running, for example:

```
SQLSRV> SHOW service <obj_name>;
*****

** This Service has been deleted from the config file.  **
** It will not exist after it is shut down.             **
*****
```

- Oracle recommends that you do not make offline modifications to a configuration file if there is a server running that is using the same file. In this situation, the SQLSRV_MANAGE utility, for example, does not prevent you from deleting a dispatcher or service object offline while the dispatcher or service is running.

A client application using a service or dispatcher that has been deleted offline continues to have use of that object until it disconnects from the server object. However, once the client application disconnects from the server, it cannot reconnect to the dispatcher or service that was deleted. Before the object that was deleted is shut down, a SHOW command displays a message for the deleted object as shown in the previous list item.

- The DROP command removes the specified object from the configuration file.

Examples

Example 1: Delete the database_3 service object.

```
SQLSRV> SHUTDOWN SERVICE database_3;
```

```
SQLSRV> DROP SERVICE database_3;
```

Example 2: Delete the disp_tcpip dispatcher object.

```
SQLSRV> SHUTDOWN DISPATCHER disp_tcpip;
```

```
SQLSRV> DROP DISPATCHER disp_tcpip;
```

DROP SERVER Command

Deletes the current server, including the configuration file.

Format

DROP SERVER;

Usage Notes

- The server to be deleted cannot currently be active; it must first be shut down online and then deleted offline.
- The DROP SERVER command is an offline operation; you cannot be connected to the server.
- The DROP SERVER command deletes the configuration file.

Examples

Example 1: Delete the current server object.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> SHUTDOWN SERVER;  
SQLSRV> DISCONNECT SERVER;  
SQLSRV> SET CONFIG_FILE 'my_config_file';  
SQLSRV> DROP SERVER;  
Configuration file will be deleted, continue? (N) y  
Configuration file deleted
```

EXIT Command

Exits the SQLSRV_MANAGE environment.

Format

`EXIT[;]`

Usage Notes

- You can exit the SQLSRV_MANAGE environment or utility in the following two ways:
 - Using the EXIT command
 - When an end-of-file is encountered on the last input source

If you are using the SQLSRV_MANAGE utility interactively, you can enter Ctrl/Z to exit the SQLSRV_MANAGE utility.

If you specify an input file on the command line with the `-input` file switch, and the file is the last input source, and an end-of-file is reached, SQLSRV_MANAGE exits.

- Use of the terminating semicolon (;) is optional.

Examples

Example 1: Exit the SQLSRV_MANAGE environment.

```
SQLSRV> EXIT
```

EXTRACT Command

Extracts server object definitions from the active configuration file and writes them to a SQL/Services command script if an output file has been defined, or to the current output device. The script can be used to re-create servers, dispatchers, and services.

Format

EXTRACT keyword [option];

Argument

keyword [option]

Keyword can be one of the following:

- **SERVER**
Extracts the definition for the server. There is no option for this keyword.
- **DISPATCHER [disp_name]**
If the disp_name is omitted or represented by an asterisk (*), definitions for all dispatchers are extracted. If the disp_name is specified, the definition for just that dispatcher is extracted.
- **SERVICE [service_name]**
If the service_name is omitted or represented by an asterisk (*), definitions for all services are extracted. If the service_name is specified, the definition for just that service is extracted.

Usage Notes

To extract definitions to a file, use the -output switch on the SQLSRV_MANAGE command, or issue an OPEN command before issuing the EXTRACT command.

Examples

Example 1: Extract the definition for the service OCI_AAA and display the information on the output device that is currently defined for the session.

```
$ SQLSRV_MANAGE71
```



```

SQLSRV> EXTRACT SERVICE OCI_AAA;
Create Service OCI_AAA
  Owner                'AAA'
  Protocol              OCI
  SQL version          7.1
  Autostart            off
  Process_initialization 'DBD_USER6:[JONES]INIT_OCI_ENG70.COM'
  ATTACH 'filename DBD_USER6:[JONES]mf_personnel'
  Reuse scope is       SESSION
  Database Authorization CONNECT USERNAME
  SQL_init_file        'DBD_USER6:[JONES]init.sql'
  Application Transaction Usage SERIAL
  Idle Executor Timeout 1800
  Min Executors        1
  Max Executors        10
  Clients Per Executor 1
;
Grant use on service OCI_AAA
  To 'AAA'
;
SQLSRV>

```

Example 2: Extract definitions for the entire dispatcher definition and write them to an output file.

```
$ SQLSRV_MANAGE71 -OUTPUT A.SQL
```

```
SQLSRV> EXTRACT DISP;
```

```
SQLSRV> EXIT
```

```
$TYPE A.SQL
```

```

Create Dispatcher SQLSRV_DISP
  Autostart            on
  Max connects         101
  Idle User Timeout   0
  network_port spxipx port_id 0x84b1      protocol SQLServices
  network_port DECnet object 81           protocol SQLServices
  network_port tcpip  port_id 118         protocol SQLServices
  Log path             'SYS$MANAGER:'
  Dump path            'SYS$MANAGER:'
;
Create Dispatcher RMU_DISP
  Autostart            on
  Max connects         100
  Idle User Timeout   0
  network_port tcpip  port_id 1571        protocol Native

```

```
Log path          'SYS$MANAGER:'
Dump path         'SYS$MANAGER:'
;
Create Dispatcher OCI_DISP
Autostart         on
Max connects      35
Idle User Timeout 0
network_port sqlnet listener oci_listener protocol OCI
Log path          'SYS$MANAGER:'
Dump path         'SYS$MANAGER:'
;
Create Dispatcher SQLSRV_MANAGE
Autostart         off
Max connects      100
Idle User Timeout 0
network_port spxipx port_id 0x84b1      protocol SQLServices
network_port DECnet object 81          protocol SQLServices
network_port tcpip port_id 118         protocol SQLServices
Log path          'SYS$MANAGER:'
Dump path         'SYS$MANAGER:'
;
```

Example 3: Extract definitions to an output file using the OPEN command.

```
$ SQLSRV_MANAGE71
SQLSRV> OPEN aaa.sql;
SQLSRV> EXTRACT SERVICE OCI_AAA;
SQLSRV> CLOSE;
```

GRANT USE ON SERVICE Command

Grants the USE privilege descriptor for a service to a user. Use this command to grant USE to an identifier and permit access to the specified service to a user who holds that specific identifier.

Format

```
GRANT USE ON SERVICE <service-name-list> TO <grant-element-list> ;

<service-name-list> ::= <service-name> [ , <service-name> ] ...
<service-name> ::= <identifier>
<grant-element-list> ::= <grant-element> [ , <grant-element> ] ...
<grant-element> ::= { PUBLIC | PRIVILEGED_USER
    | [ USER[S] ] <user-name> [ , <user-name> ] ...
    | IDENTIFIER[S] <identifier-name> [ , <identifier-name> ] ...
    | GROUP[S] <group-name> [ , <group-name> ] ... }
<user-name> ::= { <quoted-string> | <identifier> }
<identifier-name> ::= { <quoted-string> | <identifier> }
<group-name> ::= { <quoted-string> | <identifier> }
```

Arguments

<service-name-list>

Lists service names on which the GRANT USE ON SERVICE command operates. The service name is expressed as an identifier.

<grant-element-list>

Lists grant elements on which the GRANT USE ON SERVICE command acts. A grant list element can be the keyword PUBLIC or PRIVILEGED_USER, a list of user names, a list of identifier names, or a list of group names. A PRIVILEGED_USER is defined as a user with SYSPRV privilege (either default or granted privilege). A user name, identifier name, or group name is expressed as either a quoted string or an identifier.

Usage Notes

- For release 7.0 and later, Oracle SQL/Services grants a single privilege descriptor, USE.
- Granting a new user the USE privilege descriptor takes effect upon the user's next attempt to use Oracle SQL/Services *after* the privilege change is complete. For example, a new user, once granted the USE privilege descriptor, can use Oracle SQL/Services on the next attempt.
- If you use the keyword IDENTIFIER[S] or GROUP[S], the specified identifier is added to the list of granted identifiers and permits a user who holds that specific identifier to access the specified service. If the IDENTIFIER[S] or GROUP[S] keyword is omitted, then the specified user name is granted access to use the service.

Examples

Example 1: Grant the USE privilege descriptor for the general service to PUBLIC.

```
SQLSRV> GRANT USE ON SERVICE general TO PUBLIC;
```

Example 2: Grant the USE privilege descriptor for the database_2 service to fred and wilma.

```
SQLSRV> GRANT USE ON SERVICE database_2 TO fred,wilma;
```

Example 3: Grant the USE privilege descriptor for the system management SQLSRV_MANAGE service to fred and wilma.

```
SQLSRV> GRANT USE ON SERVICE sqlsrv_manage TO fred,wilma;
```

Example 4: Grant the USE privilege descriptor for the system management SQLSRV_MANAGE service to the identifiers payroll_dba and operator.

```
SQLSRV> GRANT USE ON SERVICE sqlsrv_manage  
_SQLSRV> TO IDENTIFIERS payroll_dba,operator;
```

HELP Command

Gets help on a topic within the SQLSRV_MANAGE environment.

Format

```
HELP           [ <help-keyword> ] ... [ ; ]  
  
                <help-keyword> ::= <identifier>
```

Arguments

<help-keyword>

A help keyword. The help keyword is expressed as an identifier.

Usage Notes

Use of the terminating semicolon (;) is optional.

Examples

Example 1: Get help on a topic within the SQLSRV_MANAGE environment.

```
SQLSRV> HELP
```

KILL EXECUTOR Command

Kills the specified executor.

Format

```
KILL EXECUTOR      { PID <process-id> | <executor-name> };  
  
                    <process-id> ::= <number>  
                    <executor-name> ::= <identifier>
```

Arguments

{PID <process-id> | <executor-name>}

The process ID or executor name. The process ID is expressed as an integer and can be represented either in decimal or hexadecimal format. The executor name is expressed as an identifier. To determine the executor name, perform a SHOW CLIENTS FULL command.

Usage Notes

- The process ID can be represented in either decimal or hexadecimal format. To represent a process ID in hexadecimal format, precede the process ID value with the value '0x' or '0X' (for example, 0x0000088a).
- You can kill an executor only as an online operation; that is, you must be connected to a running server (CONNECT TO SERVER command) to kill an executor running on that server.

Examples

Example 1: Kill an executor by process ID (represented in hexadecimal format).

```
SQLSRV> KILL EXECUTOR PID 0x0000072a;
```

Example 2: Kill an executor by process ID (represented in decimal format).

```
SQLSRV> KILL EXECUTOR PID 324693;
```

Example 3: Kill an executor by name.

```
SQLSRV> KILL EXECUTOR generi004000280;
```

OPEN Command

Opens an output file in the SQLSRV_MANAGE environment. Subsequent output by SQLSRV_MANAGE, including error messages, is written to this file.

Format

```
OPEN                <file-spec>;

                       <file-spec> ::= { <identifier> | <quoted-string> }
```

Arguments

<file-spec>

The output file name. The file name is expressed either as an identifier or as a quoted string.

Usage Notes

The OPEN command creates the specified file and writes all subsequent output to that file. If you enter the OPEN command, the OPEN command does an implicit close of the current output file if an output file was already open.

Examples

Example 1: Open an output file.

```
SQLSRV> OPEN test_file;
```

Example 2: Open an output file and extract dispatcher definitions to that file.

```
SQLSRV> OPEN ocidisp.lis;
SQLSRV> EXTRACT disp oci_disp;
SQLSRV> CLOSE;
SQLSRV> TY ocidisp.lis;
Create Dispatcher OCI_DISP
  Autostart                on
  Max connects              35
  Idle User Timeout        0
  network_port sqlnet listener oci_listener protocol OCI
  Log path                  'SYS$MANAGER:'
  Dump path                 'SYS$MANAGER:'
```

RESTART SERVER Command

Restarts the current server.

Format

```
RESTART SERVER [ AUTOSTART { ON | OFF } ] ;
```

Arguments

AUTOSTART {ON | OFF}

Determines whether or not other server objects (dispatchers and services) automatically start up again when you issue a RESTART SERVER command. ON is the default. If the argument is specified as ON, other server objects automatically restart (shut down and start again) if each object's AUTOSTART argument value is also set as ON. If you do not want to restart other server objects, specify the AUTOSTART attribute value as OFF in the RESTART SERVER command. The AUTOSTART OFF attribute setting overrides each object's AUTOSTART attribute setting and allows you to individually start each object after restarting just the server object.

Usage Notes

- You can restart a server only as an online operation; that is, you must be connected to the server (CONNECT TO SERVER command) to restart it.
- Use the RESTART SERVER command to restart the server. By default, all server components (dispatchers and services) for the current server will also restart unless these server objects have the AUTOSTART argument specified as OFF in their definitions.

Examples

Example 1: Restart the current server.

```
SQLSRV> CONNECT TO SERVER;  
Connecting to server ...  
Connected  
SQLSRV> ALTER SERVER MAX_SHARED_MEMORY_SIZE 4000;  
%DBS-S-ALTER_RESTART, Restart object to have altered settings take effect  
SQLSRV> RESTART SERVER;  
Disconnected from Server
```

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected
```

REVOKE USE ON SERVICE Command

Revoke the USE privilege descriptor for a service from a user. You can revoke USE from an identifier to remove access to the specified service by users who hold that identifier.

Format

```

REVOKE USE ON SERVICE <service-name-list> FROM <grant-element-list> ;

<service-name-list> ::= <service-name> [ , <service-name> ] ...
<service-name> ::= <identifier>
<grant-element-list> ::= <grant-element> [ , <grant-element> ] ...
<grant-element> ::= { PUBLIC | PRIVILEGED_USER
    | [ USER[S] ] <user-name> [ , <user-name> ] ...
    | IDENTIFIER[S] <identifier-name> [ , <identifier-name> ] ...
    | GROUP[S] <group-name> [ , <group-name> ] ... }
<user-name> ::= { <quoted-string> | <identifier> }
<identifier-name> ::= { <quoted-string> | <identifier> }
<group-name> ::= { <quoted-string> | <identifier> }

```

Arguments

<service-name-list>

Lists service names on which the REVOKE USE ON SERVICE command operates. The service name is expressed as an identifier.

<grant-element-list>

Lists grant elements on which the REVOKE USE ON SERVICE command acts. A grant list element can be the keyword PUBLIC or PRIVILEGED_USER, a list of user names, a list of identifier names, or a list of group names. A PRIVILEGED_USER is defined as a user with SYSPRV privilege (either default or granted privilege). A user name, identifier name, or group name is expressed as either a quoted string or an identifier.

Usage Notes

- For release 7.0 and later, Oracle SQL/Services revokes a single privilege descriptor, USE.
- Revoking the USE privilege descriptor from an existing user takes effect upon the user's next attempt to use Oracle SQL/Services *after* the privilege change is complete. For example, a user whose USE privilege descriptor is revoked but who is still using Oracle SQL/Services, will not be able to use Oracle SQL/Services after disconnecting and then attempting to reconnect to the service.
- If you use the keyword IDENTIFIER[S] or GROUP[S], any specified identifier is removed from the service's list of granted identifiers. If you omit the IDENTIFIER[S] or GROUP[S] keyword, the specified user name is removed from the service's list of granted user names.

If you revoke use of a service by a specific user name, that user is still able to access the service if the user holds an identifier that has been granted use of the service. Likewise, if you revoke use of a service by a specific identifier, a user who holds that identifier is still able to access the service if the user's name has been granted use of the service.

Examples

Example 1: Remove the USE privilege descriptor for the general service from PUBLIC.

```
SQLSRV> REVOKE USE ON SERVICE general FROM PUBLIC;
```

Example 2: Remove the USE privilege descriptor for the database_3 service from fred and wilma.

```
SQLSRV> REVOKE USE ON SERVICE database_3 FROM fred,wilma;
```

Example 3: Remove the USE privilege descriptor for the system management SQLSRV_MANAGE service from fred and wilma.

```
SQLSRV> REVOKE USE ON SERVICE sqlsrv_manage FROM fred,wilma;
```

Example 4: Remove the USE privilege descriptor for the system management SQLSRV_MANAGE service from the identifier names payroll_dba and operator.

```
SQLSRV> REVOKE USE ON SERVICE sqlsrv_manage  
_SQLSRV> FROM IDENTIFIERS payroll_dba,operator;
```

SET CONFIGURATION_FILE Command

Enables you to select a server configuration file with which to start a server or to make server changes offline. Any subsequent management commands are written to the configuration file only and do not affect the running server except for GRANT USE and REVOKE USE commands and any restarted dispatchers and services.

Format

```
SET CONFIG[URATION] FILE <file-name>;

<file-name> ::= <quoted-string>
```

Arguments

<file-name>

The configuration file name. The file name is expressed either as an identifier or as a quoted string.

Usage Notes

- CONFIG_FILE is a synonym for the keyword CONFIGURATION_FILE.
- When the SQLSRV_MANAGE utility starts up, it establishes a default configuration file name, as follows:

```
SYS$MANAGER:SQLSRV_CONFIG_FILE71.DAT
```

To override the default, set the SQLSRV_CONFIG_FILE71 logical name or supply a different file name to the SET CONFIGURATION_FILE command.

- The SHOW SETTINGS command shows the configuration file that offline modifications act upon. The SHOW SERVER command shows the configuration file that online modifications act upon.
- If you issue the SET_CONFIG_FILE command and specify a configuration file specification that does not exist, you are prompted whether or not you want to create one now. The default is NO. If the SET CONFIRM command is set to OFF, then you are not prompted. A SHOW SETTINGS command displays the current settings and file

specification for this new configuration file. If you issue a CREATE SERVER command, a server using this configuration file is created.

- When you make modifications to a configuration file using the SET CONFIG_FILE command, all changes are made offline and do not affect the running server, except GRANT and REVOKE command changes. Changes made to a server's configuration file can be applied to the running server by restarting the object changed.

Examples

Example 1: Set the configuration file.

```
SQLSRV> SET CONFIG_FILE 'my_config_file';
```

SET CONFIRM Command

Displays a confirmation prompt on the default output device when it is set as ON in the SQLSRV_MANAGE environment.

Format

```
SET CONFIRM      { ON | OFF } ;
```

Arguments

{ON | OFF}

When confirm is set as ON, a confirmation prompt is displayed on the default output device requiring confirmation for certain management operations. When confirm is set as OFF, a confirmation prompt is no longer displayed and no longer requires confirmation for certain management operations. ON is the default.

Usage Notes

- If the SET CONFIRM command is set as ON (the default) and you issue a SQLSRV_MANAGE command that in turn presents a confirmation prompt, this prompt is displayed on the default output device. For example, if you shut down and delete a server and then issue a SET_CONFIG_FILE command, and specify a configuration file that does not exist, you are prompted whether or not you want to create one now. The default is NO or not to create one now. If the SET CONFIRM command is set as OFF, you are not prompted to confirm this operation.
- A SHOW SETTINGS command displays, among other things, the current setting for the SET CONFIRM command.

Examples

Example 1: No longer display a confirmation prompt on the default output device.

```
SQLSRV> SET CONFIRM OFF;
```

SET CONNECTION Command

Enables you to establish the specified connection as the current connection so that you can manage that server.

Format

```
SET CONNECT[ION] [ <connect-name> ] ;  
  
      <connect-name> ::= <identifier>
```

Arguments

<connect-name>

The name of the connection. The identifier that uniquely identifies the connection to a server on a particular node. The connection name is expressed as an identifier.

Usage Notes

The SET CONNECT command allows you to manage multiple servers from a single SQLSRV_MANAGE session by switching between connections to the servers you are managing.

To manage a server online, you must first connect to the server using the CONNECT TO SERVER command. When you establish a new connection to a server using the CONNECT TO SERVER command, the new connection becomes the current connection. All online system management commands operate on the current connection. You can establish connections to multiple servers by issuing multiple CONNECT TO SERVER commands. You then use the SET CONNECT command to select the server that you wish to manage. Use the DISCONNECT SERVER command to disconnect from a server, at which time one of the remaining connections, if any, becomes the current connection.

Examples

Example 1: Manage two servers on nodes EAGLE and FALCON from node EAGLE.

```
SQLSRV> CONNECT SERVER AS EAGLE;  
Connecting to server ...  
Connected  
SQLSRV> CONNECT SERVER AS FALCON NODE FALCON
```



```
_SQLSRV> USER 'dbsmgr' USING 'password';
```

```
Connecting to server ...
```

```
Connected
```

```
SQLSRV> SHOW CONNECT;
```

```
Active connections:
```

```
CURRENT: FALCON
```

```
Service: SQLSRV_MANAGE
```

```
User: dbsmgr Node: FALCON Local: No
```

```
Transport: DECNET Object: SQLSRV_SERVER
```

```
Request bufsize: 1024 Response bufsize: 1024
```

```
EAGLE
```

```
Service: SQLSRV_MANAGE
```

```
User: <unknown> Node: EAGLE Local: Yes
```

```
Transport: DECNET Object: SQLSRV_SERVER
```

```
Request bufsize: 1024 Response bufsize: 1024
```

```
SQLSRV> SHOW SERVICES;
```

Name	State	C l i e n t s			E x e c u t o r s		
		Per-Exec	Max	Active	Min	Max	Running
RMU_SERVICE	RUNNING	1	100	0	4	100	4
GENERIC	RUNNING	1	10	0	2	10	2
SQLSRV_MANAGE	RUNNING	100	0	1	0	0	0

```
SQLSRV> SET CONNECT EAGLE;
```

```
SQLSRV> SHOW CONNECT;
```

```
Active connections:
```

```
FALCON
```

```
Service: SQLSRV_MANAGE
```

```
User: dbsmgr Node: FALCON Local: No
```

```
Transport: DECNET Object: SQLSRV_SERVER
```

```
Request bufsize: 1024 Response bufsize: 1024
```

```
CURRENT: EAGLE
```

```
Service: SQLSRV_MANAGE
```

```
User: <unknown> Node: EAGLE Local: Yes
```

```
Transport: DECNET Object: SQLSRV_SERVER
```

```
Request bufsize: 1024 Response bufsize: 1024
```

```
SQLSRV> SHOW SERVICES;
```

Name	State	C l i e n t s			E x e c u t o r s		
		Per-Exec	Max	Active	Min	Max	Running
V71	RUNNING	1	20	2	5	20	5
RMU_SERVICE	RUNNING	1	100	3	4	100	4
GENERIC	RUNNING	1	50	5	20	50	20
SQLSRV_MANAGE	RUNNING	100	0	1	0	0	0

SET OUTPUT Command

Directs output to the default device if set as ON in the SQLSRV_MANAGE environment.

Format

```
SET OUTPUT      { ON | OFF } ;
```

Arguments

{ON | OFF}

When output is set as ON it is directed to the default output device. When output is set as OFF it is suppressed.

Usage Notes

None.

Examples

Example 1: Set the output to the default device.

```
SQLSRV> SET OUTPUT ON;
```

SET VERIFY Command

Displays command file input on the default output device as it is read in the SQLSRV_ MANAGE environment.

Format

```
SET VERIFY          { ON | OFF } ;
```

Arguments

{ON | OFF}

When verify is set as ON, command file input is displayed on the default output device.

When verify is set as OFF, command file input is no longer displayed on the default output device.

Usage Notes

None.

Examples

Example 1: Display command file input on the default output device as it is read.

```
SQLSRV> SET VERIFY ON;
```

SHOW CLIENTS Command

Shows the active users for services for a configuration.

Format

```
SHOW CLIENTS    [[ FOR ] <name-spec> [ FULL ] ] ;
```

```
<name-spec> ::= { * | [ SERVICE ] <service-name-list>
                | [ USERNAME ] <username-list> | [ PID ] <executor-pid> }
<service-name-list> ::= <identifier> [ , <identifier> ] ...
<user-name-list> ::= <identifier> [ , <identifier> ] ...
<executor-pid> ::= <number>
```

Arguments

<name-spec>

The name specification. You can show:

- The clients connected to one or more services
- All clients for a specific executor PID
- All clients connected to a server using a specific user name or list of user names
- Clients connected to a particular executor

The default is "*", which displays all clients for all services.

FULL

Displays a full description of information for each client. The default is to display brief information (one line of output) for each client. When no service name is specified, `SQLSRV_MANAGE` displays clients grouped by service name.

Usage Notes

- To show all clients for all services, you can either use the `SHOW CLIENTS` command and not specify the `[FOR] [SERVICE]` keywords, or specify an asterisk (*). Either method displays all clients for all services. For example:

```
SQLSRV> SHOW CLIENTS;  
SQLSRV> SHOW CLIENTS *;
```

- To show all clients for a specific executor PID, specify the SHOW CLIENTS FOR PID command and specify the executor PID.
- The executor PID can be represented in either decimal or hexadecimal format. To represent an executor PID in hexadecimal format, precede the executor PID value with the value '0x' or '0X' (for example, 0x0000088a).
- Client connections serviced by a session reusable service can be in one of three possible states (see [Section 3.2](#) for more information):
 - Running Binding – The client is running and in the process of binding to an executor.
 - Running Bound – The client is running and is bound to an executor.
 - Canceling – The client connection is in the process of being disconnected.
- Client connections serviced by a transaction reusable database service can be in one of five possible states (see [Section 3.2](#) for more information):
 - Running Binding – The client is running and in the process of binding to an executor.
 - Running Bound – The client is running and is bound to an executor.
 - Running Unbound – The client is not submitting requests, therefore is not bound to an executor, but it is still connected to its executor.
 - Canceling Binding – The client is in the process of informing the executor that the bound connect is going away (this operation precedes the Canceling operation).
 - Canceling – The client connection is in the process of being disconnected.
- This command also shows the management clients that are using the management service. The SHOW CLIENTS command allows server system managers to determine if other server system managers are connected to the server and using the management service.
- This command shows the actual location of executor log and error files and the location of an executor dump file should one be created.

Examples

Example 1: Show the clients for the universal service named generic and display a brief description.

SHOW CLIENTS Command

```
SQLSRV> SHOW CLIENTS FOR SERVICE generic;
```

```
Service: GENERIC
```

Connect	Client		Executor	
Username	Node	State	PID	Application
User1	123.0.0.1	RUNNING BOUND	28c0c4e6	Personnel
User2	121.0.0.1	RUNNING BOUND	30b0a4d5	Personnel

Example 2: Show the clients for all user names for all services and display a brief description.

```
SQLSRV> SHOW CLIENTS;
```

```
Service: SS_SERV
```

Connect	Client		Executor	
Username	Node	State	PID	Application
	123.0.0.1	RUNNING BINDING	00000000	
	123.0.0.1	RUNNING BINDING	00000000	

```
Service: MMS
```

Connect	Client		Executor	
Username	Node	State	PID	Application
Rdbuser1	NODE1	RUNNING BOUND	00001045	Personnel

```
Service: SQLSRV_MANAGE
```

Connect	Client		Executor	
Username	Node	State	PID	Application
User1	NODE2	RUNNING BOUND	00000af2	SQLSRV_MANAGE

Example 3: Show the clients for the universal service named generic and display a full description.

```
SQLSRV> SHOW CLIENTS FOR SERVICE generic FULL;
```

```
Client Connect Username sqsapim1
```

```
Service: GENERIC
```

```
Application: Personnel
```

```
State: RUNNING BOUND
```

```
Node: 12.34.567.89
```

```
Executor: GENERI0050002
```

```
Executor PID: 543173877 0X20602cf5
```

```
Log File: SYS$$SYSROOT:[SYSMGR]SQS_EAGLES_GENERI0050002.LOG
```

```
Dump File: SYS$$SYSROOT:[SYSMGR]SQS_EAGLES_GENERI0050002.DMP
```

```
Client Connect Username sqsapim2
```

Service: GENERIC
Application: Personnel
State: RUNNING BOUND
Node: LOCAL:.mypc
Executor: GENERI0080004
Executor PID: 543173877 0X20602cf5
Log File: SYS\$SYSROOT:[SYSMGR]SQS_EAGLES_GENERI0080004.LOG
Dump File: SYS\$SYSROOT:[SYSMGR]SQS_EAGLES_GENERI0080004.DMP

SHOW CONNECTIONS Command

Shows information about the current server.

Format

`SHOW CONNECT[ION]S;`

Usage Notes

- CONNECTS is a synonym for the keyword CONNECTIONS.
- The SHOW CONNECT[ION]S command shows you information about all of the active management connections that the SQLSRV_MANAGE utility has to each server. Use the SHOW CONNECT[ION] command first to determine the current connection before issuing additional server management commands.

Examples

Example 1: Show information about the current server and connections to other servers.

```
SQLSRV> SHOW CONNECT;  
Active connections:  
CURRENT: SQLSRV_MANAGE  
        Service: SQLSRV_MANAGE  
        User: system Node: hawk Local: No  
        Transport: TCP/IP Port-id: 2199  
        Request bufsize: 1024 Response bufsize: 1024  
  
        SQLSRV_MANAGE  
        Service: SQLSRV_MANAGE  
        User: system Node: falcon Local: Yes  
        Transport: TCP/IP Port-id: 2199  
        Request bufsize: 1024 Response bufsize: 1024
```


SHOW DISPATCHER Command

Shows the static definition of dispatcher objects and their operational state for the current server.

Format

```
SHOW DISPATCHER [ <dispatcher-spec> ] ;
```

```
<dispatcher-spec> ::= { * | <dispatcher-name-list> }
```

```
dispatcher-name-list ::= <identifier> [, <identifier> ] ...
```

Arguments

<dispatcher-spec>

The dispatcher object specification. This can be one or more dispatcher object names or can be specified with an asterisk (*). If an * is specified, information for all dispatcher object names is displayed.

Usage Notes

- If no dispatcher object is specified, then information for *all* dispatcher objects is displayed.
- The dispatcher state and network port states can be one of three possible states:
 - Running – The dispatcher or dispatcher network port is running.
 - Inactive – The dispatcher or dispatcher network port is shut down.
 - Unknown – The management client is not connected to the server online so it cannot determine the state of the dispatcher and its network ports. The management client used the SET CONFIG_FILE command to manage the server offline. Use the CONNECT TO SERVER command to connect to the server online to determine the dispatcher state and the state of its network ports.
- When a difference exists for an attribute between the running server and its configuration file, the SHOW DISPATCHER command displays this difference at the end of the show output and indicates that when the server is restarted, the running

server's dispatcher is updated to match the server's dispatcher definition in the configuration file.

- This command shows the actual location of dispatcher log and error files and the location of a dispatcher dump file if one was created.

Examples

Example 1: Show information for the sqlsrv_disp dispatcher.

```
SQLSRV> CONNECT TO SERVER NODE hawk USER system USING password;
Connecting to server ...
Connected
SQLSRV> SHOW DISPATCHER sqlsrv_disp;
Dispatcher SQLSRV_DISP
State:                                RUNNING
Autostart:                             on
Max connects:                           100 clients
Idle user Timeout:                       <none>
Max client buffer size:                   5000 bytes
Network Ports:                           (State) (Protocol)
  SPX/IPX port      0x84b1                Inactive  SQL/Services
  DECnet object    81                     Running   SQL/Services
  TCP/IP port      118                     Running   SQL/Services
  SQL*Net listener FUBAR                 Running   SQL/Services
Log Path:                                USER1:[SQLSRV_TEST1]
Dump path:                                USER1:[SQLSRV_TEST1]
Log File:                                USER1:[SQLSRV_TEST1]:SQS_EAGLE_SQLSRV_DIS100380.LOG
Dump File:                                USER1:[SQLSRV_TEST1]:SQS_EAGLE_SQLSRV_DIS1003.DMP
```

Example 2: Show changed values for log path and dump path that will be in place after the dispatcher is restarted.

```
SQLSRV> SHOW DISP SQLSRV_DISP;
Dispatcher SQLSRV_DISP
.
.
.
Log Path:                                USER1:[SQLSRV_TEST1]
Dump path:                                USER1:[SQLSRV_TEST1]
.
.
.
```

** This Dispatcher will be updated as follows when it is restarted **

Log path: USER1:[SQLSRV_TEST1.AAA]
Dump path: USER1:[SQLSRV_TEST2.BBB]

SHOW SERVER Command

Shows the static definition of the server object and its operational state.

Format

SHOW SERVER;

Usage Notes

- The server network port state can be one of three possible states:
 - Running – The server network port is running.
 - Inactive – The server network port is shut down.
 - Unknown – The management client is not connected to the server online so it cannot determine the state of the server network ports. The management client used the SET CONFIG_FILE command to manage the server offline. Use the CONNECT TO SERVER command to connect to the server online to determine the state of its network ports.
- When a difference exists for an attribute between the running server and its configuration file, the SHOW SERVER command displays these differences at the end of the show output and indicates that when the server is restarted, the running server is updated to match the server's definition in the configuration file.
- Shows the actual location of server log and error files and the location of the server dump file should one be created.

Examples

Example 1: Show information for the server defined in the configuration file.

```
SQLSRV> SHOW SERVER;
Server Version:      7.1
Server Platform:    Digital OpenVMS Alpha
Max Shared Mem Size: 2000 Kb
Config file:        SYS$SYSROOT:[SYSMGR]SQLSRV_CONFIG_FILE71.DAT;1
Log path:           SYS$MANAGER:
Dump path:          SYS$MANAGER:
Proc start time:    <none>
Proc shut time:    <none>
Network Ports:                                           (State)  (Protocol)
```

```
DECnet object  SQLSRV_SERVER                Running  Native
TCP/IP  port   2199                          Running  Native
Current shared memory usage:
Allocation unit:    65536 bytes
Total memory:      2031616 bytes ( 31 units)
Free memory:       1769472 bytes ( 27 units)
Partly allocated:  196608 bytes ( 3 units)
Log File:          SYS$SYSROOT:[SYSMGR]SQS_CRANES_SQLSRV_MON_0071.LOG;
Dump File:         SYS$SYSROOT:[SYSMGR]SQS_CRANES_SQLSRV_71.DMP;
```

SHOW SERVICE Command

Shows the static definition of a service object or all service objects currently defined in the configuration file.

Format

```
SHOW SERVICE[S] [ <service-spec> ] [ FULL ] ;
```

```
<service-spec> ::= { * | <service-name-list>
```

```
<service-name -list> ::= { <identifier> [ , <identifier> ] ... }
```

Arguments

<service-spec>

The service object specification. This can be one or more service object names or can be specified with an asterisk (*). If one or more service object names are specified, then only information for those named service objects is displayed. If an * is specified, information for all service object names is displayed.

FULL

Displays a full description of information for each service. The default is to display brief information (one line of output) for each service.

Usage Notes

- SERVICE is a synonym for the keyword SERVICES.
- If no service object is specified, then information for *all* service object names is displayed.
- The SHOW SERVICE * command and the SHOW SERVICE command both show you all of the services currently defined.
- The service state can be one of five possible states:
 - Starting – The service is starting.
A service with MIN_EXECUTORS set to 0 never shows the Starting state when the service starts up. The state displays as either Running or Failed.

- Failed – The service failed to start.
 - Running – The service is running.
 - Inactive – The service is shut down.
 - Unknown – The management client is not connected to the server online so it cannot determine the state of the service. The management client used the SET CONFIG_FILE command to manage the server offline. Use the CONNECT TO SERVER command to connect to the server online to determine the service state.
- When a difference exists for an attribute between the running service and its configuration file, the SHOW SERVICE command displays these differences at the end of the show output and indicates that when the service is restarted, the running server's service is updated to match the server's service definition in the configuration file.
 - If FULL is specified, this command shows the list of user names and identifiers granted access to the specified services.

Examples

Example 1: Show the services defined for a configuration and display a brief description.

```
SQLSRV> SHOW SERVICES;
```

Name	State	C l i e n t s			E x e c u t o r s		
		Per-Exec	Max	Active	Min	Max	Running
SQLSRV_MANAGE	UNKNOWN	100	100	1	1	1	0
GENERIC	UNKNOWN	1	10	0	2	10	1
RMU_SERVICE	UNKNOWN	1	100	0	0	100	0

Example 2: Show the payroll service defined for a configuration and display a full description.

```
SQLSRV> SHOW SERVICE payroll FULL;
```

```
Service PAYROLL
```

```
State:                RUNNING
Owner:                PAYROLLACCNT
Protocol:             SQL/Services
Default Connect Username: <not specified>
SQL version:         7.1
Autostart:            on
Process init:         <not specified>
Attach:               ATTACH 'FILENAME PAYROLL_DISK:PAYROLL_DB'
Schema:               <not specified>
Reuse:                SESSION
Database Authorization: CONNECT USERNAME
dbsrc file:           <not specified>
```

SHOW SERVICE Command

```
SQL init file:           <not specified>
Appl Transaction Usage:  SERIAL
Idle User Timeout:      <none>
Idle Exec Timeout:      1800 seconds
Min Executors:          5
Max Executors:          10
Clients Per Executor:   1
Active Clients:         0
```

Access to service PAYROLL

Granted to users:

```
PRIVILEGED_USER 'PAYROLLACCNT'
```

Granted to identifiers:

```
'PAYROLL_DBA' 'PAYROLLDEPT'
```


SHOW SETTINGS Command

Shows information about the current SQLSRV_MANAGE settings.

Format

SHOW SETTINGS;

Usage Notes

After starting the server, use the SHOW SETTINGS command to determine the current settings for the SQLSRV_MANAGE environment. Modify these SQLSRV_MANAGE environment settings for your own use.

Examples

Example 1: Show information about the current settings for the SQLSRV_MANAGE environment.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> SHOW SETTINGS;  
Settings:  
  version:    v7.1  
  verify:    off  
  output:    on  
  config-file:  SYS$SYSROOT:[SYSMGR]SQLSRV_CONFIG_FILE.DAT;1  
  confirm:    on
```

SHOW VERSION Command

Shows the version of the SQLSRV_MANAGE management client.

Format

SHOW VERSION;

Usage Notes

Use the SHOW VERSION command to determine the version of the SQLSRV_MANAGE management client.

Examples

Example 1: Show the version of the SQLSRV_MANAGE management client.

```
SQLSRV> SHOW VERSION;  
Version:          v7.1-6
```

SHUTDOWN DISPATCHER Command

Shuts down the specified dispatcher.

Format

```
SHUTDOWN DISPATCHER          <dispatcher-name> ;  
  
                                <dispatcher-name> ::= <identifier>
```

Arguments

<dispatcher-name>

Specifies the dispatcher object name. The dispatcher object name is expressed as an identifier.

Usage Notes

- Use the SHOW CLIENTS command to ensure no clients are connected to a service using a network transport being provided by the dispatcher that you are shutting down.
- You can shut down a dispatcher only as an online operation; that is, you must be connected to a running server (CONNECT TO SERVER command) to shut down a dispatcher defined and running for that server.
- A dispatcher that has failed to start is left in a failed state and must be shut down. Correct the problem (usually an argument value is incorrectly specified), then start the dispatcher again.

Examples

Example 1: Shut down the dispatcher named disp_tcpip.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> SHOW CLIENTS;  
SQLSRV> SHUTDOWN DISPATCHER disp_tcpip;
```

SHUTDOWN SERVER Command

Shuts down the current server.

Format

SHUTDOWN SERVER;

Usage Notes

- Use the SHOW CONNECT[ION] command to ensure that you are shutting down the correct server.
- You can shut down a server only as an online operation; that is, you must be connected to a running server (CONNECT TO SERVER command) to shut it down.

Examples

Example 1: Shut down the current server.

```
SQLSRV> CONNECT SERVER;
Connecting to server ...
Connected
SQLSRV> SHOW CONNECT;
Active connections:
CURRENT: SQLSRV_MANAGE
      Service: SQLSRV_MANAGE
      User: run_username Node: EAGLE Local: Yes
      Transport: DECNET Object: SQLSRV_SERVER
      Request bufsize: 1024 Response bufsize: 1024
SQLSRV> SHUTDOWN SERVER;
Disconnected from Server
```

SHUTDOWN SERVICE Command

Shuts down the specified service.

Format

```
SHUTDOWN SERVICE      <service-name> ;  
  
                        <service-name> ::= <identifier>
```

Arguments

<service-name>

Specifies the service object name. The service object name is expressed as an identifier.

Usage Notes

- Use the SHOW CLIENTS command to ensure that no clients are connected to the service that you are shutting down.
- You can shut down a service only as an online operation; that is, you must be connected to a running server (CONNECT TO SERVER command) to shut down a service defined and running for that server.
- A service that has failed to start is left in a failed state and must be shut down. Correct the problem (usually an argument value is incorrectly specified), then start the service again.

Examples

Example 1: Shut down the universal service named generic.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> SHUTDOWN SERVICE generic;
```

START DISPATCHER Command

Starts a dispatcher process for the defined dispatcher object with the specified name for the current server.

Format

```
START DISPATCHER <disp-name> ;  
  
                <disp-name> ::= <identifier>
```

Arguments

<disp-name>

Specifies the dispatcher name. The dispatcher name is expressed as an identifier.

Usage Notes

You can start a dispatcher only as an online operation; that is, you must be connected to a running server (CONNECT TO SERVER command) to start a dispatcher defined for that server.

Examples

Example 1: Start the tcpip_disp dispatcher.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> START DISPATCHER tcpip_disp;
```

START SERVER Command

Starts the current server and optionally any defined dispatcher and service objects for the current server when AUTOSTART is set as ON and then connects to the server after starting it when AUTOCONNECT is set as ON.

Format

START SERVER

->[USER { <user-name> } USING { <password> }]

->[AUTOCONNECT { ON | OFF }]

->[AUTOSTART { ON | OFF }] ;

<user-name> ::= { <quoted-string> | <identifier> }

<password> ::= { <quoted-string> | <identifier> }

Arguments

USER {<user-name> USING <password>}

Specifies a user name and password with which to connect to the server after the server has started. The user name and password are expressed as either quoted-strings or identifiers. You need not specify a user name and password if you are starting a server running DECnet or if you are starting a server running TCP/IP and you have SYSPRV or BYPASS privilege.

AUTOCONNECT {ON | OFF}

Determines whether or not to automatically connect to the server after you issue a START SERVER command. If the argument is specified as ON, SQLSRV_MANAGE automatically connects to the server after you issue a START SERVER command. A value of OFF starts the server but does not attempt to connect to the server after it has started. The default is ON.

AUTOSTART {ON | OFF}

Determines whether all other server objects (dispatchers and services) automatically start up when you issue a START SERVER command. If the argument is specified as ON, the default, all other server objects automatically start if each object's AUTOSTART argument value is also set as ON. If you do not want to start all other server objects, specify the AUTOSTART attribute value as OFF in the START SERVER command. The AUTOSTART

OFF attribute setting overrides each object's AUTOSTART attribute setting and allows you to individually start each object after starting just the server object. The default is ON.

Usage Notes

- You can start a server only as an offline operation; that is, you must use the SET CONFIG_FILE command to select the configuration file of the server you want to start or use the default.
- After the server starts up with the AUTOCONNECT argument specified as ON, SQLSRV_MANAGE attempts to connect to any network port defined for the server. It tries each network port in a round-robin fashion up to three times each to establish the connection.
- You must have the SETPRV privilege or all privileges.
- When SQLSRV_MANAGE starts up, it establishes a default configuration file name:
 - The default configuration file is:
`SYS$MANAGER:SQLSRV_CONFIG_FILE71.DAT`
 - To override the default, set the SQLSRV_CONFIG_FILE71 logical name or supply a different file name to the SET CONFIGURATION_FILE command.

Examples

Example 1: Start the current server.

```
SQLSRV> SET CONFIG_FILE 'my_config_file';
SQLSRV> START SERVER;
Server started
Connecting to server ...
Connected
```


START SERVICE Command

Starts the specified, defined service object for the current server.

Format

```
START SERVICE    <service-name> ;  
  
                  <service-name> ::= <identifier>
```

Arguments

<service-name>

Specifies the service name. The service name is expressed as an identifier.

Usage Notes

You can start a service only as an online operation; that is, you must be connected to a running server (CONNECT TO SERVER command) to start a service defined for that server.

Examples

Example 1: Start the universal service named V71.

```
SQLSRV> CONNECT SERVER;  
Connecting to server ...  
Connected  
SQLSRV> START SERVICE v71;
```

Logging and Troubleshooting

This section describes how to enable logging for SQL/Services and OCI Services for Oracle Rdb, and how to identify and fix some of the more common errors.

You can isolate problems that you might experience with an Oracle SQL/Services server system by:

- Inspecting log files
- Investigating different types of problems

The following sections describe what log files are generated, what is contained in each type of log file, a number of different error conditions that you may encounter, and how to identify a particular problem.

7.1 Problem Reporting

If an error occurs while you are using Oracle SQL/Services or OCI Services for Oracle Rdb and you believe that the error is caused by a problem with an Oracle product, contact your Oracle support representative for technical assistance.

When you experience a reproducible problem, it is important to provide as much detailed information as possible. Enable full logging to collect detailed information about the current session. Include the following items along with your problem report:

- Copies of the Oracle SQL/Services monitor log files, dispatcher log files, any applicable executor log files, and any relevant client log files
- A copy of the Oracle SQL/Services configuration file
- Copies of any bugcheck dump files produced

7.2 Error Messages

The language you specify determines which language is used for Oracle error messages and boilerplate text, including month and day names in dates. By default, Oracle Rdb error messages are supplied in English, unless you use a specially translated error message file and define the file to the Rdb database. Oracle error message files are provided in all of the supported languages listed in [Table 4–1](#).

For Oracle Rdb error messages, the language specified when you attach to the database is used for the duration of the attach.

Sorting, collating, and comparisons are done according to the Oracle Rdb semantics. Refer to the *Oracle Rdb SQL Reference Manual* for more information.

The server error message files contain all of the server errors with explanations of the error and possible user actions. PostScript and text versions of the server error message files are located in the following directories:

- `SY$$HELP:SQLSRV_MESSAGES71.PS` - Oracle SQL/Services server error message PostScript file
- `SY$$HELP:SQLSRV_MESSAGES71.TXT`

7.3 Log Files on the Server

This section describes how to enable and disable logging on the server, and how to use the log files.

There are several kinds of log files on the server side:

- Oracle SQL/Services monitor log file
- Oracle SQL/Services dispatcher log files
- Oracle SQL/Services executor log files

Oracle SQL/Services uses the following convention to generate log file names for server components, where `nodename` is the node name, `component-id` is the server component, and `version` is the version number:

- If the `SCSNODE SYSGEN` parameter is set
`sq_s_<nodename><component-id><instance><version>.log`
- If the `SCSNODE SYSGEN` parameter is blank
`sq_s_<component-id><instance><version>.log`

7.3.1 Oracle SQL/Services Monitor Log File

Oracle SQL/Services logs the following information in the monitor log file:

- Dispatcher and executor process startup and shutdown informational messages
- Dispatcher and executor process failure error messages, including names and locations of component log files
- Oracle SQL/Services authentication and authorization failures for Oracle SQL/Services system management clients
- Name and location of a monitor process bugcheck dump if the monitor encounters a nonrecoverable error

Use the following command to list monitor log files:

```
$ DIRECTORY SYS$MANAGER:SQS*MON*.LOG
```

For example:

```
SYS$MANAGER:SQS_NODE1_SQLSRV_MON_0071.LOG
```

7.3.2 Oracle SQL/Services Dispatcher Log Files

Oracle SQL/Services logs the following information in a dispatcher log file:

- Oracle SQL/Services authentication and authorization failures for Oracle SQL/Services, OCI clients, and Oracle RMU clients
- Server-side client network link disconnections due to executor process failures
- Client-side client network link failures
- Name and location of a dispatcher process bugcheck dump if the dispatcher encounters a nonrecoverable error

Use the following command to display the location and file names for the dispatcher log files:

```
SQLSRV> SHOW DISPATCHER <dispatcher_specification>
```

For example:

```
SQLSRV> SHOW DISPATCHER SQLSRV_DISP
```

```
Dispatcher SQLSRV_DISP
State: RUNNING
Autostart: on
```

```
Max connects: 100 clients
Idle User Timeout: <none>
Max client buffer size: 5000 bytes
Network Ports: (State) (Protocol)
DECnet object 81 Running SQL/Services
TCP/IP port 119 Running SQL/Services
Log path: SYS$MANAGER:
Dump path: SYS$MANAGER:
Log File: SYS$SYSROOT:[SYSMGR]SQS_NODE1_SQLSRV_DIS00371.LOG;
Dump File: SYS$SYSROOT:[SYSMGR]SQS_NODE1_SQLSRV_DIS003.DMP;
```

The location and file name for the log file are shown in the line beginning with "Log File:."

Use the following command to list the log files:

```
$ DIRECTORY <directory_name>:<filename>
```

For example:

```
$ DIRECTORY SYS$SYSROOT:[SYSMGR]SQS*DIS*.LOG
```

```
SYS$SYSROOT:[SYSMGR]SQS_NODE1_SQLSRV_DIS00371.LOG
```

The following entries may be written to the dispatcher log file every time a SQLNET or OCI client successfully disconnects from Oracle SQL/Services:

```
-----EVENT BEGIN:  EVENT_LOG at Tue Oct  8 2004 18:40:01.055-----
%SQLSRV-I-EVENT_LOG, event logged at line 1674 in file COM_TNS.C:1
%SQLSRV-E-TNSFAILURE, SQL*Net TNS nsrecv() service has failed
%SQLSRV-E-TNSEXTENDED, SQL*Net TNS error codes: primary (12537) secondary (0)
-----EVENT END   :  EVENT_LOG at Tue Oct  8 2004 18:40:01.172-----

-----EVENT BEGIN:  EVENT_LOG at Tue Oct  8 2004 18:40:01.519-----
%SQLSRV-I-EVENT_LOG, event logged at line 284 in file DISP.C:1
%SQLSRV-W-EXCEPTION_RAISE, Exception raised: DBS_TNSFAILED
%SQLSRV-I-CONNECTNAME, Connect : CONNECT_0000001
%SQLSRV-I-CONNECTSTATE, Connect state: 4
%SQLSRV-I-USERNAME, User name: dtmtest1
%SQLSRV-I-NODENAME, Node : unknown
%SQLSRV-I-SERVICENAME, Service : NATCONN_OCI
-----EVENT END   :  EVENT_LOG at Tue Oct  8 2004 18:40:01.527-----
```

You can ignore these entries.

7.3.3 Oracle SQL/Services Executor Log Files

Oracle SQL/Services logs the following information in an executor log file:

- Executor process startup errors
- Oracle Rdb authentication and authorization failures for Oracle SQL/Services clients for database services with database authorization set to connect user
- Oracle Rdb and SQL error messages
- Name and location of an executor process bugcheck dump if the executor encounters a nonrecoverable error
- For OCI services, log messages for OCI Services for Oracle Rdb as specified by the ALTER SESSION LOG BRIEF or ALTER SESSION LOG FULL command, or the SQLNET_DEBUG_FLAGS logical name.

Executor log files are created in the default directory of the service owner account. For example, use the following commands to list executor log files for a service named GENERIC with a service owner account named SQLSRV\$DEFLT that has a default directory of SYS\$SYSDEVICE:[SQLSRV\$DEFLT].

```
$ DIRECTORY SYS$SYSDEVICE:[SQLSRV$DEFLT]SQS*GENERI*.LOG
```

```
SYS$SYSDEVICE:[SQLSRV$DEFLT]SQS_NODE1_GENERI004000171.LOG
```

7.3.4 Using Debug Flags in the Executor Log File for OCI Services for Oracle Rdb

Use debug flags to log additional information in the SQL/Services executor log file. To enable this feature, define the logical SQLNET_DEBUG_FLAGS in your SQL/Services service process initialization file. The following case-sensitive debug flags are currently supported:

- H
Causes routine header information to be logged whenever a routine is executed.
- T
Causes the current time to be logged whenever header information is logged. This flag is dependent on the H flag. If the H flag is not specified, time information is not logged.

For example, if the following line is inserted into your process initialization file, header information is logged:

```
$ DEFINE SQLNET_DEBUG_FLAGS "H"
```

Information such as the following appears in the executor log file. This information is

especially useful when you report problems to Oracle, for example:

```
... Server entered ...
GTZI_SERVER STARTUP UP -> Wed Aug  8 13:34:33 2004
*** GTZGPRD ***
*** GTAGPRD ***
NLS initialization ...
* GTWILAN: Setting LANGUAGE *
Language is AMERICAN_AMERICA.US7ASCII
```

If the following line is inserted into your process initialization file, header information and the current time is logged:

```
$ DEFINE SQLNET_DEBUG_FLAGS "HT"
```

Information such as the following appears in the executor log file:

```
... Server entered ...
Current Time = Wed Aug  8 13:34:33 2004

GTZI_SERVER START UP -> Wed Aug  8 13:34:33 2004

Current Time = Wed Aug  8 13:34:33 2004

*** GTZGPRD ***
Current Time = Wed Aug  8 13:34:33 2004

*** GTAGPRD ***
Current Time = Wed Aug  8 13:34:33 2004

NLS initialization ...
Current Time = Wed Aug  8 13:34:34 2004

* GTWILAN: Setting LANGUAGE *
Current Time = Wed Aug  8 13:34:34 2004
Language is AMERICAN_AMERICA.US7ASCII
Current Time = Wed Aug  8 13:34:34 2004
```

Avoid using the T debug flag unless you are trying to debug a timing or performance issue. Using the T flag adds an additional timestamp to every command and can make the log files exceptionally large.

7.3.5 Enabling Logging from SQL and Oracle Rdb

You can enable logging from SQL and Oracle Rdb by defining the `RDMS$DEBUG_FLAGS` logical. This logging will also be written to the executor log file.

Refer to the *Oracle Rdb Guide to Database Performance and Tuning* for more information.

7.3.6 Disabling Logging in SQL/Services

You can disable logging at the dispatcher level, or the executor level.

For dispatcher logging, use the `CREATE` or `ALTER` commands:

```
SQLSRV> CREATE DISPATCHER SQLSRV_DISP1 LOG PATH 'NOLOG'
_SQLSRV> DUMP PATH 'SYS$MANAGER';
or
```

```
SQLSRV> ALTER DISPATCHER SQLSRV_DISP1 LOG PATH 'NOLOG'
_SQLSRV> DUMP PATH 'SYS$MANAGER';
%SQLSRV-S-ALTER_RESTART
```

Restart the object to have altered settings take effect.

For executor logging:

Define the `SQLSRV_EXEC_LOG` logical name at the system level. For example:

```
$DEFINE/SYSTEM SQLSRV_EXEC_LOG NOLOG
```

7.3.7 Moving Dispatcher Log Files Off the System Disk

Starting in 7.1.5.5 of SQL/Services, you can move dispatcher log files off the system disk by using the `CREATE` or `ALTER` commands for the dispatcher. For example:

```
SQLSRV> ALTER DISPATCHER SQLSRV_DISP1 LOG PATH 'USER1:[SQLSRV_TEST1.AAA]'
_SQLSRV> DUMP PATH 'USER1:[SQLSRV_TEST2.BBB]';
%SQLSRV-S-ALTER_RESTART
```

Restart the object to have altered settings take effect.

7.4 Enabling Full Logging for OCI Services for Oracle Rdb

To enable full logging for OCI Services for Oracle Rdb:

1. Logging for the executors must be enabled.

2. Create a SQL.INI file in the service owner's login directory. The file should contain the following line:

```
ALTER SESSION LOG FULL;
```

3. Alter the service so that the SQL INIT parameter points to this file:

```
SQLSRV> ALTER SERVICE FOO SQL_INIT_FILE "<pathname>:SQL.INI";
```

4. Restart the service:

```
SQLSRV> SHUTDOWN SERVICE FOO;  
SQLSRV> START SERVICE FOO;
```

7.5 Inspecting SQL/Services API Log Files

When a problem arises, you can attempt to isolate the problem by inspecting the log files generated on the client side as well as those generated on the server side.

There are up to four kinds of logging on the client side:

- Client and driver logging
- Winsock logging
- Oracle Net logging
- ODBC tracing

7.5.1 Client and Driver Logging

Client and driver logging is used by both Oracle SQL/Services and the Oracle ODBC Driver for Oracle Rdb.

Enable Oracle SQL/Services client logging by using a parameter in the `sqlsrv_associate` routine, or by using an `sqsapiw.ini` file or an `sqsap32.ini` file for a Windows 95 or later client. The Oracle SQL/Services client API creates log files named `clientxx.log` in the application's default directory. A `clientxx.log` file records calls to Oracle SQL/Services, API services, data values, and message protocol.

Check a `clientxx.log` file to see what SQL statements you are passing to the server, or what error messages you are getting back from the server. See the *Guide to Using the Oracle SQL/Services Client API* for more information.

If you are using the Oracle ODBC Driver for Rdb, you can turn on ODBC logging by changing the default settings for client logging and driver logging in the `oraodbc.ini` file

(rdbodbc.ini for the Version 2 driver). The default value is 0, which turns off logging. All valid values are described in the sqldb.hlp help file that comes with the driver.

The client logging traps the Oracle SQL calls that are passed from the client through the Oracle SQL/Services API to the server. The information is written to log files named clientxx.log where xx is a value from 01 to 99. Do not let the number exceed 99 because it is not reset automatically.

The driver logging traps the driver calls that are made in the process of passing information to the server. The information is written to the rdbodbc.log file that is located in the application directory.

7.5.2 Winsock Logging

If you are using a Winsock transport on Windows 95 or later, this logging option helps you diagnose network problems. To enable this option, uncomment the line "; Winsock Logging = 1" in the sqsapi32.ini file. This logging traces the Winsock or network calls that are made when the query is passed. The log file is named sqsapiw.log and is located in the working directory of your application.

7.5.3 Oracle Net Logging

If you are using an OCI service, you can enable Oracle Net logging by defining the TRACE_LEVEL_CLIENT or TRACE_LEVEL_SERVER in the SQLNET.ORA file in your Oracle environment.

Follow these steps to set tracing parameters using component configuration files:

1. Execute or start the component to be traced.

Set the following trace parameters in the component configuration file: SQLNET.ORA for client or server, LISTENER.ORA for listener:

```
TRACE_LEVEL_<CLIENT | LISTENER | SERVER> = (0 | 4 | 10 | 16)
TRACE_DIRECTORY_<CLIENT | LISTENER | SERVER> = <directory_name>
TRACE_FULL_<CLIENT | LISTENER | SERVER> = <directory_name>
```

The following table describes the output for each trace level:

Trace Level	Output
0 or OFF	The default. No trace output

Trace Level	Output
4 or USER	User trace information
10 or ADMIN	Administration trace information
16 or SUPPORT	WorldWide Customer Support trace information

The default directory is the login directory of the process owner on OpenVMS, or the current directory of the executable image on Microsoft platforms. SQLNET.LOG and SQLNET.TRC are the default file names. You can change both the file name and the directory location through the Oracle Network Manager when you create or change an Oracle Net configuration. Do this for any client process by editing the Logging and Tracing tab of a client profile object.

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

7.5.4 ODBC Tracing

To enable ODBC tracing, select the TRACING tab in the ODBC administrator's application. The default file name is SQL.LOG; you must specify where to create the file. Click on "Start tracing now" to turn tracing on. This log captures the SQL that the ODBC driver is sending to Oracle SQL/Services.

7.6 Process Failures

Oracle SQL/Services handles process failures in different ways depending on the type of process that fails.

7.6.1 Monitor Process Failures

Oracle SQL/Services does not attempt to recover if the monitor process fails. If a monitor process does fail, then all of the processes in the server configuration are shut down. In this way, a monitor process failure does not leave the system in an inconsistent state.

7.6.2 Dispatcher Process Failures

Oracle SQL/Services does not automatically restart a failed dispatcher process; however, the server will continue running unless the failure occurred during a critical operation in which the integrity of the server might be compromised. Therefore, you must manually restart a failed dispatcher process.

7.6.3 Executor Process Failures

Oracle SQL/Services automatically tries to restart a failed executor process unless the failure occurred during a critical operation in which the integrity of the server might be compromised. However, if an executor process fails more than twice during startup, then Oracle SQL/Services shuts down the executor unless the `SQLSRV$MAX_EXECUTOR_FAILURES` logical is defined. If there are no other active executors for the service, it also shuts down the service and marks it as failed.

You can define the `SQLSRV$MAX_EXECUTOR_FAILURES` logical to change the maximum number of failures from the default of two to any positive integer value. In this way, you can control how often executors and services shut down during routine database maintenance.

7.7 Investigating Different Types of Problems

As a system administrator, you may be called upon to investigate a number of different types of problems. The following is a set of general error conditions with guidelines for each that may help you track down and identify a particular problem.

7.7.1 Network Transport Problems

A problem sometimes experienced by new users or with a new server configuration is the inability to connect to the server at all. In this situation, client applications receive network errors from Oracle SQL/Services API routines and OCI Services for Oracle Rdb routines.

In the event of this type of error, first verify that the dispatcher supporting the selected transport is running and that the specified network port or object is active. If you are using alternate network ports or objects in a multiversion environment, verify that you specified the correct network port or object at the client. The `SHOW SERVER` command in the `SQLSRV_MANAGE` utility shows all network port numbers. Make sure that your client application is using the correct port. If the dispatcher appears to be functioning correctly, use a transport-specific tool, such as the TCP/IP Ping utility, to verify connectivity between the client and server nodes.

If the dispatcher is not running or the selected network port or object is not active, check the dispatcher log to determine the reason for the problem. If a dispatcher process fails completely, then Oracle SQL/Services writes the name and location of the dispatcher log file to the monitor log. Check the dispatcher log file to determine if a bugcheck dump was produced when the dispatcher failed.

7.7.2 User Authentication and Authorization Problems

Authentication and authorization errors are another class of problems that may be experienced by new users. In this case, the server is functioning correctly. However, users are unable to connect to the server or to a particular service provided by the server.

You should first check the dispatcher log file to determine the reason for the error. Remember to check the appropriate dispatcher log if you have configured multiple dispatchers for different transports.

For example, for SQLSRV executors, to resolve an authentication or authorization problem, you may need to authorize network access or grant the SQLSRV\$CLIENT identifier to a user's account. Or perhaps you need to grant access to a particular service to a new user.

For OCI Services for Oracle Rdb executors, a problem could occur when a service owner does not have attach privileges in the database (SELECT), a connect user is not included in the USER\$stable, or the user does not have required Oracle Rdb privileges to access the database or tables.

All of these types of errors are logged in the dispatcher log file.

If the user is connecting to a database service with database authorization set to connect user, then you must also authorize the user's account to access the database. If a user is not authorized to access the database, then Oracle Rdb returns a no privilege (-1008) error, the text of which Oracle SQL/Services returns to the client application and writes to the executor's log file.

7.7.3 Executor Failures During Service Startup

You may sometimes encounter errors when initially creating and starting a new service. Whenever an executor process fails to start correctly, Oracle SQL/Services writes the name of the executor's log file to the monitor log. From the executor log, you can then find the reason for the error.

For example, to determine why a new service fails after you start it, display the contents of the monitor log to determine the log file names of the failed service's executors. Then display the contents of one of the executor log files to determine the reason for the failure. You may have typed an invalid SQL ATTACH statement, mistyped the database file name, or did not grant the right to attach to the database to the service owner account of a database service. All problems such as these result in the service's SQL ATTACH statement failing.

7.7.4 Executor Problems During Client Connect

In some situations, a service may start successfully, but an executor process created for a new client connection might fail during startup. This can happen if the MIN_EXECUTORS

attribute of a service is set to 0. In this case, you can successfully start the service, but the service eventually changes to the failed state as executors created for new client connects fail during startup. This problem can also occur if a database is changed after a service is started. For example, if the right to attach to the database is revoked from the service owner account of a database service after the service is started and the minimum number of executors has been created, then new executors that are created for the service will fail trying to execute the service's ATTACH statement.

If a user tries to connect to a service and an executor created for the new connect fails during startup, the monitor records the executor failure event in the monitor log together with the name of the executor log. The dispatcher then logs a summary error message in the dispatcher log and returns the executor failed (-2035) error code to the client application along with an executor startup error message. If a user tries to connect to a service that previously changed to the failed state, then the dispatcher logs the event in the dispatcher and returns the executor failed (-2035) error code to the client application along with a service failed error message.

To investigate problems of this nature, first check the dispatcher log to determine why new client connects are being rejected. Then review the monitor log to find an entry detailing an executor failure for the service. Finally, check the executor log to determine the reason for the failure.

7.7.5 Executor Problems During Client Request Execution

You may experience a situation where most users are successfully accessing a service, but the executor for one particular user fails. In this situation, the dispatcher returns the executor failed (-2035) error to the client application and the monitor records the executor failure event in the monitor log, together with the name of the executor log. You first check the monitor log to determine the name of the log file for the failed executor, then check the log of the executor to determine the reason for the failure. For example, perhaps data being accessed by a particular user is located on a disk that is beginning to fail. Alternatively, perhaps Oracle SQL/Services or Oracle Rdb or SQL encountered an internal error. In this situation, check the executor log file to see if a bugcheck dump file was produced by one of these components.

7.7.6 Server Failed Due to an Internal Error

In extremely rare circumstances, it is possible for an entire server to fail. For example, perhaps a component encountered an internal error and failed while performing a critical operation. In this situation, the entire server shuts down so as not to further compromise the integrity of the server configuration.

The Oracle SQL/Services monitor process manages all of the processes in an Oracle SQL/Services server configuration. Therefore, the monitor log file is the best place to start. In this situation, the monitor will always produce a bugcheck dump; however, the reason for the error may have been the earlier failure of a dispatcher or executor process. Therefore, your next step is to review the log files of any dispatchers and executors that failed just prior to the server failure. Check these log files for references to any Oracle SQL/Services and Oracle Rdb and SQL bugcheck dumps.

If you find a reference to a bugcheck dump file while isolating a problem, refer to [Section 7.1](#) for more information about submitting a problem report form to Oracle. The bugcheck dump file is directed by default to SYSS\$MANAGER unless you specified another location by using the CREATE SERVER, ALTER SERVER, CREATE DISPATCHER, or ALTER DISPATCHER command. The only exception is that executor bugcheck dump files are written to the default directory of the connect user or service owner, depending on the database authorization.

7.7.7 Connections from Clients No Longer Work After Installing Oracle SQL/Services

When you install the Oracle SQL/Services client API in a multiversion environment, you are asked to modify the network port numbers to something other than the default. These non-default port numbers must be specified in an sqsapi32.ini file on each client accessing this version of SQL/Services. Each client must modify the file by specifying:

- The node to which they are connecting
- The network transport they are using
- The port on which the network is listening

For example, Node NODE1 has TCPIP configured for port 119. The sqsapi32.ini entry looks like this:

```
[NODE1]
TCPIPPortNumber=119
```

7.7.8 Network Errors

When you receive the primary QLSRV_NETERR error, look at the network error documentation for the network error referred to in the secondary error message. For example, [Table 7-1](#) and [Table 7-2](#) contain platform-specific error information for DECnet and TCP/IP, respectively. You should also look at your own platform-specific documentation for more information on the secondary error code resulting from a network error.

Information about DECnet error codes can be found at the locations listed in [Table 7-1](#).

Table 7-1 Error Code Files for DECnet

Operating System	File Specification	Description
OpenVMS	SYSSLIBRARY:SSDEF.H	System service return status code definitions for DECnet
Windows 95 and later	skerrno.h	DECnet error codes (check PATHWORKS SDK)

Information about TCP/IP error codes can be found at the locations listed in [Table 7-2](#).

Table 7-2 Error Code Files for TCP/IP

Operating System	File Specification	Description
OpenVMS	SYSSLIBRARY:ERRNO.H	System service return status code definitions for TCP/IP
Windows 95 and later	winsoc.h	TCP/IP error codes (check Microsoft Windows 95 SDK or Microsoft C++)

7.8 Error Messages Returned to OCI Client Applications

This section describes error messages that are frequently encountered and returned to OCI client applications. Other error messages are described in the *Oracle Database Error Messages* manual.

7.8.1 Logon Error

ERROR: ORA-01017: invalid username/password; logon denied

Cause: You supplied invalid logon information. This can also happen if there is no entry in the USER\$ table for the user name and password, or if program validation is enabled and there is no corresponding entry in the ORA_VALID_PROGRAMS table.

Action: Log in again supplying the correct information. For more information, see the Oracle SQL/Services executor log file.

7.8.2 Database Setup Error

ERROR: ORA-00904: invalid column name

Database not setup correctly for OCI Services for Oracle Rdb

For details, look in Oracle SQL/Services executor log file

<...full file specification of the executor log...>

Cause: The database is not set up correctly for OCI Services for Oracle Rdb.

Action: For more information about this error, see the Oracle SQL/Services executor log file. See [Section 4.1](#) in *Oracle Rdb SQL/Services Release 7.1.6 Installation Guide* for instructions to help you prepare your database.

7.8.3 SQL Initialization File Error

ERROR: ORA-00900: invalid SQL statement

Error in executing SQL initialization file

For details, look in Oracle SQL/Services executor log file

<...full file specification of the executor log...>

Cause: Error when executing SQL initialization file.

Action: For details, see the Oracle SQL/Services executor log file.

7.8.4 Errors Attaching to an Rdb Database or with Oracle SQL/Services Database Service

ERROR: ORA-03113: end-of-file on communication channel

This error is returned due to a variety of reasons. See the Oracle SQL/Services executor log file, the monitor log file, and the dispatcher log file for more information. See [Section 7.3](#) for information about inspecting these log files. The following descriptions show some common reasons for this error. Inspect the log file for the actual cause.

Cause: The Oracle SQL/Services service did not start, or it is unavailable.

Action: Check to see if the service was started, and if not, start the service. If the service failed, check the Oracle SQL/Services monitor log file for a pointer to the failed executor log. Then, check the executor log file for more information.

Cause: The Oracle SQL/Services service name requested is invalid.

Action: Look in the Oracle SQL/Services dispatcher log file to see if there is an error entry about the service being requested. If the Oracle SQL/Services service name requested is invalid, modify the Oracle SQL/Services service name in the Oracle Net configuration file TNSNAMES.ORA accordingly.

Cause: The protocol is not set to OCI.

Action: Alter your dispatcher and specify PROTOCOL OCI. Then, stop and restart your Oracle SQL/Services dispatcher so the change can take effect.

7.9 Errors When Oracle SQL/Services Server or OCI Dispatcher Is Not Available

7.9.1 ERROR: ORA-12203: TNS:unable to connect to destination

Cause: Oracle SQL/Services and the OCI_DISP dispatcher are not started.

Action: Start Oracle SQL/Services and the OCI_DISP dispatcher.

Cause: The OCI_DISP dispatcher failed.

Action: If the dispatcher failed, look in the Oracle SQL/Services monitor log file for a pointer to the failed dispatcher log. Then, look in the dispatcher log file for the failure reason.

If you see the following entry in the dispatcher log file, the dispatcher cannot find the definition of the listener name used in the Oracle SQL/Services dispatcher specification:

```
%SQLSRV-E-TNSFAILURE, Oracle SQL*Net TNS nlpagas() service has failed  
%SQLSRV-E-ERROR_TEXT, Error text: listener
```

Check the Oracle Net configuration file LISTENER.ORA. The location of this file varies depending on your installation. The subdirectory containing your Oracle Net configuration file is stored in the following location:

```
SYS$MANAGER:SQLSRV_SQLNET $nn$ .DAT
```

nn represents the Oracle SQL/Services version number.

The [.NETWORK.ADMIN] subdirectory under the location stored in this file contains your LISTENER.ORA file.

7.9.2 Error When Oracle Net Service Name Is Not Defined

ERROR: ORA-12154: TNS:could not resolve service name

Cause: The Oracle Net service name might be defined improperly.

Action: Check the Oracle Net configuration files to see if the Oracle Net service name is defined properly. If you are using a file-based Oracle Net configuration, look in the

TNSNAMES.ORA file. The location of this file varies depending on where your client system is installed:

- On an OpenVMS system, the subdirectory containing the location of your configuration file is stored in the following location:

```
SYS$MANAGER:SQLSRV_SQLNETnn.DAT
```

nn represents the Oracle SQL/Services version number.

The [.NETWORK.ADMIN] subdirectory under the location stored in this file contains your TNSNAMES.ORA file.

- On any Microsoft Windows system, the location is
<Oracle installation directory>\network\admin

For example

```
c:\orawin\network\admin
```

Note: In some situations, Oracle clients are unable to interpret the error codes returned by Oracle SQL/Services. Therefore, you should always check the Oracle SQL/Services log files for accurate explanations.

Tables and Relations

This appendix contains information about:

- Oracle metadata views that emulate the Oracle data dictionary
- Current session information stored in the `ORA_SESSION` table
- Recommendations for 31-character table names
- Table and view definitions needed to store application data in your Oracle Rdb database

A.1 Oracle Metadata Objects

After you install OCI Services for Oracle Rdb, you must prepare each Oracle Rdb database you are going to use. The PREPARE process:

- Creates a set of procedures and functions that OCI Services for Oracle Rdb uses to help manage your session.
- Creates a large set of relations and views used to emulate the Oracle data dictionary.

These relations and views are how OCI client applications select Oracle metadata. The set of relations and views is not complete and neither is the data in them, because not all pieces of Oracle metadata exist in an Oracle Rdb database.

You can see the definition for the views, functions, and procedures by invoking interactive SQL and issuing these statements: `SHOW VIEW`, `SHOW PROCEDURE`, or `SHOW FUNCTION`. After you do that, you can determine which columns in the views are fixed values and which have meaningful information.

The database objects shown in the following table are created to emulate the Oracle metadata.

Note: The objects marked with an asterisk are created only for Oracle Rdb release 7.0 databases to provide emulation of the Oracle sequences functionality. Sequences are implemented natively in Oracle Rdb release 7.1.

Table A–1 OCI Services for Oracle Rdb Objects

Tables

FRM50_BUFFER	GLOBAL_NAME	ORA_ALL_USERS
ORA_COMM_TRANS	ORA_OBJECTS	*ORA_SEQUENCES
ORA_VALID_PROGRAMS	PRODUCT_PROFILE	ROLE_SYS_PRIVS
ROLE_TAB_PRIVS	ROSLFDESC	SESSION_ROLES
SNAP\$	SQL_CAPABILITY	TOOL_ACCESS
TOOL_LRAW	TOOL_MODULE	USERS\$
USER_ROLE_PRIVS	USER_SYS_PRIVS	USER_TABLESPACES
V\$OPTION	VG_COLOR	VG_IMAGE

Temporary Tables

DUAL	ORA_DBMS_OUTPUT	ORA_SESSION
V\$NLS_PARAMETERS	V\$SESSION	V\$SQLAREA
V\$VERSION		

Views

ACCESSIBLE_COLUMNS	ALL_ALL_TABLES	ALL_ARGUMENTS
ALL_CATALOG	ALL_COLL_TYPES	ALL_COL_COMMENTS
ALL_COL_PRIVS	ALL_CONSTRAINTS	ALL_CONS_COLUMNS

Views

ALL_DB_LINKS	ALL_INDEXES	ALL_IND_COLUMNS
ALL_OBJECTS	ALL_OBJECT_TABLES	ALL_SEQUENCES
ALL_SNAPSHOTS	ALL_SYNONYMS	ALL_TABLES
ALL_TAB_COLUMNS	ALL_TAB_COMMENTS	ALL_TAB_PRIVS
ALL_TRIGGERS	ALL_TRIGGER_COLS	ALL_TYPES
ALL_USERS	ALL_VIEWS	CAT
DBA_CATALOG	DBA_COL_COMMENTS	DBA_COL_PRIVS
DBA_CONSTRAINTS	DBA_CONS_COLUMNS	DBA_DB_LINKS
DBA_OBJECTS	DBA_SNAPSHOTS	DBA_SYS_PRIVS
DBA_TABLES	DBA_TAB_COLUMNS	DBA_TAB_PRIVS
DBA_USERS	NLS_DATABASE_PARAMETERS	NLS_INSTANCE_PARAMETERS
NLS_SESSION_PARAMETERS	PRODUCT_COMPONENT_VERSION	PRODUCT_PRIVS
PRODUCT_USER_PROFILE	USER_ARGUMENTS	USER_CATALOG
USER_COL_COMMENTS	USER_COL_PRIVS	USER_CONSTRAINTS
USER_CONS_COLUMNS	USER_DB_LINKS	USER_INDEXES
USER_IND_COLUMNS	USER_OBJECTS	USER_SEQUENCES
USER_SNAPSHOTS	USER_SYNONYMS	USER_TABLES
USER_TAB_COLUMNS	USER_TAB_COMMENTS	USER_TAB_PRIVS
USER_TRIGGERS	USER_TRIGGER_COLS	USER_USERS
USER_VIEWS	V\$PARAMETER	

Procedures

ORA_APPLINFO_READ_CLIENT_INFO	ORA_APPLINFO_READ_MODULE	ORA_APPLINFO_SET_ACTION
-------------------------------	--------------------------	-------------------------

Procedures

ORA_APPLINFO_SET_CLIENT_INFO	ORA_APPLINFO_SET_MODULE	ORA_CREATE_DBLINK
ORA_CREATE_ERROR	ORA_CREATE_OBJECT	*ORA_CREATE_SEQUENCE
ORA_CREATE_TRANS	ORA_CREATE_USER	ORA_CVT_ROWID_TO_CHAR
ORA_CVT_TO_CHAR	ORA_CVT_TO_DATE	ORA_CVT_TO_NUMBER
ORA_CVT_TO_ROWID	ORA_DATE_ROUND	ORA_DATE_TRUNC
ORA_DELETE_ERROR	ORA_DELETE_OBJECT	ORA_DELETE_PHANTOMS
ORA_DELETE_TRANS	ORA_DROP_PASSWORD	*ORA_DROP_SEQUENCE
ORA_DROP_USER	ORA_ENCRYPT_PASSWORD	ORA_GETSYI
ORA_INIT	ORA_LOGIN	ORA_LOGIN2
ORA_LOGIN3	ORA_LOGON	ORA_OUTPUT_DISABLE
ORA_OUTPUT_ENABLE	ORA_OUTPUT_GET_LINE	ORA_OUTPUT_GET_LINES
ORA_OUTPUT_NEW_LINE	ORA_OUTPUT_PUT	ORA_OUTPUT_PUT_LINE
ORA_SET-NLS	ORA_SET_PASSWORD	ORA_TEST_BIT
ORA_TO_CHAR	V\$NLS_SET_FUNC	V\$NLS_UPDATE

Domains

NUMBER	ORA_LONG	ORA_MAXCHAR
ORA_OBJECT_NAME	RDB1LONG	RDB2LONG
RDB4LONG	RDBLONGRAW	

Functions		
CHARTOROWID	*CURRVAL	*NEXTVAL
ORASTATE	ORA_DDL_RDB	ORA_NUMBER
ORA_OWNER	ROUN2	ROWIDTOCHAR
TEST_BIT	TO_CHAR	TO_DATE
TO_NUMBER	TRUN2	USEREN2
USERENV		

Modules		
ORA_APPLINFO	ORA_CONVERT	ORA_LOG
ORA_MISC1	ORA_MISC2	ORA_MISC3
ORA_OUTPUT	*ORA_SEQUENCE	ORA_SESS
ORA_TRANS	ORA_USER	

A.2 Current Session Information

There are several tables that contain information specific to the current session. They are all Oracle Rdb global temporary tables and their contents are unique to each session. (See the *Oracle Rdb SQL Reference Manual* for more information about temporary tables.)

The table V\$NLS_PARAMETERS contains information about the current settings of all the NLS parameters. The PARAMETER column contains the name of the NLS_PARAMETER and the VALUE column contains its current value for the session.

The table V\$VERSION contains the version numbers of all the underlying pieces of software. The single column BANNER contains the the version numbers of the following:

- Oracle Rdb OCI Server
- Oracle Rdb SQL
- CORE

- TNS Forms
- NLSRTL
- Metadata Views
- Oracle Compatible Release

The table V\$SESSION includes information such as USER# (USERID), USERNAME, SCHEMA# (same as USER#), SCHEMANAME (same as USERNAME), OSUSER (same as USERNAME), PROCESS (process id, sent from client), MACHINE (machine name, sent from client), TERMINAL (terminal name, sent from client), PROGRAM (client program name, sent from client).

Table A–2 shows the type of information stored in the ORA_SESSION table.

Table A–2 ORA_SESSION Table

String Value in the INFO_ TYPE Column	Corresponding String Value in the INFO column
ERROR_MSG	/Error message text; inserted by call to ORA_CREATE_ERROR, retrieved by Select Error from...
DEFAULT_SCHEMA	Default schema for the session
INIT_STATEMENTn	An initialization statement stored by the ORA_INIT stored procedure. The variable n is the zero-based ordinal sequence number of the initialization statement.
DB_LINK	Alias name; inserted by call to ORA_CREATE_DBLINK

A.3 Handling 31-Character Object Names

OCI Services for Oracle Rdb supports 31-character object names. However, because it is unclear if all client applications support 31-character names, Oracle recommends that you use a maximum of 30-character object names.

To determine whether or not you have names with more than 30 characters, use the following queries:

Object	Query
Constraint	<pre>SELECT RDB\$CONSTRAINT_NAME FROM RDB\$RELATION_CONSTRAINTS WHERE CHARACTER_LENGTH(TRIM(RDB\$CONSTRAINT_NAME)) > 30;</pre>
Field	<pre>SELECT RDB\$FIELD_NAME FROM RDB\$RELATION_FIELDS WHERE CHARACTER_LENGTH(TRIM(RDB\$FIELD_NAME)) > 30;</pre>

Object	Query
Index	<pre>SELECT RDB\$INDEX_NAME FROM RDB\$INDICES WHERE CHARACTER_LENGTH(TRIM(RDB\$INDEX_NAME)) > 30;</pre>
Module	<pre>SELECT RDB\$MODULE_NAME FROM RDB\$MODULES WHERE CHARACTER_LENGTH(TRIM(RDB\$MODULE_NAME)) > 30;</pre>
Routine	<pre>SELECT RDB\$ROUTINE_NAME FROM RDB\$ROUTINES WHERE CHARACTER_LENGTH(TRIM(RDB\$ROUTINE_NAME)) > 30;</pre>
Table	<pre>SELECT RDB\$RELATION_NAME FROM RDB\$RELATIONS WHERE CHARACTER_LENGTH(TRIM(RDB\$RELATION_NAME)) > 30;</pre>
Trigger	<pre>SELECT RDB\$TRIGGER_NAME FROM RDB\$TRIGGERS WHERE CHARACTER_LENGTH(TRIM(RDB\$TRIGGER_NAME)) > 30;</pre>

In order to support 31-character names, the Oracle data dictionary (metadata tables) provided with OCI Services for Oracle Rdb defines the domain `ORA_OBJECT_NAME` as `VARCHAR(31)` instead of `VARCHAR(30)` data type. Names that are 31 characters long may not be shown correctly when you use Oracle metadata views.

Index

Symbols

@

- command, 6-10, 6-52
- indirect command file, 6-10, 6-52 to ??
 - SQLSRV manage, 6-10
 - SQLSRV_MANAGE, 6-10, 6-52

A

- Add users and passwords, 4-12
- ALTER DISPATCHER command, 6-11
- ALTER SERVER command, 6-16
- ALTER SERVICE command, 6-20
- ALTER SESSION statement, 5-2
 - changing NLS parameters, 5-2
 - data formatting, 4-3
 - date and numeric data formatting, 4-3
 - format, 5-2
 - LOG clause, 5-5
 - logging with LOG BRIEF, 4-4
 - SCHEMA EMULATION clause, 4-7
 - SET ISOLATION LEVEL clause, 5-3
 - SET NLS clause, 5-3
 - SET SCHEMA EMULATION clause, 5-3
 - specifying character sets, 4-17
 - usage environment, 5-2
- Altering
 - dispatcher, 2-11, 6-11 to 6-15
 - server, 2-8, 6-16 to 6-19
 - service, 2-14, 6-20 to 6-27
- Application development
 - common SQL programs, 1-12
- Applications

- attaching to Rdb with universal service, 4-9
- executing multiple ATTACH statements, 4-9
- AR8ISO8859P6 character set, 4-15
- AR8MSWIN1256 character set, 4-16
- Architecture
 - client/server, 1-3
 - server system, 1-2
- ATTACH statement, 4-9
- Authentication of clients, 2-30
- Authorization
 - database and data access, 2-24, 2-32
 - to access services, 2-31
- AUTHORIZE utility, 2-42
- Authorizing unknown users, 2-27, 6-22, 6-42

B

- BIGINT(2) data type, 4-5
- Bugcheck dump files, 7-5

C

- CAST function
 - handling with TO_DATE object, 4-6
 - to match the DATE ANSI format, 4-4
- Changes, 6-11, 6-16, 6-20
- CHAR data types, 4-5
- Character sets
 - default, 4-15
 - defining on the server, 4-14
 - error messages, 7-2
 - example specifying French language and territory, 4-16
 - example specifying Western European, 4-16

- multibyte, 4-15
 - rules and recommendations, 4-17
 - specifying using the ALTER SESSION statement, 4-17
 - US7ASCII, 4-16
- Choosing a service type, 2-24
- CL8ISO8859P5 character set, 4-15
- CL8MSWIN1251 character set, 4-16
- Client applications
 - error messages returned to, 7-15
 - interacting with OCI Services for Oracle rdb, 1-12
 - programming, 1-12
 - querying, 4-8
 - run against either Rdb or Oracle, 1-10
- Client connection states
 - Session reusable services, 3-2
 - transaction reusable database services, 3-3
- Client connections
 - monitoring, 3-2
- Client system
 - error and trace messages, 7-10
- Client systems
 - character set compatibility with server systems, 4-17
 - defining character sets, 4-14, 4-17
 - rules and recommendations for specifying character sets, 4-17
- Clients
 - definition, 1-1
 - identification and authentication, 2-30
 - SHOW CLIENTS command, 6-72 to 6-75
 - showing, 6-72 to 6-75
- Client/server processing, 1-10
- CLOSE command, 6-28
- Closing an output file, 6-28
- Commands
 - management commands, 6-1
- Configuration
 - altering dispatcher, 2-11
 - altering server, 2-8
 - altering service, 2-14
 - copying, 2-10
 - creating dispatcher, 2-10
 - creating server, 2-6
 - creating service, 2-13
 - deleting dispatcher, 2-13
 - deleting server, 2-10
 - deleting service, 2-16
 - restarting dispatcher, 2-11
 - restarting service, 2-14
 - shutting down dispatcher, 2-11
 - shutting down service, 2-14
- Configuration file
 - definition, 1-5
 - SET CONFIGURATION_FILE command, ?? to 6-66
 - showing, 6-85
- Configurations
 - OCI enables diverse combinations, 1-11
- Configuring executor processes, 2-36
- Confirmation prompt
 - providing, 6-67
- Connect string
 - example supplying Rdb database name, 4-18
 - supplying an Rdb database name, 4-9
- CONNECT TO SERVER command, 6-29
- Connect user name, 2-43
 - database access authorization, 2-24, 2-26, 2-40
 - default, 2-27
 - on OpenVMS, 2-40, 2-41, 2-43, 2-44
- Connecting to server, 6-29
- Connection
 - establishing as current, 6-68
 - SET CONNECTION command, 6-68
 - SHOW CONNECTS command, 6-76
- Copying a configuration, 2-10
- CREATE DATABASE LINK statement
 - example, 4-18
- CREATE DISPATCHER command, 6-32
- CREATE SERVER command, 6-36
- CREATE SERVICE command, 6-40
- Creating
 - dispatcher, 2-10, 6-32 to 6-35
 - server, 2-6, 6-36 to 6-39
 - service, 2-13, 6-40 to 6-46
- Cursor
 - management, 4-2
 - SQL semantics, 4-3

D

- Data formatting, 4-3
 - logging information with ALTER SESSION statement, 5-5
- Data types
 - conversion, 4-2
 - decimal representation, 4-5
 - DOUBLE PRECISION, 4-6
 - OCI Services for Oracle Rdb descriptions, 4-4
- Database
 - accessing remote, 4-17
 - accessing with OCI applications, 1-12
 - attaching to Rdb by multiple client applications, 4-9
 - connection using database links, 4-18
 - multischema, 4-7
- Database access authorization, 2-32
 - connect user name, 2-24, 2-26, 2-43
 - on OpenVMS, 2-40, 2-41, 2-43, 2-44
 - service owner, 2-25, 2-26, 2-43
 - on OpenVMS, 2-41, 2-43, 2-44
 - universal service, 2-41
- Database authorization, 2-32
- Database links
 - example definition, 4-18
 - example of an SQL UPDATE statement, 4-19
 - restrictions, 4-19
- Database objects
 - installed for Oracle Data Dictionary emulation, 4-6
- Database service, 2-18, 2-28
 - recommendations, 2-24
 - session reusable, 2-21
 - setting database access authorization, 2-26, 2-43
 - on OpenVMS, 2-41
 - transaction reusable, 2-22
- DATE ANSI format, 4-4
- DATE data type
 - mimicking with DATE VMS date, 4-4
- Date literals
 - formatting, 4-3
- DATE VMS data type
 - handling with TO_DATE object, 4-6
- DATE VMS date, 4-4
- DDL
 - enabling statements in an SQL initialization file, 4-7
 - Oracle Rdb mimics Oracle behavior, 4-2
- Debug flags
 - executor log file, 7-5
- Deciding
 - access to a service, 2-28
 - access to data, 2-29
 - database access authorization, 2-24
 - default connect user name, 2-27
- Decimal representation, 4-5
- DECnet software
 - error codes, 7-15
- Default settings
 - dispatcher, 2-12
 - dispatcher objects, 2-12
 - server, 2-9
 - server objects, 2-9
 - service, 2-15
- Deleting
 - dispatcher, 2-13
 - objects, 6-48 to 6-49
 - server, 2-10, 6-50
 - service, 2-16
- Developer/2000
 - Oracle Forms compatibility, 4-9
- Directing output, 6-70
- Disabling logging, 7-7
- DISCONNECT SERVER command, 6-47
- Disconnecting from the server, 6-47
- Dispatcher
 - altering, 6-11 to 6-15
 - attributes, 2-12
 - creating, 6-32 to 6-35
 - default settings, 2-12
 - definition
 - ALTER DISPATCHER command, 6-11 to 6-15
 - CREATE DISPATCHER command, 6-32 to 6-35
 - deleting, 2-13
 - log file, 7-3
 - restarting, 2-11
 - setting up, 2-17
 - SHOW DISPATCHER command, 6-77 to 6-79
 - SHUTDOWN DISPATCHER command, 6-87
 - START DISPATCHER command, 6-90
 - system management, 2-10
 - transport selection, 2-17

- Display usernames, 4-14
- Displaying command file output, 6-71
- DOUBLE PRECISION data type, 4-6
- DROP command, 6-48
- DROP SERVER command, 6-50
- Dump files
 - See* Bugcheck dump files

E

- Echoing confirmation prompt, 6-67
- EL8ISO8859P7 character set, 4-15
- EL8MSWIN1253 character set, 4-16
- Emulation
 - multischema database, 4-7
 - Oracle Data Dictionary, 4-6
- ENQLM account quota, 2-41
- Environment commands, 6-1, 6-7
- Environment switches, 6-1, 6-7
- Error code file
 - location of DECnet, 7-15
 - location of TCP/IP, 7-15
- Error handling
 - statement parsing, 4-3
- Error log files
 - dispatcher, 7-3
 - executor, 7-5
 - monitor, 7-3
- Error messages
 - logging with ALTER SESSION statement, 5-5
 - reporting problems to Oracle, 7-1
 - returned to OCI client applications, 7-15
 - when specifying character sets, 7-2
- Executor
 - definition, 1-2
 - kill, 6-58
 - log file, 7-5
 - process characteristics, 2-36
 - process configuration, 2-36
- Executor failures and problems, 7-12
- Executor processes
 - configuring mechanisms, 2-36
 - prestarting, 4-9
- EXIT command, 6-51
- Exiting

- the SQLSRV_MANAGE environment, 6-51

- External functions
 - using, 2-42
- EXTRACT command, 6-52

F

- Failure recovery, 7-10
- Figure, 2-33
- Files
 - bugcheck dump files, 7-5
 - dispatcher log file, 7-3
 - executor log file, 7-5
 - input, 6-8
 - monitor log file, 7-3
 - ODBC logging, 7-8
 - SQL initialization, 2-45
 - winsock logging, 7-9
- Format
 - data, 4-3
- Format string
 - providing with TO_CHAR function, 4-6

G

- Getting started with SQL/Services, 2-1
- GRANT USE ON SERVICE command, 6-55
- Granting
 - access to a service, 2-28
 - privilege, 6-55

H

- HELP command, 6-57

I

- Identification of clients, 2-30
- Initialization file
 - initializing the SQL execution environment, 4-7
 - syntax conventions when setting up universal service, 4-10
- Input
 - input switch, 6-8
 - specifying an input file

- SQLSRV_MANAGE, 6-8
- input switch, 6-8
- Installation
 - relations and views created to emulate Oracle Data Dictionary, 8-1
- INTEGER data type, 4-5
- Isolating problems
 - check in log files, 7-5
- Isolation levels
 - setting with ALTER SESSION statement, 5-3
- IW8ISO8859P8 character set, 4-15
- IW8MSWIN1255 character set, 4-16

J

- JA16SJIS character set, 4-15
- JA16VMS character set, 4-15
- JTQUOTA account quota, 2-41

K

- KILL EXECUTOR command, 6-58
- Killing a specified
 - executor, 6-58 to 6-59
- KO16KSC5601 character set, 4-15

L

- Languages
 - supported character sets, 4-15
- LIST OF BYTE VARYING column, 4-5
- LOG clause
 - ALTER SESSION statement, 5-5
- Log files, 7-2
 - dispatcher, 7-3
 - error, 7-3, 7-5
 - executor, 7-5
 - isolating problems, 7-5
 - monitor, 7-3
- Logging
 - disabling, 7-7
 - enabling in an SQL initialization file, 4-8
- Logging for OCI Services for Oracle Rdb, 7-7
- Logical names, 2-46

M

- Management commands, 6-5
- Manager Client GUI utility, 1-5, 1-7
 - system management, 1-7
- Mechanisms
 - security, 2-30
 - used to configure executor processes, 2-36
 - used to set user names, 2-36
- Memory
 - setting size, 2-3
- Message mapping with OCI, 4-2
- Metadata
 - obtaining from Oracle Data Dictionary, 1-12
 - Oracle Data Dictionary emulation, 4-6
 - selected by OCI applications, 8-1
- MODIFY command
 - AUTHORIZE utility, 2-42
- Modify passwords, 4-12
- Monitor
 - definition, 1-4
 - log file, 7-3
 - server activity, 3-1
- Monitoring client connections, 3-2
- MS Windows
 - network error codes, 7-15
- Multibyte character sets, 4-15
- Multischema databases, 4-7
- Multischema emulation, 4-7
 - setting relaxed or strict emulation, 5-3

N

- NATCONN.COM. See Rdb_NATCONN.COM
- Native binary data types, 4-5
- Native data types, 4-2
- NETMBX privilege, 2-41
- Network
 - DECnet error codes, 7-15
 - TCP/IP error codes, 7-15
- Network errors, 7-14
- Network transport problems, 7-11
- NLS parameters
 - changing with ALTER SESSION statement, 5-3
- NLS_LANG logical name, 4-16

NUMBER data type, 4-5
Numeric data
 formatting, 4-3
Numeric data types, 4-5

O

Objects

actions on (commands), 6-5
deleting, 6-48 to 6-49
DROP command, 6-48 to 6-49
DROP SERVER command, 6-50

OCI applications

cursor management, 4-2
message mapping, 4-2
new and existing, 1-12
run against Oracle or Rdb, 1-10
select Oracle metadata, 8-1
using the emulated Oracle Data Dictionary, 4-6

OCI Services for Oracle Rdb

connect string, 4-18
processing features, 1-12, 4-2
translates character sets, 4-17

OCI technology, 1-11

connects client applications to Rdb, 1-11

ODBC logging, 7-8

Offline system management, 1-7

Online system management, 1-7

OPEN command, 6-60

Opening output file, 6-60

OpenVMS operating system

status code definitions, 7-15

Operating system process user name, 2-36

ORA_INIT stored procedure

execute SQL initialization statements, 4-7
storing DDL and other statements, 4-7

ORA-2025 error message, 4-20

ORA-2085 error message, 4-19

Oracle

DATABASE LINK clause, 4-17
error messages, 7-2
SQL UPDATE statement, 4-19

Oracle Call Interface (OCI), 1-11

Oracle Data Dictionary

client queries, 1-12

emulation, 4-6, 8-1
multischema emulation, 4-7
supporting 31-character table names, 8-7
TO_CHAR object, 4-6
TO_DATE object, 4-6
TO_NUMBER object, 4-6
USERENV function, 4-7

Oracle Forms

compatibility with OCI Services for Oracle Rdb, 4-9

Oracle Level1 dialect

DATE VMS data type, 4-6
SQL statement parsing, 4-3

Oracle Network Manager

changing error and trace file names, 7-10
supplying Rdb database name as a connect string, 4-9

Oracle Rdb

current user name, 2-38
session user name, 2-37
system user name, 2-37

Oracle SQL ALTER SESSION statement

See ALTER SESSION statement

Oracle SQL/Services

database service recommendations, 2-24
environment switches, 6-7
initialization file, 4-7

Manager Client GUI

definition, 1-5

objects

actions on (commands), 6-5

Oracle System Identifier (SID)

used for database links, 4-18

Output

file

closing, 6-28
-output switch, 6-9
SET OUTPUT command, 6-70
specifying an output file, 6-9

-output switch, 6-9

P

PAGFILCNT system parameter, 2-41

Parsing

SQL statements, 4-3

- PGFLQUOTA system parameter, 2-41
- Planning a server, 2-2
- PL/SQL statements
 - example usage when DML does not work, 4-20
 - using over database links, 4-19
- Portable data types, 4-2
- Preface, 1-xv
- Prepare a database, 4-11
- Privilege
 - grant, 6-55 to 6-56
 - GRANT USE ON SERVICE
 - command, 6-55 to 6-56
 - needed for system management, 1-6
 - revoke, 6-63 to 6-64
 - REVOKE USE ON SERVICE
 - command, 6-63 to 6-64
- Pro* compilers
 - client applications access databases, 1-12
- Problem Reporting, 7-1
- Problems
 - check in log files, 7-5
- Process failures
 - dispatcher, 7-10
 - executor, 7-11
 - monitor, 7-10
- Property class
 - modify Oracle Forms using RDB_PROPERTY_ CLASS, 4-9

R

- Rdb
 - referencing database tables from Oracle, 4-17
- RDB\$DBHANDLE alias
 - setting up in the ATTACH statement, 4-9
- RDB\$DDTM_XG_INFO, 2-47
- RDB_NATCONN.COM, 4-10
 - DROP, 4-11
 - MODIFY_USER, 4-12
 - NEW_USER, 4-12
 - PREPARE, 4-11
 - REMOVE_USER, 4-14
 - SHOW_USERS, 4-14
 - UPGRADE, 4-11
- Recovery, 7-10

- Remove
 - database, 4-11
 - user name, 4-14
- RESTART SERVER command, 6-61
- Restarting
 - dispatcher, 2-11
 - server, 6-61
 - service, 2-14
- Restricting access to a service, 2-28
- REVOKE USE ON SERVICE command, 6-63
- Revoking a privilege, 6-63

S

- Security
 - client identification and authentication, 2-30
 - database and data access authorization, 2-32
 - how the tiers work, 2-33
 - mechanisms, 2-30
 - on servers, 2-33
 - service access authorization, 2-31
- Selecting service owner user name, 2-40
- Server
 - altering, 6-16 to 6-19
 - architecture, 1-3
 - attributes, 2-9
 - configuration_file, 6-65
 - CONNECT TO SERVER command, 6-29 to 6-31
 - connecting to, 6-29 to 6-31
 - creating, 6-36 to 6-39
 - default settings, 2-9
 - definition
 - ALTER SERVER command, 6-16 to 6-19
 - CREATE SERVER command, 6-36 to 6-39
 - deleting, 2-10, 6-50
 - DISCONNECT SERVER command, 6-47
 - disconnecting from, 6-47
 - management commands, 6-1
 - management utilities, 1-5
 - monitoring activity, 3-1
 - operating system independence, 1-11
 - passing information to with ALTER SESSION
 - statement, 4-3
 - planning, 2-2
 - RESTART SERVER command, 6-61 to 6-62

- restarting, 6-61
- SET CONNECTION command, 6-68 to 6-69
- SHOW SERVER command, 6-80 to 6-81
- showing, 6-80 to 6-81
- SHUTDOWN SERVER command, 6-88
- shutting down, 6-88
- START SERVER command, 6-91 to 6-92
- starting, 6-91 to 6-92
- system, 1-2
- system management, 2-6
- system requirements, 2-2
- ways to manage, 1-5

Server object definitions

- extracting, 6-52

Server systems

- defining character sets, 4-14
- example of defining character sets, 4-16
- rules and recommendations for specifying character sets, 4-17

Server-side solution, 1-11

Service

- access
 - authorization, 2-31, 2-32
 - restricting, 2-28
 - to data, 2-29
- altering, 6-20 to 6-27
- attributes, 2-15
- choosing a service type, 2-24
- creating, 6-40 to 6-46
- database, 2-28
- database access authorization, 2-24
- default connect user name, 2-27
- default settings, 2-15
- definition, 1-2
 - ALTER SERVICE command, 6-20 to 6-27
 - CREATE SERVICE command, 6-40 to 6-46
- deleting, 2-16
- GRANT USE ON SERVICE command, 6-55
- granting access, 2-28
- owner, 2-43
 - database access authorization, 2-25, 2-26
 - on OpenVMS, 2-41, 2-44
- restarting, 2-14
- session reusable database, 2-21, 2-22
- session reusable universal, 2-20, 2-21
- session reuse, 2-19
- SHOW SERVICE command, 6-82 to 6-84
- showing, 6-82 to 6-84
- SHUTDOWN SERVICE command, 6-89
- shutting down, 6-89
- START SERVICE command, 6-93
- starting, 6-93
- system management, 2-13
- transaction reusable database, 2-22, 2-23
- transaction reuse, 2-19
- universal, 2-18, 2-27

Service owner

- database access authorization, 2-25, 2-26, 2-41

Service owner user name

- selecting, 2-40

Session information, 8-5

Session reusable database service, 2-21

Session reusable universal service, 2-20

Session reuse service, 2-19

Sessions

- recommendations for character set specification, 4-17

SET CONFIGURATION_FILE command, 6-65

SET CONFIRM command, 6-67

SET CONNECTION command, 6-68

SET ISOLATION LEVEL

- READ COMMITTED clause, 5-3
- SERIALIZABLE clause, 5-3

SET NLS clause

- ALTER SESSION statement, 5-3

SET OUTPUT command, 6-70

SET SCHEMA EMULATION

- RELAXED clause, 5-3
- STRICT clause, 5-3

SET VERIFY command, 6-71

Setting

- configuration file, ?? to 6-66
- confirm, 6-67
- connection, 6-68 to 6-69
- mechanisms used to set user names, 2-36
- output, 6-70
- SET VERIFY command, 6-71
- shared memory size, 2-3
- SHOW SETTINGS command, 6-85
- showing, 6-85

- verification, 6-71
- verify, 6-71
- Setting up
 - dispatchers, 2-17
 - security, 2-29
- Shared memory size, 2-3
- SHOW CLIENTS command, 6-72
- SHOW commands
 - monitoring server activity, 3-1
- SHOW CONNECTS command, 6-76
- SHOW DISPATCHER command, 6-77
- SHOW SERVER command, 6-80
- SHOW SERVICE command, 6-82
- SHOW SETTINGS command, 6-85
- SHOW VERSION command, 6-86
- Showing
 - clients, 6-72 to 6-75
 - connections, 6-76
 - dispatcher, 6-77 to 6-79
 - server, 6-80 to 6-81
 - service, 6-82 to 6-84
 - settings, 6-85
 - usernames, 4-14
 - version, 6-86
- SHUTDOWN DISPATCHER command, 6-87
- SHUTDOWN SERVER command, 6-88
- SHUTDOWN SERVICE command, 6-89
- Shutting down
 - dispatcher, 2-11, 6-87
 - server, 6-88
 - service, 2-14, 6-89
- SID parameter
 - used for database links, 4-18
- SQL ALTER SESSION statement
 - See* ALTER SESSION statement
- SQL initialization file, 2-45, 4-7
 - syntax conventions, 4-10
- SQL statements
 - cursor semantics, 4-3
 - emulating Oracle semantics, 1-10
 - in an SQL initialization file, 4-7
 - logging transfers between client and server, 5-5
 - modifications made by OCI Services for Oracle
 - Rdb, 4-3
 - parsing, 4-3
 - parsing failures, 4-3
 - run against either Oracle or Rdb databases, 1-12
 - using ALTER SESSION to specify character sets, 4-17
- SQL*Net
 - connects Oracle clients to Rdb servers, 1-10
- SQLNET_BLOB, 2-46
- SQLNET_BUGCHECK_FILE, 2-46
- SQLNET_DEBUG_FLAGS, 2-46
- SQLNET_DOMAIN, 2-46
- SQLNET_RECO_USER, 2-46
- SQLNET_STRUCTURED_DATE_TYPES, 2-47
- SQLNET_TIMESTAMP_DATE_TYPE, 2-47
- SQLNET_VALIDATE_PROGRAM, 2-47
- SQLSRV manage utility
 - @ indirect command file, 6-10
- SQLSRV\$LOG_CONNECTIONS, 2-47
- SQLSRV\$MAX_EXECUTOR_FAILURES, 2-47
- SQLSRV_DISP_DUMP_PATH, 2-46
- SQLSRV_DISP_LOG_PATH, 2-46
- SQLSRV_EXEC_LOG, 2-46
- SQLSRV_MANAGE client
 - definition, 1-4
- SQLSRV_MANAGE utility
 - @ indirect command file, 6-10, 6-52
 - closing output file, 6-28
 - exiting, 6-51
 - getting help, 6-57
 - killing executor, 6-58
 - offline management, 1-8
 - online management, 1-7
 - opening output file, 6-60
 - privileges needed, 1-6
 - running, 1-6
 - specifying input file, 6-8
 - specifying output file, 6-9
 - system management, 1-6
- START DISPATCHER command, 6-90
- START SERVER command, 6-91
- START SERVICE command, 6-93
- Starting
 - dispatcher, 6-90
 - server, 6-91 to 6-92
 - service, 6-93
- static table contents, 8-1

- Storage areas, 2-41
- SYSGEN utility
 - using, 2-41
- System management
 - copying a configuration, 2-10
 - dispatcher, 2-10
 - monitoring server activity, 3-1
 - offline, 1-7
 - online, 1-7
 - Oracle SQL/Services Manager GUI utility, 1-7
 - planning a server, 2-2
 - privileges needed, 1-6
 - server, 2-6
 - server system requirements, 2-2
 - service, 2-13
 - setting shared memory size, 2-3
 - setting up
 - dispatchers, 2-17
 - security, 2-29
 - SQLSRV_MANAGE utility, 1-6
 - ways to manage server, 1-5
- System user name for Oracle Rdb, 2-37

T

- Table names
 - supporting 31 characters, 8-6
- tables
 - installed, 8-1
- TCP/IP software
 - error codes, 7-15
- TH8TISASCII character set, 4-15
- TINYINT column, 4-6
- TMPMBX privilege, 2-41
- TNSNAMES.ORA file, 4-18
 - example, 4-18
- TO_CHAR function
 - Oracle Data Dictionary emulation, 4-6
 - processing, 4-3
 - providing a format string, 4-6
- TO_DATE function
 - Oracle Data Dictionary emulation, 4-6
 - processing, 4-3
 - statement parsing, 4-4
- TO_NUMBER function

- Oracle Data Dictionary emulation, 4-6
- processing, 4-3
- Transaction reusable database service, 2-22
- Transaction reuse service, 2-19
- Transport selection
 - dispatchers, 2-17

U

- Universal service, 2-18, 2-27
 - session reusable, 2-20
 - setting database access authorization, 2-25, 2-41, 2-43
 - specifying the Rdb database, 4-9
 - support, 4-9
 - syntax conventions for SQL initialization files, 4-10
- Unknown users
 - authorizing, 2-27, 6-22, 6-42
- UPDATE statement
 - example usage over database links, 4-19
 - invalid usage and workaround, 4-20
- Upgrade a database, 4-11
- US7ASCII character set, 4-16
- USE privilege
 - granting, 6-55
 - revoking, 6-63
- User authentication and authorization problems, 7-12
- User name
 - connect user name, 2-24, 2-26, 2-40
 - operating system process, 2-36
 - Oracle Rdb current, 2-38
 - Oracle Rdb session, 2-37
 - Oracle Rdb system, 2-37
- USERENV function, 4-7
- Using an SQL initialization file, 2-45
- Using external functions, 2-42
- UTF8 character set, 4-15
- Utilities
 - system management, 1-5

V

- VARCHAR data types, 4-5, 4-7
- Verifying command file input, 6-71
- Version

SHOW VERSION command, 6-86

Views

installed, 8-1

VIRTUALPAGECNT system parameter, 2-41

W

.WE8DEC character set, 4-15

WE8ISO8859P1 character set, 4-15

WE8ISO8859P15 character set, 4-15

WE8MSWIN1252 character set, 4-16

Windows 95

network error codes, 7-15

Windows NT Alpha

network error codes, 7-15

Windows NT X86

network error codes, 7-15

Winsock logging, 7-9

Z

ZHS16CGB2312-80 character set, 4-15

ZHS32GBI1030 character set, 4-16

ZHT16BIG5 character set, 4-15

