Oracle9*i* Enterprise Edition

System Administration Guide

Release 2 (9.2.0.1.0) for OS/390

May 2002

Part No. A97313-01

ORACLE®

Oracle9*i* Enterprise Edition System Administration Guide, Release 2 (9.2.0.1.0) for OS/390

Part No. A97313-01

Primary Author: Enterprise Platforms Division

Contents

Send Us Your Comments

Oracle9*i* Enterprise Edition System Administration Guide, Release 2 (9.2.0.1.0) for OS/390

Part No. A97313-01

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

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

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

infoibm_us@oracle.com

If you would like a reply, please include your name, address, telephone number, and e-mail address.

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

Preface

Intended Audience

Read this guide if you are responsible for performing tasks such as:

- Administering Oracle9*i* Enterprise Edition for OS/390
- Maintaining OS/390 data sets and Oracle system files
- Managing backups, recovery, memory, and storage on OS/390
- Diagnosing and reporting errors
- Issuing OSDI subsystem commands

This guide provides information only for Oracle products and their interactions with OS/390. A thorough understanding of the fundamentals of OS/390 is necessary before attempting to use this software.

Product Name

The complete name for the product described in this book is Oracle9*i* Enterprise Edition for OS/390. To maintain readability and conciseness in this document, the product is also referred to as Oracle9*i* for OS/390.

Related Documents

The documentation set has two parts: OS/390-specific documentation and product-specific documentation. Your site automatically receives both for the Oracle products that you have purchased. The product-specific documentation is intended to assist you in learning how to use a product, and the OS/390-specific documentation will provide assistance regarding special requirements or restrictions for using that product under System/390.

OS/390-Specific Documentation

The OS/390-specific documentation set is used to install, maintain, and use Oracle9*i* for OS/390 products, and consists of:

- *Oracle9i Enterprise Edition Installation Guide for OS/390*
- *Oracle9i Enterprise Edition Messages Guide for OS/390*
- *Oracle9i Enterprise Edition Release Notes for OS/390*
- *Oracle9i Enterprise Edition System Administration Guide for OS/390*
- *Oracle9i Enterprise Edition User's Guide for OS/390*0

Product-Specific Documentation

Product-specific documentation describes how to use the Oracle9*i* products. The information in the product-specific books applies to all operating systems under which the products run.

Conventions

Examples of input and output to the system are shown in a special font:

```
//SYSIN   DSN=oran.orav.INSTJCL(member)
```

All output is shown as it actually appears. For input, the following conventions apply:

Convention	**Meaning**
`italic font`	indicates that a word or phrase of your choice must be substituted for the term in *italic font*, such as the actual member name. For example: `member`
`oran.orav`	is the standard example for high-level and second-level data set name qualifiers. Substitute your system's actual high-level and second-level qualifiers. These qualifiers may appear in lowercase or in UPPERCASE typeface.
<> Angle brackets	indicate that the enclosed arguments are required and at least one of the arguments must be entered. Do not enter the brackets themselves.
[ ] Square brackets	indicate that the enclosed arguments are optional. Do not enter the brackets themselves.
{ } Braces	indicate that one of the enclosed arguments is required. Do not enter the braces themselves.
| Vertical lines	separate choices.
. . . Ellipses	indicate that the preceding item can be repeated. You can enter an arbitrary number of similar items.
Other punctuation	must be entered as shown unless otherwise specified. For example, commas and quotes.

Commands, reserved words, and keywords appear in uppercase in both examples and text. A fileid can appear with both uppercase and lowercase text. When portions of a fileid appear in *italics*, the use of *italic characters* indicates that those portions can vary. Reserved words and keywords must always be entered as is, because they have reserved meanings within Oracle.

Storage Measurements

Storage measurements use the following abbreviations:

- K, for kilobyte, which equals 1,024 bytes
- M, for megabyte, which equals 1,048,576 bytes
- G, for gigabyte, which equals 1,073,741,824 bytes

Documentation Accessibility

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

Accessibility of Code Examples in Documentation

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

Documents Referenced in this Guide

Oracle OS/390 Books

Oracle9i Enterprise Edition Installation Guide for OS/390

Oracle9i Enterprise Edition Messages Guide for OS/390

Oracle9i Enterprise Edition Release Notes for OS/390

Oracle9i Enterprise Edition User's Guide for OS/390

Other Oracle Books

Oracle Advanced Security Administrator's Guide

Oracle9i Application Developer's Guide - Advanced Queuing

Oracle9i Application Developer's Guide - Fundamentals

Oracle9i Application Developer's Guide - Large Objects

Oracle9i Data Guard Concepts and Administration

Oracle9i Database Administrator's Guide

Oracle9i Database Concepts

Oracle9i Database Globalization Support Guide

Oracle9i Database Migration: Release 2 (9.2)

Oracle9i Database Performance Book Set

Oracle9i Database Reference

Oracle9i Database Utilities

Oracle9i Net Services Book Set

Oracle9i Real Application Clusters Book Set

Oracle9i Recovery Manager User's Guide

Oracle9i SQL Reference

Oracle9i User-Managed Backup and Recovery Guide

IBM Books

DFSMS/MVS Access Method Services for ICF

IMS/ESA System Definition Reference

OS/390 Initialization and Tuning Reference

OS/390 MVS Planning: Workload Management

Setting Up a Sysplex

USS Planning Guide

1 Introduction to OSDI Architecture

OSDI (Operating System Dependent Interface) is a platform architecture for Oracle products running on OS/390. This chapter provides a broad overview of OSDI and introduces concepts and terminology used throughout this book.

The following topics are included:

- Overview on page 1-2
- Benefits of OSDI on page 1-2
- OSDI Architecture on page 1-3

Overview

Oracle9*i* for OS/390 is based on OSDI, an execution environment for Oracle products on OS/390, which was introduced with Oracle8*i* for OS/390, release 3 (8.1.7). The Oracle8*i* products on OS/390 were released in two forms: one based on the older MPM and TNS subsystems and another based on OSDI. In Oracle9*i*, MPM and TNS have been desupported, and products on OS/390 are available only with the new OSDI architecture.

OSDI represents a significant change from the old subsystems in terms of how Oracle products interact with the OS/390 operating system. These changes do not generally affect Oracle product behavior and the interfaces used by customer applications, but they do affect the installation, configuration, and administration of these products on OS/390.

If you are already an Oracle for OS/390 customer with existing MPM or TNS subsystems, you will need to read this manual carefully to understand the OSDI differences. Of particular importance is Chapter 18, "Migration and Upgrade Considerations", which covers considerations for moving from the old subsystems to OSDI.

If you are a new Oracle for OS/390 customer, you can proceed directly to installing the software and configuring Oracle9*i* products. Before doing so, we recommend that you read the first three chapters of this manual.

Benefits of OSDI

Principal benefits of OSDI are:

- Virtual storage constraint relief

  Before OSDI, an Oracle database instance on OS/390 ran in a single address space and was, therefore, constrained by the System/390 architectural limit of 2 gigabytes of addressable virtual memory. This limit makes it difficult or impossible to support very large client populations with a single server.

  OSDI, however, allows a database server to utilize multiple OS/390 address spaces, all operating as a single Oracle instance. Up to 255 address spaces can be configured for a single database instance.

- Enhanced client workload management

  Oracle9*i* for OS/390 with OSDI can set the importance or priority of network client requests relative to each other and relative to all other work in the OS/390 system. Database requests from remote (network) clients of an OSDI server are executed using preemptable enclave SRBs (service request blocks), a lightweight OS/390 dispatching mechanism. Network clients are subject to installation-controlled classification in OS/390 Workload Manager (WLM) terms.

- Improved performance and throughput

  OSDI provides new implementations of a number of the fundamental operating system services that are required by the Oracle database and network products. Many of these implementations have shorter CPU instruction paths, reduced serialization, and/or fewer artificial limits than previous facilities.

- Improved reliability, availability, and serviceability (RAS)

  Extensive recovery mechanisms and improved diagnostic facilities in OSDI reduce the likelihood of problems associated with software or hardware failure and increase the probability of diagnosing failures on the first occurrence. Consumption of cross-memory global resources (ASIDs and linkage indexes) that can lead to an IPL is reduced. Strict segregation of Oracle product code from the supporting OSDI infrastructure code improves fix isolation and granularity.

OSDI Architecture

OSDI consists of several well-defined components. The first of these components is a common management layer that is shared by all Oracle products that are implemented under OSDI. Below this, separate components are provided for each Oracle product or type of product. OSDI currently has two product implementation components, one for the Oracle RDBMS and one for Oracle Net.

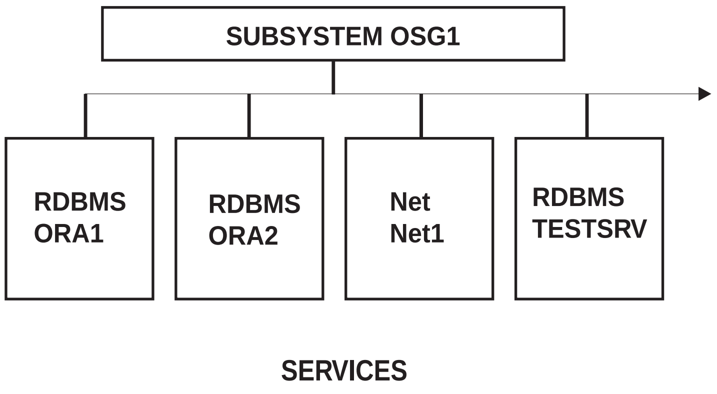
Subsystem

The common management layer of OSDI is implemented as a formal OS/390 subsystem. It is not associated with any single Oracle product or product instance. One OSDI subsystem can support multiple Oracle database instances and multiple Oracle Net services. The term **service group** refers to a collection of database instances and network services that are managed by an OSDI subsystem.

The OSDI subsystem has no associated address space. When the subsystem is initialized (normally during OS/390 IPL), its program code is loaded into system common storage and a few data structures are built, but no address spaces are started. Subsequent interaction with the subsystem is accomplished via OS/390 system operator commands and via programmatic requests issued over the subsystem interface (SSI). No address spaces are created until Oracle products are started.

The OSDI subsystem is what OS/390 calls a dynamic subsystem. This means that it does not have to be initialized at system IPL: it can be initialized at any time using the SETSSI ADD system operator command. Once initialized, the OSDI subsystem is present in the OS/390 system until the next IPL. Because the OSDI subsystem is a dynamic subsystem, it can be deactivated (SETSSI DEACTIVATE) and activated (SETSSI ACTIVATE) at will. When a subsystem is deactivated, no commands can be issued to it, no new Oracle product address spaces can be started, and no new client connections can be made to product instances currently running. Currently-running product instances and any existing client connections are not disturbed.

Figure 1–1 OSDI Architecture



Services

The primary role of the subsystem is to manage **services**. A service is a named, configured instance of an Oracle product. Each Oracle database instance and each network service that you run under OSDI will be a separate service. One subsystem can manage as many services as you desire. It is not necessary to create multiple OSDI subsystems in order to run multiple database instances or to run multiple network services.

Services must be defined to the subsystem. This is done with an OSDI DEFINE command. While OSDI commands can be issued from an OS/390 system console (or similar interfaces such as TSO SDSF or Netview), service definition commands are fairly lengthy and change infrequently. A significant advantage is gained by placing service definition commands in the parameter file that is read during OSDI subsystem initialization. Among the items that are specified when a service is defined are the user-selected service name, the type of Oracle product, and the name of the JCL procedure that is invoked to start an address space for the service. After a service is defined, the definition persists for the life of the IPL. The service cannot be deleted or renamed, but it is possible to change certain parts of the service definition using an OSDI ALTER command. The OSDI database service name is restricted to 6 characters or less.

Defining or altering services manipulates only data structures that belong to the subsystem. These actions do not cause the execution of any Oracle products. In order to run a product that is configured as a service, an OSDI START command must be issued. The OSDI START command is similar to the OS/390 command of the same name: it causes a specified service to begin execution in an OS/390 address space using the JCL procedure specified in the service definition. You must use the OSDI START command (not the OS/390 command) to start services. OSDI-managed services cannot be executed as OS/390 batch jobs or as independent started tasks or STCs.

After a service is started, its behavior and operation depend on the Oracle product and how that product is configured. In general, the subsystem keeps track of the operating state of each service. The primary operating states are inactive, starting, active, and stopping. A service is normally in starting and stopping states only briefly, from the time it is started until initialization is complete (the service becomes active) and from the time termination begins until it actually ends (the service becomes inactive). Only services that are active are available for applications to use. You can display the state of a service using the OSDI DISPLAY command.

Connections

The subsystem is responsible for processing OSDI-specific commands and for managing the definition and operation of services. It also is responsible for managing connections between OS/390 address spaces, specifically between clients and services. In this context, the term **client** differs from the usual Oracle usage of the term. A client is any OS/390 address space that issues requests to an OSDI-managed service via OS/390 cross-memory services. This includes applications that you normally think of as clients, such as an Oracle application or utility that connects to a local OS/390 Oracle database instance. It also includes applications that are less obvious: when the Oracle network service connects an inbound remote client to an OS/390 Oracle database, the network service address space is a client of the database service in OSDI terms. Conversely, when a local OS/390 database user accesses a remote Oracle database via an Oracle database link, the local database address space operates as a client of the network service when the local database address space requests a connection to the remote database.

OSDI calls this process of connecting a client address space to a service address space a **bind**. Bind processing is internal to Oracle product operation and is not directly visible to applications or users. The bind request uses the OS/390 subsystem interface (IEFSSREQ macro), which is why deactivating a subsystem makes its services inaccessible to new client connections, or in other words, to new binds. OSDI has two types of bind requests. One type is used only by Oracle products that have special requirements for managing connections to services, including Oracle Net and the Oracle Access Manager products. The other type of bind is a normal application bind used by customer applications and by Oracle tools and utilities running in TSO, batch, and Unix System Services (USS) address spaces. While bind is not an external mechanism, security considerations that are in bind processing are both visible and important. Security considerations are described in the following section.

Security

OSDI has several integrated security features. They are implemented using the OS/390 RACROUTE interface, which provides program access to the OS/390 System Authorization Facility, or SAF. The discussions and examples in this manual are based on IBM's OS/390 Security Server (RACF) product, but any product that fully supports SAF can be used.

OSDI provides security processing at four points:

1. when an OSDI command is issued
2. when an address space binds to an OSDI-managed service
3. when a user requests a connection to an Oracle RDBMS instance with SYSOPER or SYSDBA privileges
4. when certain types of users log on to an Oracle RDBMS instance for normal application processing

Command and bind authorization checking are part of the subsystem. When an OSDI command is processed, the subsystem performs a check to see if the user or console that is issuing the command is authorized to do so. The check is based on a resource name consisting of the subsystem name and the command verb. Similarly, subsystem bind processing checks to make sure that the user, address space, or task that is issuing the bind is authorized to access the target OSDI service that is specified in the bind request. The bind authorization check distinguishes between the two different types of bind discussed in the previous section. This is because the **managed connection** type of bind carries certain privileges that should not be made available to normal applications.

The SYSOPER/SYSDBA authorization check is performed by the database instance to which the connection is being made and is performed in addition to any bind authorization check made by the subsystem. This check uses a resource name consisting of the OSDI subsystem name, the database service name, and a fixed suffix string (either OPER or DBA).

In the first three of these authorization tests, if the SAF response is an indication that the associated resource is not defined, then the request is allowed. This means that RACF (or comparable) resource definitions are mandatory if command, bind, and Oracle SYSOPER/SYSDBA access are to be subject to SAF-based authorization controls.

In the fourth of these authorization tests, Oracle provides optional SAF-based password authentication at Oracle logon time for certain types of users.

Database Service

Starting an Oracle database service with an OSDI START command creates an OS/390 address space and executes the JCL procedure that is specified in the service definition. In OS/390 terms, the service runs as a system address space (similar to a started task or STC). The JCL procedure contains a single step that executes the OSDI database region control program. A few JCL DD statements are required. The region control program reads an input parameter file that specifies the name of the Oracle RDBMS kernel load module and several OSDI-specific operating parameters. The region control program then loads the Oracle kernel code into the address space and initializes various internal facilities.

If the database service has been defined to use multiple address spaces, and if the operating parameters indicate that additional address spaces are to be started immediately, then those address spaces are started automatically when initialization of the first address space is complete. Each **auxiliary** address space appears as a new STC executing the same JCL procedure as the first address space. Depending on how the service was defined, the new address spaces may have the same jobname as the first, or each may have a distinct jobname containing a 3-digit suffix number. In addition to being started automatically, auxiliary address spaces can be started manually by issuing repeated OSDI START commands for the service. Each

START command (after the first one) starts one more auxiliary address space, up to a maximum number that was specified when the service was defined.

When a database service uses multiple address spaces, client sessions are distributed more or less uniformly across those spaces. Service address space selection is performed by the subsystem during the processing of a client bind request. This is completely transparent to client software. No mechanism exists for a client bind request to designate or control the server address space to which it is assigned. Once a client session is bound to a given address space, the client remains there until it either disconnects from the database or terminates. If multiple database connections are made from a single client address space (in, for example, a multi-tasking application or in one of the Oracle Access Manager products), then those connections are distributed among the service address spaces just as though they had come from different client address spaces.

The auxiliary address spaces of a multi-address-space service do not respond to operator commands. All command interaction with the service is through the first address space, referred to as **AS1**. After auxiliary address spaces are started, they cannot be stopped individually. The auxiliary address spaces terminate when the entire service is terminated. Operator interaction with a running database service uses the OSDI STOP command and the OS/390 STOP (P) and MODIFY (F) system commands, which are processed only by AS1.

OSDI initialization of a database service does not include the generic Oracle database startup. With OSDI, that step (which is required to make the database instance usable by applications) is performed separately. After the service address space, or spaces, are fully initialized, Oracle SQL*Plus (or a comparable mechanism) must issue an Oracle startup request to complete the instance startup processing. Only one startup request is required regardless of the number of address spaces the service is using.

Database service address spaces can persist over multiple Oracle startup and/or shutdown requests. In other words, OSDI does not terminate the database address spaces when an Oracle database shutdown is performed.

Network Service

Like the database, an Oracle Net network service is started with an OSDI START command and executes as a system address space. It uses a different JCL procedure from the database service, executes a different program, and has different JCL DD statement requirements. The network service runs in a single address space only and does not require a separate startup step as is required by the database service. When the network service address space starts, it initializes network protocol tasks and other internal facilities and initiates **listen** operations for inbound clients on designated network endpoints.

The network service must be running to support both inbound (remote) client connections to OS/390 servers as well as outbound connection requests from local OS/390 clients that want to connect to remote Oracle servers. Some important changes exist, however, in the way that inbound connections to OS/390 servers are processed.

OSDI simplifies the networking implementation on OS/390 and makes it behave in a manner more similar to Oracle Net on other platforms. Regardless of which OS/390 database servers are to be accessed, the network service listens for inbound clients on a single endpoint for each protocol. Clients that are connecting via a given protocol specify the target OS/390 server using a SID parameter that is part of the Oracle network address string. The network service does not need to know in advance which servers will be accessed by inbound clients because the network service locates servers using the SID. Nothing needs to be done to a database server to prepare it for access by network clients.

2
Configuring and Initializing the Subsystem

This chapter describes how you create an OS/390 subsystem for OSDI. The following topics are included:

Overview

To run any Oracle products under OSDI, you must first create an OS/390 subsystem. The subsystem provides the internal interfaces that OSDI uses to process operator commands, to manage the execution of database and network services, and to manage connections between address spaces. One OSDI subsystem can support any number of distinct database instances and network services.

Chapter 18, "Migration and Upgrade Considerations", covers issues related to compatibility of OSDI with previous releases of Oracle7 and Oracle8.

Choosing a Subsystem Name and Command Prefix

The subsystem name is a 1-character to 4-character identifier. OS/390 requires that subsystem names begin with an alphabetic or national character, while subsequent characters can be alphanumeric or national. The name must be unique.

The subsystem requires a unique character string prefix to distinguish commands issued to it from system consoles and other system command sources. By default, OSDI will use the subsystem name as the command prefix. You can override this default if you wish, and specify a special character or a different alphanumeric string. Any character string prefix that you choose must not duplicate, or be a leading subset of, the command prefix of any other subsystem. Also, the character string prefix must not match any native OS/390 system command name or abbreviation. Oracle Corporation recommends using the subsystem name for the command prefix.

The Subsystem Parameter File

During initialization, the subsystem opens and reads a sequential data set to obtain **bootstrap** initialization parameters and, optionally, any OSDI commands that are to be issued immediately after the subsystem is initialized. The file is usually a member of a PDS, and is created, viewed, and edited with TSO ISPF or a similar tool. It can have either fixed-length or variable-length records. If fixed-length records are used, then sequence numbers in the rightmost record positions (for example, columns 73-80 of 80-byte records) are ignored.

Oracle Corporation recommends creating a PDS specifically for OSDI parameters and using it for subsystem parameters as well as for parameter files that are used by other components. Ten tracks of primary disk space, three tracks of secondary space, and eight directory blocks should be sufficient for most installations. This data set must be accessible for the subsystem to initialize, so it should not be subject to HSM migration or similar involuntary moves.

Only a single record, called the INIT record, is required in the subsystem initialization file. The INIT record has the following format:

```
INIT (ORASSI[,cmd-prefix][,cmd-class][,bind-class])
```

- INIT

  The word INIT can begin in position 1 or can be preceded by one or more blanks. It must be followed by one or more blanks and then by one to four positional parameters separated by commas and enclosed in a single pair of parentheses. The parentheses are required even if only the first positional parameter is coded. Do not include blanks in or among the positional parameters. If an optional positional parameter is not used, but a following one is, then a comma must be included for the unused parameter.

- First positional parameter

  The first positional parameter is the name of the OSDI subsystem load module that was copied to a system linklist library during installation. This will normally be ORASSI, exactly as shown. This parameter is required.

- Second positional parameter (`cmd-prefix`)

  The second positional parameter is the command prefix for the subsystem. Any characters that are legal in a command prefix (except the comma and the left or right parenthesis) can be included. Do not enclose the prefix in apostrophes or quotes unless those are part of the prefix. The command prefix can be up to 8 characters in length. If you omit this parameter, then the command prefix is assumed to be the subsystem name.

- Third positional parameter (`cmd-class`)

  The third positional parameter is the System Authorization Facility (SAF) resource class name that is to be used when authorizing access to OSDI commands. This parameter should be specified if you have created a specific resource class for command authorization as discussed in the "Pre-Installation Activities" chapter of the *Oracle9i Enterprise Edition Installation Guide for OS/390*. If you omit this parameter, then OSDI uses the FACILITY class for command authorization requests.

- Fourth positional parameter (`bind-class`)

  The fourth positional parameter is the SAF resource class name that is to be used when authorizing client address spaces during bind to a service and when authorizing a local database connection with SYSOPER or SYSDBA privileges. This parameter should be specified if you have created a specific resource class for bind and connect authorization as discussed in the "Pre-Installation Activities" chapter of the *Oracle9i Enterprise Edition Installation Guide for OS/390*. If you omit this parameter, then OSDI uses the FACILITY class for bind and connect authorization requests.

The following INIT record provides an example in which the SAF resource class names have been specified while the command prefix is allowed to default:

```
INIT (ORASSI,,$ORACMD,$ORACONN)
```

OSDI Commands in the Subsystem Parameter File

The INIT record is the only item that is required to be in the subsystem initialization file. As a convenience, however, you can include OSDI commands in the initialization file immediately after the INIT record. Usually, you will have a number of OSDI DEFINE commands there. You might also have OSDI START commands for any services that you always want started as soon as possible during an IPL. Oracle Corporation recommends that you place into the parameter file both the DEFINE SERVICEGROUP command and the DEFINE SERVICE commands for commonly-used services.

The purpose and complete syntax of all OSDI commands is covered in Appendix A, "OSDI Subsystem Command Reference". You should consider the following items regarding commands that are supplied in the subsystem parameter file:

- Do not include the subsystem command prefix. All commands in the subsystem parameter file apply to the subsystem which is being initialized and is reading the file.
- Each command must begin on a new record. The command verb can begin in position 1 or it can be preceded by blanks.
- A command can be continued by including a hyphen (or minus sign) as the last non-blank character on a record (excluding any sequence number). The continuation can begin in position 1 of the next record or it can be preceded by blanks. You cannot continue (split across records) a command parameter that is enclosed in apostrophes. The continuation hyphen is not interpreted as part of the command.

As commands in the parameter file are processed by the subsystem, the command images are not displayed on the console log, but the subsystem response messages appear as if the commands were entered at a console.

Initializing the Subsystem

Before attempting to initialize the subsystem, the required OSDI subsystem modules must reside in the system linklist. Refer to job ORIJC00 in the *Oracle9i Enterprise Edition Installation Guide for OS/390* for information about copying modules to the Linklist Library.

The subsystem can be initialized in either of two ways:

1. At system IPL, based on an entry that you add to the IEFSSN*xx* member of SYS1.PARMLIB
2. At any other time by using the SETSSI ADD system operator command

Oracle Corporation recommends that you add regularly-used subsystems to the IEFSSN*xx* member so that they are initialized correctly and automatically at every IPL. Use the SETSSI ADD system operator command when necessary, such as when you first install OSDI and want to try bringing up a subsystem without having to IPL.

Because OSDI uses OS/390 dynamic subsystem interfaces, the IEFSSNxx entry for an OSDI subsystem must use the newer keyword parameter format, not the old positional format.

Assuming that you have chosen ORSS as your OSDI subsystem name and that the subsystem parameter file is member SSP01 of the data set *ORAN.ORAV*.PARMLIB, an appropriate IEFSSN*xx* entry would be similar to the following:

```
SUBSYS SUBNAME(ORSS) INITRTN(ORASSINI)
       INITPARM('ORAN.ORAV.PARMLIB(SSP01)')
```

In the above example, ORASSINI is the name of the subsystem's initialization routine. This module was copied to a system linklist library during OSDI installation. It must be specified exactly as shown.

The subsystem parameter string (INITPARM keyword) must specify the data set name and, if applicable, the member name of the subsystem parameter file containing the INIT record and optional commands. If no member name is supplied, then this must be a sequential (DSORG=PS) data set.

After updating IEFSSN*xx*, an IPL is necessary to process the added entry. If you do not want to wait for an IPL, or if other circumstances exist in which a subsystem must be created without an IPL, then you can use a SETSSI ADD system operator command equivalent to the following:

```
SETSSI ADD,S=ORSS,I=ORASSINI,P='ORAN.ORAV.PARMLIB(SSP01)'
```

This example uses the minimal keyword abbreviations. The longer forms, SUBNAME (for S), INITRTN (for I), and INITPARM (for P), can be used if desired.

Caution: Oracle Corporation recommends that you use caution when entering the SETSSI ADD command. If you make a mistake in the parameter string (such as misspelling the data set or member name), then the subsystem will fail to initialize, and the subsystem name that you specified will be unusable until the next IPL.

Examples

The following code represents the contents of a subsystem initialization file that includes OSDI commands to define the service group and several services. The file ends with an OSDI SHOW command that will display the definitions in the system log.

```
INIT (ORASSI,ORSS)
DEFINE SERVICEGROUP SSID(ORSS) DESC('Oracle OSDI Subsystem')
DEFINE SERVICE ORAPROD  DESC('Oracle Production DB') -
  TYPE(ORA) PROC(ORADB01) JOBNAME(OPROD*)  MAXAS(8) -
  SID(ORA1) PARM('ORAN.ORAV.PARMLIB(ORAPROD1)')
DEFINE SERVICE ORATEST DESC('Oracle Test DB') TYPE(ORA) -
  PROC(ORADB01) SID(ORAT) -
  PARM('ORAN.ORAV.PARMLIB(ORATEST)')
DEFINE SERVICE ORANET DESC('Oracle Net') -
  TYPE(NET) PROC(ORANET01)  -
  PARM('HPNS PORT(1521)')
SHOW SERVICEGROUP LONG
```

Consider this file to be member SSP01 of *ORAN.ORAV*.PARMLIB as in the earlier examples, and assume that the subsystem will be initialized with a SETSSI ADD command using subsystem name ORSS. The OS/390 system log that results from the SETSSI ADD command would be similar to the following:

```
SETSSI ADD,S=ORSS,I=ORASSINI,P='ORAN.ORAV.PARMLIB(SSP01)'
IEF196I IEF237I 0C9A ALLOCATED TO SYS00049
MIS0012I INITIALIZATION OF ORACLE SUBSYSTEM ORSS COMPLETE 500
MIS0196I Service group ORSS defined 501
```

```
MIS0198I Service ORAPROD  defined 502
MIS0198I Service ORATEST  defined 503
MIS0198I Service ORANET   defined 504
MIS0195I Service group ORSS (Oracle OSDI Subsystem) 505
         Mode=*SYS   , Systems=*ALL
         Service ORANET   Type NET (Oracle Net)
         Service ORATEST  Type ORA (Oracle Test DB)
         Service ORAPROD  Type ORA (Oracle Production DB)
IEF196I IEF285I   ORAN.ORAV.PARMLIB                      KEPT
IEF196I IEF285I   VOL SER NOS= WHODAT.
IEFJ022I SETSSI ADD COMMAND FOR SUBSYSTEM ORSS COMPLETED 508
SUCCESSFULLY
```

At this point, a subsystem has been initialized, and three services have been defined. Configuring and operating a database service and network service are described in Chapter 3, "Configuring a Database Service and Creating a New Database" and Chapter 10, "Oracle Net".

After Initializing the Subsystem

When you have successfully initialized the subsystem, you can proceed to configure and start Oracle products that will run under its control. Before doing so, you may want to enable security mechanisms that the subsystem provides. These security mechanisms control access to subsystem commands, and they control program access to the services that you define in the subsystem. Refer to Chapter 8, "Security Considerations", for information about these and other security features.

3 Configuring a Database Service and Creating a New Database

After you have created an OSDI subsystem, you can configure and initialize one or more Oracle databases to run under that subsystem. This chapter describes how to set up OSDI definitions, JCL procedures, parameter files, and other OS/390-specific items required by an Oracle database instance. The three chapters, Chapter 4, "Defining OS/390 Data Sets for the Oracle Database", Chapter 5, "Operating a Database Service", and Chapter 6, "Database Backup and Recovery", provide additional details on Oracle database files, database service operation, and database backup and recovery. We suggest that you read these chapters before configuring a new database service, and that you review Chapter 8, "Security Considerations", for information about OS/390 security features that affect an Oracle database service.

If you are migrating an existing Oracle for OS/390 database to Oracle9i, you will not be creating a new database as described in this chapter. Refer to Chapter 18, "Migration and Upgrade Considerations", for details on migrating your existing database. If you are new to OSDI, read this chapter to learn how OSDI differs from the MPM subsystem as far as database configuration is concerned.

The following topics are included:

Overview

To create an Oracle database instance under OSDI, you must first define the instance as a service using the OSDI DEFINE SERVICE command. In addition to defining the service, some other items must be set up before the service can be started: a JCL procedure, several parameter files, and possibly security resource definitions.

After you have defined the instance as a service and set up the additional items, you can start the service, which creates one or more OS/390 address spaces based on controls that you have specified. After the address spaces are initialized, you must run Oracle SQL*Plus (or a similar tool) to perform the Oracle database startup function. When the startup is complete, you can use the same tool to issue the CREATE DATABASE SQL statement. This statement causes the Oracle server to create the VSAM linear data sets that comprise a database (if you chose not to pre-allocate them) and to initialize their contents. After the database is created, a series of SQL scripts is executed to create the Oracle server's internal database objects (tables, views, stored PL/SQL procedures, and so forth). After the execution of the scripts is complete, your database is ready to use.

An ISPF panel-driven configuration process is included in this chapter on page 3-27.

Database Service Definition

The OSDI DEFINE SERVICE command is described completely in Appendix A, "OSDI Subsystem Command Reference". Here, we cover DEFINE parameter considerations that are specific to an Oracle database service.

Service Name

The service name for a database can be anything that you want within the content limitations described in Appendix A. By default, OSDI will use the service name as the SID for the service. (The SID is an identifier that end users or application developers must supply to connect an application to a particular database.) The SID can be specified separately, however, and is not required to be the same as the service name.

Note: If you specify a service name that is the same as any existing subsystem name in your system (Oracle database or otherwise), then you must also specify a JOBNAME parameter that is not the same as any existing subsystem. If you do not use unique names, then OSDI starts the service using the service name as the job identifier. When OS/390 processes a start for an address space whose job name or job identifier matches a known subsystem, the job runs under control of the master subsystem instead of under control of JES.

Warning: Running OSDI services under the master subsystem is not supported. This situation must be avoided by making sure that the service runs with a job name or a job identifier that is not the same as any subsystem name.

TYPE

The TYPE parameter for a database service must be specified as ORA.

PROC

This parameter specifies the name of a service JCL procedure that you will place in one of your system procedure libraries. The procedure need not exist when DEFINE SERVICE is issued, but it must be in place before the service is started. The procedure name can be anything that you choose or that the naming standards of your installation require. The requirements for this procedure are discussed in section "Database Region JCL" on page 3-6.

PARM

The PARM for a database service specifies the name of an OS/390 data set containing service initialization parameters. These are OS/390-specific parameters (not the Oracle RDBMS init.ora startup parameters) and are described in the section "Database Region Parameters" on page 3-10. Typically, PARM will specify a member of a PDS (Partitioned Data Set) that is used for various Oracle parameter files. If no member name is included in the PARM string, then the specified data set must be sequential (DSORG=PS).

MAXAS

If you want to exploit the multiple-address-space server features of OSDI, then you should specify the MAXAS parameter on DEFINE SERVICE with a value greater than the default of 1. This sets the maximum number of address spaces for the service, which may be greater than the number started when the service is first brought up. (The number of address spaces to start initially is a database region parameter.) This parameter can be altered with OSDI commands as long as the database service is not active.

JOBNAME

When you run a database service with multiple address spaces, the JOBNAME parameter of DEFINE SERVICE can be used to cause each address space to have a distinct jobname. Although this is not required, it may be desirable if you use OS/390 facilities (such as RMF) that distinguish address spaces by jobname. To do this, specify JOBNAME(*name**), where *name* is a 1-character to 5-character jobname prefix followed by an asterisk, as shown. As each address space is started, OSDI substitutes a 3-digit address space counter for the asterisk (001, 002, and so on) to produce the final jobname. You can also use JOBNAME to cause the service to run with a jobname different from the service name (which is used by default).

As discussed in the "Note" on page 3-4, you must specify a JOBNAME parameter if the service name matches any existing subsystem name in your OS/390 system.

SID

The SID parameter specifies a unique identifier for the service. It is a critical element in the process that is used by Oracle database applications to specify the instance to which they need to connect. (Inbound network clients specify a SID in the Oracle database network address string that must match the SID that is specified in DEFINE SERVICE. Local OS/390 clients connecting via cross-memory specify the SID in any of several ways.) Although the SID can be up to 8 characters long, you may want to specify a SID that is 4 characters or less in order to enable an OS/390-specific feature that local OS/390 clients can use to specify a target database. This feature relies on a dummy JCL DD statement (or TSO allocation) whose DD name begins with "ORA@" and ends with a 1-character to 4-character SID of the target database instance. If you choose a SID longer than 4 characters (or allow it to default to a service name that is longer than 4 characters), this feature is not usable.

Although you can issue the OSDI DEFINE SERVICE command via an OS/390 system console or similar facility, you should put definition commands for services that you use regularly into the OSDI subsystem parameter file, after the DEFINE SERVICEGROUP command. This ensures that the service is always defined correctly and automatically when the subsystem is initialized (normally at system

IPL). In our sample database DEFINE SERVICE command below, the command prefix has been omitted and continuation hyphens have been included as though the command were in the subsystem parameter file:

```
DEFINE SERVICE ORADB01 TYPE(ORA) PROC(ORADB01) -
 DESC('Oracle Test Database') MAXAS(3) -
 JOBNAME(ODB1*) PARM('ORAN.ORAV.PARMLIB(ODB1P)') -
 SID(ODB1)
```

Database Region JCL

Defining a database service requires you to specify a JCL procedure name in a system procedure library. You must create the procedure before you try to start the service, and the procedure must invoke the OSDI database region program with an EXEC statement such as the following:

```
// EXEC PGM=ORARASC,REGION=0M
```

`REGION=0M` is specified to ensure that the server can allocate as much private virtual memory as it needs. Some OS/390 systems may prohibit or alter a REGION parameter such as this, so you might want to check with your systems programmer to make sure that the system will accept your REGION parameter.

Note that no other EXEC statement parameters are needed. The PARM parameter of EXEC is not used by the database region program.

In addition to the EXEC statement, the procedure will need several DD statements, as follows:

ORA$ENV: This DD statement is optional. When used, it specifies a sequential file or PDS member containing environment variable assignment statements. Environment variables are used to supply operating parameters to certain Oracle database product features. Reliance on environment variables and considerations for setting them are discussed in feature-specific chapters of this manual as well as in the *Oracle9i Enterprise Edition User's Guide for OS/390*. The data specified by ORA$ENV is read only at database service startup. Therefore, in order for changes to the data set to take effect, the service must be stopped and started.

Note: The ORACLE_HOME environment variable (referring to the ORACLE_HOME directory name under HFS, specified during installation) is required for OS/390 Unix System Services (USS) components such as Oracle9*i* JVM, Oracle9*i* Text, and the time zone feature.

ORA$FPS: This DD statement specifies a sequential file or PDS member containing OS/390-specific parameters that control data set processing in the Oracle server. These parameters are organized by type of file (such as tablespace, control, online log, and so forth), and they primarily pertain to creation processing when the Oracle server invokes the IDCAMS utility or dynamic allocation to create an OS/390 data set. Considerations and syntax rules for the ORA$FPS parameter file are covered in "Server File Management Parameters" on page 4-8. The ORA$FPS DD is optional. If you omit it, then server file creation operations may fail unless your installation has DF/SMS ACS routines that supply defaults for data set creation parameters. At database service startup, data specified by ORA$FPS is read and checked. Any errors are reported and ignored. Valid entries are loaded as server file management parameters. After database service startup, a new set of server file management parameters can be loaded from the updated ORA$FPS specification by using the REFRESH FPS command on page 5-7.

Note: When this DD statement is omitted, an IEC130I message may appear in the system log during service address space initialization. This is normal.

ORA$LIB: This DD statement specifies a non-authorized load library from which non-executable (data) modules are fetched. The modules contain NLS data objects and messages that are associated with Oracle NLS internationalization features. Normally these modules are installed in the OSDI MESG data set, for example *ORAN.ORAV*.MESG. The ORA$LIB DD statement is optional: if you omit it, then the Oracle server attempts to fetch messages and NLS data objects modules from STEPLIB. Do not concatenate a non-APF-authorized MESG data set to STEPLIB in lieu of specifying ORA$LIB.

Note: When this DD statement is omitted, an IEC130I message may appear in the system log during service address space initialization. This is normal.

ORAPASSW: This DD statement is optional. It specifies a VSAM linear data set that has been initialized with the ORAPWD utility. This file contains encrypted passwords and is used only to authenticate a client who is connecting as SYSDBA or SYSOPER. The use of this file is described in "Controlling Access to Database SYSDBA and SYSOPER Privileges" on page 8-4 and the ORAPWD utility is discussed in Chapter 7, "Oracle9i Utilities".

SNAPCF: This DD statement is optional. When used, it specifies a VSAM linear data set that contains a copy of the database control file. The considerations for this file are discussed in Chapter 6, "Database Backup and Recovery".

SQLBSQ: This DD statement specifies an input file containing the Oracle database "bootstrap" SQL script. It is read only during an Oracle database cold start (CREATE DATABASE SQL statement) and is therefore required only when a cold start is planned. When specified, it usually designates the SQLBSQ member of a partitioned data set dedicated to SQL scripts. This data set was created during Oracle product installation.

SQLNET: This DD statement specifies an input file containing Oracle Net parameters. It is required if the Oracle instance uses any of the following:

- network data encryption
- network activity tracing
- altering of default Oracle Net file names
- outbound database links whose Oracle Net addressing requires access to an Oracle database Names server

Refer to Chapter 10, "Oracle Net", for additional information.

SQLNETLG: This DD statement is the default destination for network error messages. It is required if the Oracle instance uses Oracle Net services, either inbound (remote clients connecting to the instance) or outbound (database links to remote Oracle database instances). SYSOUT or a sequential disk data set can be specified.

STEPLIB: This DD statement must specify the APF-authorized Oracle AUTHLOAD library that was populated during installation. The IBM LE/370 runtime library must be concatenated to it unless your installation has put LE/370 runtime into the system linklist. A typical name for the LE/370 runtime library is SYS1.SCEERUN, but it may have a different name in your system.

SYSPRINT: This DD statement is optional. When used, the Oracle database instance alert log is written to it. The alert log is a sequential text file containing status messages that are related to the operation of the database instance, including startup and shutdown information, log file switches, archive operations, and certain types of error condition. The alert log is also used to log some OS/390-specific events, including IDCAMS utility output associated with database file creation and deletion. Regardless of the number of server address spaces, an Oracle database instance has only one alert log, which is opened by the first server address space (AS1). Alerts that are generated by sessions in other address spaces are routed to AS1.

You can specify a disk data set or a spool file (SYSOUT) for the alert log. If you omit the SYSPRINT DD, then OSDI dynamically allocates a spool file (using SYSOUT=*) for the alert log during service startup and writes a message to the system log identifying the system-generated DDname for the allocation. If you specify a disk data set for SYSPRINT, and if an error occurs while it is being written (including an out-of-space condition), then OSDI closes SYSPRINT, dynamically allocates a spool file, and begins writing to it. The amount of space required for the alert log depends on database activity and on how long the service address space runs.

TNSNAMES: This DD statement specifies an input sequential file or PDS member containing Oracle Net name/address assignments. It is required if the Oracle instance uses database links (connections to other Oracle database instances) whose USING clause specifies an Oracle Net service name rather than an explicit Oracle Net address. If you are using external routines or shared servers, refer to "Step 2: Create and Modify the Tnsnames.ora File" on page 10-15 to add the correct entries.

Sample Database Region JCL Procedure The following is an example of a JCL procedure for a database region:

```
//ORADB01  PROC
//ORACLE   EXEC PGM=ORARASC,REGION=0M
//STEPLIB  DD   DISP=SHR,DSN=ORAN.ORAV.AUTHLOAD
//         DD   DISP=SHR,DSN=SYS1.SCEERUN
//ORA$LIB  DD   DISP=SHR,DSN=ORAN.ORAV.MESG
//ORA$FPS  DD   DISP=SHR,DSN=ORAN.ORAV.PARMLIB(FPS01)
//ORA$ENV  DD   DISP=SHR,DSN=ORAN.ORAV.PARMLIB(ENV01)
//SQLNETLG DD   SYSOUT=*
//TNSNAMES DD   DISP=SHR,DSN=ORAN.ORAV.PARMLIB(TNS01)
//SQLBSQ   DD   DISP=SHR,DSN=ORAN.ORAV.SQL(SQLBSQ)
//ORAPASSW DD   DISP=SHR,DSN=ORAN.ORAV.DB01.ORAPWD
```

Database Region Parameters

OSDI database region parameters are supplied in a data set whose name is specified as the PARM string in the service definition. This will typically be a member of a PDS. Because the data set name is supplied via the service PARM mechanism, no DD statement is coded in the region JCL. The data set is dynamically allocated, opened, and read when the service is started. Changing parameters in the data set has no effect until the service is stopped and restarted.

The OSDI database region parameters consist of a parameter name followed by the parameter value in parentheses. Each parameter has a long descriptive name and a shorter name of eight characters or less. Each record may contain only one parameter. No continuation is allowed. Records beginning with an asterisk (*) are treated as comments and are ignored. Embedded spaces and all characters after the closing parenthesis are ignored.

DSN_PREFIX_DB | ORAPREFD

The DSN_PREFIX_DB parameter supplies a constant string that is associated with the &ORAPREFD system symbol. The &ORAPREFD system symbol can be used to form the high-level (leftmost) qualifier of OS/390 data set names generated by the Oracle server. The format of the DSN_PREFIX_DB parameter is as follows:

```
DSN_PREFIX_DB ( dsn_prefix )
```

where `dsn_prefix` is a valid 1- to 8-character data set name qualifier that conforms to your installation's requirements. In most cases, this will be the qualifier that is used for all Oracle database files associated with this instance. For example:

```
DSN_PREFIX_DB(ORADB01)
```

DSN_PREFIX_DB has no default value. If you omit this parameter, certain situations in which the Oracle server generates "default" filenames will produce errors. Refer to Chapter 4, "Defining OS/390 Data Sets for the Oracle Database", for more information.

INIT_ADR_SPACES | INTADSPC

INIT_ADR_SPACES controls how many auxiliary address spaces are started, as follows:

```
INIT_ADR_SPACES ( number_of_address_spaces )
```

where `number_of_address_spaces` is the number of address spaces to start. The default is 1, which starts only the control address space (AS1). The maximum is the number that was specified for MAXAS on the associated DEFINE SERVICE command for the database service.

INIT_SESSIONS | INITSESS

The INIT_SESSIONS parameter specifies how many sessions to start in an address space, as follows:

```
INIT_SESSIONS ( number_of_sessions )
```

where `number_of_sessions` is the number of sessions to start in an address space. The default is zero. The default of zero should normally be used.

INIT_STACK_SIZE | INTSTKSZ

INIT_STACK_SIZE controls the size of the C stack that is allocated for each session, as follows:

```
INIT_STACK_SIZE ( init_size )
```

where `init_size` determines the initial size of the C stack. This value can be specified as *n* or *n*K. The default is 128K. For more information on INIT_STACK_SIZE, refer to "The User Stack Area in OS/390" on page 16-10.

If the RDBMS Java system will be initialized, and if Java stored procedures will be used, then the value of `init_size` should be at least 256K.

JAVA_STACK_SIZE | JAVASKSZ

JAVA_STACK_SIZE controls the size of the C stack used when executing Java threads as follows:

```
JAVA_STACK_SIZE ( java_thread_size )
```

where `java_thread_size` determines the size of the C stack used for all Java threads except the main thread. The default is 256K. Because the Java thread stacks are not automatically extended as normal C stacks are, this value needs to be large enough to handle all Java thread requirements.

LOGON_AUTH | LGNAUTH

LOGON_AUTH specifies how the Oracle server interacts with a SAF-based external security product when processing Oracle logons for users defined as IDENTIFIED EXTERNALLY. The syntax is:

```
LOGON_AUTH ( auth )
```

where *auth* is:

`NONE`	IDENTIFIED EXTERNALLY not allowed (no interaction)
`SAF`	perform built-in SAF RACROUTE verification
`exitname`	call an installation-supplied logon exit; *`exitname`* is the 1- to 8- character load module name of the exit

If *`exitname`* is specified, it must reside in the system linklist or in an APF-authorized library that is part of the server region STEPLIB concatenation. The default is `NONE`.

Examples:

```
LOGON_AUTH(NONE)
LOGON_AUTH(RACFSMPO)
```

For more information about Oracle logon authorization, refer to Chapter 8, "Security Considerations".

MAX_SESSION_MEM | MAXSMEM

The MAX_SESSION_MEM parameter specifies a hard limit on the amount of virtual memory that a single database session can allocate, as follows:

```
MAX_SESSION_MEM ( session_memory )
```

where *`session_memory`* is the maximum amount of virtual memory that a single database session can allocate. This value can be specified as *n*, *n*K (denoting a multiplier of 1024), or *n*M (denoting a multiplier of 1,048,576). The default is zero (0), which means no session limit is imposed.

This parameter is useful for stopping a "runaway" session that is allocating excessive amounts of memory due, perhaps, to problems with application design. This pertains only to session-private C stack and "heap" memory allocated during Oracle server processing. It does not include Oracle SGA memory used by a session nor internal memory allocations done by the implementation.

Care must be taken in choosing a limit, particularly where certain database administration operations might be affected. The "catalog build" step of new database creation requires as much as 64M of session memory and may fail if this parameter is set to a lower value. Omit this parameter or set it to a higher value during new database creation; you can change it to a lower value afterward if desired. In the current product release a normal database STARTUP requires up to 16M of session memory, so do not set this parameter to a value less than 16M.

MAX_SESSIONS | MAXSESS

The MAX_SESSIONS parameter limits the number of sessions that can be scheduled in an address space, as follows:

```
MAX_SESSIONS ( number_of_sessions )
```

where `number_of_sessions` is the maximum number of sessions per address space. This value can be specified as *n* or *n*K. The default is 1024. The number of sessions that can be supported in an address space depends on the complexity of the work. Limiting the number of sessions per address space reduces the chances of session failure due to exhaustion of virtual storage. Refer to "Database Server Address Space Configuration" on page 16-6 for more information.

REGION_MEM_RESERVE | REGMRES

The REGION_MEM_RESERVE parameter specifies the amount of private area memory in the server address space to be "reserved" for implementation and OS/390 use (not available for the Oracle SGA and Oracle session-private purposes), as follows:

```
REGION_MEM_RESERVE ( region_memory )
```

where `region_memory` is the amount of private area memory reserved. This value can be specified as *n*, *n*K (denoting a multiplier of 1024), or *n*M (denoting a multiplier of 1,048,576).

During initialization, each server address space calculates the total available private area memory and subtracts the reserve amount from it. The result is the aggregate limit for the Oracle SGA and for all session memory requests in that address space.

The default is zero (0), which means that no aggregate limit applies. In this case, it is possible for Oracle SGA and session memory requests to exhaust the available private area of the address space, leading to unpredictable failures.

Thus, the reserve amount must be sufficient to accommodate internal implementation memory requrements as well as memory required by OS/390 services used by Oracle, particularly Local System Queue Area (LSQA) memory, which is used by all database I/O operations. Because it is difficult to predict this amount for any given workload, the best strategy is to specify a relatively large reserve amount, such as 50M or more. This has the effect of reducing slightly the number of sessions that can be accommodated in a server address space. However, additional address spaces can be started, if necessary.

SERVER_LOADMOD | SRVRLMOD

SERVER_LOADMOD specifies the name of the service load module, as follows:

```
SERVER_LOADMOD ( loadmod )
```

where *loadmod* is the name of the load module to load. For the Oracle RDBMS, this is usually ORACLE. This parameter is required.

SMF_STAT_RECNO | SMFSTRCN

SMF_STAT_RECNO specifies the SMF record number to use, as follows:

```
SMF_STAT_RECNO ( record_number )
```

where *record_number* is the number of the desired record of Oracle SMF statistics. The default is zero (0). Otherwise, the value must be specified between 128 and 255 for this parameter. Example:

```
SMF_STAT_RECNO(204)
```

The collection and writing of Oracle SMF statistics records is controlled by this single parameter in the OSDI service parameter file. A zero (0) for this parameter indicates that no SMF statistics record is to be written. The SMF record number that is chosen must not be the same as the number that is used by any other OS/390 software.

If this parameter is not specified, or if zero is specified, then no SMF statistics collection or recording is done. This saves some CPU overhead and saves the overhead of the SMF write itself (which is mostly asynchronous work done by the SMF address space, the in-line overhead is mainly just moving data into SMF buffers). For more information about SMF, refer to Chapter 9, "Oracle SMF Data".

TRACE_DSNAME | TDSN

TRACE_DSNAME specifies the destination for Oracle RDBMS trace files. This includes normal traces requested by setting the session SQL_TRACE option to TRUE, as well as diagnostic traces generated automatically in certain error situations. The format of the parameter is as follows:

```
TRACE_DSNAME ( filespec )
```

where `filespec` is either a SYSOUT specification (including class, form, and JES destination) or a data set name.

A SYSOUT specification is of the form:

```
//SYSOUT:class,form,dest
```

as described in "Server File Name Syntax" on page 4-6. When this is used, trace files are dynamically allocated SYSOUT data sets. In a multi-address space service, the trace file for a given database session is allocated in the address space that hosts the session. Thus, SYSOUT trace files can appear in all server address spaces. For example, traces written to SYSOUT class X, form AA01, would be written as:

```
TRACE_DSNAME(//SYSOUT:X,AA01)
```

As an alternative to a SYSOUT specification, you can specify a data set name. Because each trace file created as a data set must have a unique data set name, the supplied value must include system symbols that guarantee uniqueness. Refer to Appendix C, "Oracle9i for OS/390 System Symbols" for more information.

To guarantee uniqueness, use some combination of the session identifier (&ORASESST) system symbol, date (&LYYMMDD), and time (&LHHMMSS). Also use high-level qualifier(s) that are appropriate for your installation. This will avoid the possibility of duplicating trace data set names generated in other Oracle instances you run. All components of the string must resolve to produce a name that is valid for an OS/390 sequential data set. For example:

```
TRACE_DSNAME(ORA3A.TRACE.D&LYYMMDD..T&LHHMMSS..&ORASESST)
```

The allocation parameters for Oracle trace data sets are obtained from the DBTR file group of the server file management parameters, discussed in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database".

If this parameter is omitted or fails to produce a valid, unique data set name, all Oracle trace files are written to the default SYSOUT class associated with the server region.

Oracle Initialization Parameter Considerations

As an Oracle instance starts up, Oracle initialization parameters are read as follows:

- If PFILE is specified in the STARTUP statement, initialization parameters are read from the specified PFILE. The PFILE can refer to a sequential data set or member of a PDS. It cannot refer to a file in the HFS.

- If PFILE is not specified in the STARTUP statement the server will read from a server parameter file (SPFILE). If no server parameter file exists, the server will read from /DD/INITORA, which can refer to a sequential data set or member of a PDS. For more information on the SPFILE parameter, refer to the description on page 3-19.

Considerations for most of the initialization parameters in this file are the same regardless of the operating system on which the associated Oracle database instance runs. A few of them have OS/390-specific considerations as discussed here. Use this section together with other Oracle9*i* documentation when choosing initialization parameters for an Oracle server on OS/390.

BACKGROUND_CORE_DUMP, BACKGROUND_DUMP_DEST

These parameters are ignored on an OS/390 Oracle server.

CONTROL_FILES

The CONTROL_FILES parameter specifies the name, or names, of one or more database control files that are specified using the file name syntax discussed in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database". When you are first creating a database, if you choose to let the Oracle server allocate your control files (instead of pre-allocating them yourself), this parameter will specify the VSAM linear data set names that do not yet exist but will be created during processing of the CREATE DATABASE command.

The following is a sample CONTROL_FILES parameter for OS/390 using the full file name syntax:

```
CONTROL_FILES = "//'ORAPROD.ORADB1.CTL1'", "//'ORAPROD.ORADB1.CTL2'"
```

DB_BLOCK_SIZE

The DB_BLOCK_SIZE parameter is used as the default logical database blocksize for all tablespaces that do not request a different logical blocksize with the BLOCKSIZE option of CREATE TABLESPACE. DB_BLOCK_SIZE can be specified as 4096 (4K), 8192 (8K), 16384 (16K), or 32768 (32K).

The default for this parameter on OS/390 is 4096. Regardless of what logical blocksize you use, Oracle database files on OS/390 always have a physical blocksize of 4096 bytes.

DB_CREATE_FILE_DEST
DB_CREATE_ONLINE_LOG_DEST_n

These parameters are associated with Oracle Managed Files (OMF). OMF simplifies database administration by making the Oracle server responsible for naming, creating, and deleting the VSAM linear data sets comprising a database. Refer to "Oracle Database Files" on page 4-2 and the *Oracle9i Database Administrator's Guide* for more information on OMF.

On OS/390, these parameters supply character strings that are used as the left-hand portion of the data set names generated by the server when an OMF file is created. Valid OS/390 and Oracle-specific system symbols can be included. The maximum length permitted after any system symbols are resolved is 23 characters for DB_CREATE_FILE_DEST and 29 characters for DB_CREATE_ONLINE_LOG_DEST_n. To be usable for OS/390 data set name generation these strings must end with a period after any symbol substitution has been done. The following are some examples:

```
DB_CREATE_FILE_DEST = "ORACLE.ORADB01."
DB_CREATE_ONLINE_LOG_DEST_1 = "&ORAPREFD..&ORASRVN..M1."
DB_CREATE_ONLINE_LOG_DEST_2 = "&ORAPREFD..&ORASRVN..M2."
```

DB_FILE_NAME_CONVERT
LOG_FILE_NAME_CONVERT

These parameters are used in conjunction with the standby database availability feature and in certain point-in-time recovery situations. They cause the server to convert database and log file names that are read from the control file by replacing a portion of the original file name with another value.

To use the standby feature and the DB_FILE_NAME_CONVERT parameters, your database data file data set names must share a common naming convention. The easiest way to meet this requirement is to use the same high-level qualifier (or qualifiers) for all database data file names. The same logic applies to the LOG_FILE_NAME_CONVERT for the redo log files.

For example, if all of your primary database data files have data set names beginning with "ORA5.DBPRIM." and you choose "ORA5.DBSTNDBY." as the prefix for database files in your standby database, then to effect the name conversion for the standby database, you would specify:

```
DB_FILE_NAME_CONVERT=(ORA5.DBPRIM,ORA5.DBSTNDBY)
```

Information on the standby database feature can be found in *Oracle9i Data Guard Concepts and Administration*.

LOCK_SGA

The LOCK_SGA parameter is ignored on OS/390. Buffers in the Oracle SGA (System Global Area) are pagefixed during I/O operations, only; otherwise, the SGA on OS/390 is pageable.

LOG_ARCHIVE_

The parameters whose names begin with "LOG_ARCHIVE_" control Oracle database processing of filled database log files when the database runs in ARCHIVELOG mode. These parameters changed in Oracle 8.1 to allow more than two copies of archived logs and to support transmitting logs to a remote standby database. The old parameter forms are still supported in Oracle 9.0, but the new forms must be used to allow for more than two log destinations or to use remote standby logging.

With an OS/390 server, LOG_ARCHIVE_DEST and LOG_ARCHIVE_DUPLEX_DEST (the old parameters) or the LOCATION component of each LOG_ARCHIVE_DEST_n parameter are used to specify only the left-hand portion of a full OS/390 data set name. This will be the high-level data set name qualifier, or qualifiers, for all logs archived to this destination. The remainder of the data set name is specified using LOG_ARCHIVE_FORMAT, which should contain the log thread and sequence number substitution indicators (%T and %S) to ensure that each archived log data set name is unique. The destination string must end with a period, or the format string must begin with one in order to form a proper OS/390 data set name. The following example uses the newer parameters:

```
LOG_ARCHIVE_FORMAT = "T%T.S%S"
LOG_ARCHIVE_MIN_SUCCEED_DEST=1
LOG_ARCHIVE_DEST_1='LOCATION=ORA5.ARCHLOG1. MANDATORY REOPEN=5'
LOG_ARCHIVE_DEST_2='LOCATION=ORA5.ARCHLOG2. MANDATORY REOPEN=5'
LOG_ARCHIVE_DEST_STATE_1=enable
LOG_ARCHIVE_DEST_STATE_2=enable
```

With these example settings, archived logs will have data set names of the form ORA5.ARCHLOG1.T*nnn*.S*nnn* and ORA5.ARCHLOG2.T*nnn*.S*nnn* with the *nnn* parts replaced by log thread and sequence numbers.

MAX_DUMP_FILE_SIZE

This parameter is ignored on an OS/390 Oracle server.

ORACLE_TRACE_

The "otrace" facility is not implemented on OS/390. All parameters whose names begin with "ORACLE_TRACE_" are ignored on an OS/390 Oracle server.

SHADOW_CORE_DUMP

This parameter is ignored on an OS/390 Oracle server.

STANDBY_ARCHIVE_DEST

This parameter is used in conjunction with the standby database availability feature to perform database recovery. It is used along with the LOG_ARCHIVE_FORMAT parameter to generate the fully qualified standby database log filenames, which are then stored in the standby database control file. For more information on the standby database availability feature, refer to *Oracle9i Data Guard Concepts and Administration*.

With an OS/390 server, the STANDBY_ARCHIVE_DEST parameter is used to specify only the left-hand portion of a full OS/390 data set name. This will be the high-level data set name qualifier, or qualifiers, for all logs archived to this destination. The remainder of the data set name is specified using LOG_ARCHIVE_FORMAT, which should contain the log thread and sequence number substitution indicators (%T and %S) to ensure that each archived log data set name is unique. The destination string must end with a period, or the format string must begin with one in order to form a proper OS/390 data set name. The following is an example:

```
LOG_ARCHIVE_FORMAT = "T%T.S%S"
STANDBY_ARCHIVE_DEST='ORA5.ARCHLOG.'
```

SPFILE

The SPFILE (server parameter file) parameter refers to the VSAM linear data set managed by the instance. The default name of the SPFILE data set is the following concatenation: &ORAPREFD..&ORASRVN..SPFILE.ORA.

When creating the server parameter file, the PFILE parameter in the CREATE SPFILE statement must be a file in the HFS; this PFILE can be created with a text editor, or by copying your initialization parameters to the HFS using TSO OPUT or an equivalent method.

USER_DUMP_DEST

This parameter is ignored on an OS/390 Oracle server.

Database Logical Blocksize

The term **logical** blocksize means that the Oracle server reads and writes the files in chunks of the given size. The files themselves continue to be VSAM linear data set clusters with a 4K CI size and 4K physical block size. Logical blocksizes greater than 4K are required for certain database structures (such as indexes with very large keys) and they may provide I/O performance benefits in certain situations.

There are two ways to use a blocksize other than 4K. One is to specify a different blocksize in your init.ora (DB_BLOCK_SIZE parameter) when doing a coldstart. It must be one of the four supported values expressed as a full integer, as in the following:

```
DB_BLOCK_SIZE=16384
```

The specified blocksize is used for the control file(s) and as the default for all tablespace data files. (Log files are always written in 4K units, regardless of the value of DB_BLOCK_SIZE.) Note that DB_BLOCK_SIZE cannot be changed after CREATE DATABASE.

You also can specify logical blocksize at the tablespace level in the CREATE TABLESPACE statement. This is more flexible as it allows blocksize tailoring by application. Using this feature requires changes to the init.ora parameters that govern the database buffer portions of the SGA.

First, the old DB_BLOCK_BUFFERS parameter, which specifies the number of default-sized (DB_BLOCK_SIZE) SGA buffers, must be changed to the new DB_CACHE_SIZE parameter. The new parameter has the same effect as the old one but it is expressed as an amount of memory (with a K, M, or G multiplier if desired) instead of as a number of buffers. To convert a DB_BLOCK_BUFFERS parameter to the equivalent DB_CACHE_SIZE, multiply DB_BLOCK_BUFFERS by the logical blocksize. For example, if DB_BLOCK_SIZE is 4096 and DB_BLOCK_BUFFERS is 5000, then you would specify the following:

```
DB_CACHE_SIZE=20000K
```

To enable the use of tablespace logical blocksizes other than your DB_BLOCK_SIZE, you must specify a separate buffer cache amount for each size to be used. The init.ora parameters that do this are named DB_nK_CACHE_SIZE where *n* is 4, 8, 16, or 32 for the respective logical blocksize. These parameters are used only for blocksizes that differ from DB_BLOCK_SIZE. Specifying one of these when you have no tablespaces that use the corresponding blocksize does no harm, but the associated SGA space sits idle.

Example 1 This example shows part of an init.ora file for a database whose default blocksize is 8K and which provides additional buffer cache areas for tablespaces with a 4K and 32K logical blocksize. A total of 84M of buffer memory is divided among the three blocksizes.

```
DB_BLOCK_SIZE = 8192
DB_CACHE_SIZE = 48M
DB_4K_CACHE_SIZE = 12M
DB_32K_CACHE_SIZE = 24M
```

Example 2 This example is a SQL statement which creates a tablespace with a logical blocksize of 32K. An AUTOEXTEND clause is included to enable the tablespace to be enlarged dynamically.

```
CREATE TABLESPACE TS32K DATAFILE 'ORACLE.WFM109.TS32K.DBF1'
  SIZE 60M AUTOEXTEND ON NEXT 8M MAXSIZE 192M BLOCKSIZE 32K;
```

Pre-Allocating Database Files

You can allow the Oracle server to create the database VSAM clusters during CREATE DATABASE processing, or you can pre-allocate them yourself using OS/390 IDCAMS. If you choose to use OS/390 IDCAMS, now is the time to do it. Considerations for defining database files are covered in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database". If you are going to let the server create any files, then be sure to provide creation parameters for the associated file types via the server ORA$FPS DD statement as described in Chapter 4.

Configuring OS/390 Security

If you plan to use any of the SAF-based security features discussed in Chapter 8, "Security Considerations", you may want to configure them now. OSDI subsystem command security affects the service START command that you will issue to start the database service address spaces. Both OSDI bind security and protection of the database SYSDBA or SYSOPER privilege affect the connection that you will make to startup Oracle and create the database.

Even if you do not utilize any of these features, you must ensure that the database service address space is authorized to access resources (such as data sets) that might be protected by default in your system. Refer to Chapter 8 for information on database service interaction with OS/390 security.

Configuring for Shared Servers

Shared servers on Oracle9*i* for OS/390 are supported by the generic listener running under OS/390 Unix System Services (USS). For information on configuring the generic listener for shared servers, refer to Chapter 10, "Oracle Net". For more information about shared servers, refer to the *Oracle9i Net Services Administrator's Guid*e.

Creating the Database

When the OSDI service has been defined, all required JCL procedures and parameter files have been prepared, and any database files that you want pre-allocated have been defined, then you are ready to create the Oracle database.

First, start the OSDI service with an OSDI START command. This is described in Chapter 5, "Operating a Database Service", and in Appendix A, "OSDI Subsystem Command Reference". Check the OS/390 system log to make sure that the service starts successfully, as indicated by message MIR0002I. If problems occur, the service address space(s) will terminate. In this case, correct the problems and issue another OSDI START command.

After the database service is started, you can use SQL*Plus or a similar tool to perform Oracle startup and database creation. OS/390 considerations for running SQL*Plus are covered in the *Oracle9i Enterprise Edition User's Guide for OS/390*. SQL*Plus can be run as a batch job or as a TSO command.

Before issuing STARTUP or CREATE DATABASE, the tool must connect to the running database service. The form of CONNECT command that is used when starting the database is special and does not specify an Oracle userid:

```
CONNECT / AS SYSDBA
```

This command connects the tool as Oracle userid SYS. If you configured SAF-based OSDI bind authorization as discussed in Chapter 8, "Security Considerations", the OS/390 userid associated with the TSO session or batch job must be authorized for the UBIND resource associated with the database service. If you configured SAF-based protection of the SYSDBA and SYSOPER privileges as discussed in Chapter 8, "Security Considerations", the userid must also be authorized for the DBA or OPER resource associated with the database service.

When SQL*Plus processes the CONNECT, it must determine the target database service. In OS/390, you can specify the target service in several ways, all based on the SID that is specified in DEFINE SERVICE. These methods of specification are described in detail in the *Oracle9i Enterprise Edition User's Guide for OS/390*. In our examples, we have used the dummy ORA@sid DD statement to specify the target service.

If CONNECT fails, the subsequent STARTUP and CREATE statements will fail as well. This does not affect the running service. You can simply correct the CONNECT problem and try again.

After CONNECT, you must issue STARTUP with the NOMOUNT option to cause the Oracle instance to initialize without attempting to open any database files. STARTUP also specifies your init.ora file via the PFILE= keyword. The Oracle tools on OS/390 use different file notation for data sets than the server uses. In our example, the init.ora file has been specified as a DD name. The file is opened and read by the tool, not the target server, the DD statement is therefore included in the tool JCL.

If STARTUP fails (because of an error in your init.ora file, for example), the subsequent CREATE will fail as well. In addition to any messages displayed by the tool, messages might be in the instance alert log (usually, though not necessarily, a SYSOUT file). In general, you can correct errors and retry the whole sequence without stopping and restarting the service.

After STARTUP is successful, you can issue CREATE DATABASE. This is a SQL statement and is therefore documented in *Oracle9i SQL Reference*. (CONNECT and STARTUP are commands, not SQL statements, and are described in the SQL*Plus tool documentation.) There are few OS/390-specific considerations for CREATE DATABASE. The syntax conventions for file names that are specified to the server are covered in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database".

If you pre-allocated any of the database files, be sure to specify REUSE in the appropriate clauses. For files that you want the server to create, omit REUSE and, except for the control files, specify SIZE. Oracle automatically calculates SIZE for the control files based on other parameters in CREATE DATABASE. With Oracle9*i*, you can take advantage of the Oracle Managed Files feature and allow the server to generate names for data sets created during CREATE DATABASE. Pay careful attention to the CREATE DATABASE parameters whose names begin with "MAX", because these parameters specify limits that cannot be changed later without recreating the control file.

As CREATE DATABASE is processing in the server, it reads the Oracle bootstrap SQL file, usually referred to as sql.bsq. On OS/390, this file must be specified via a SQLBSQ DD statement in the service JCL procedure as described in "Database Region JCL" on page 3-6. This file is read only during processing of a CREATE DATABASE, and the DD statement can therefore be removed from the procedure if desired. It does no harm to leave it in, however.

After CREATE DATABASE, you must run Oracle-supplied SQL scripts to build the Oracle dictionary, stored PL/SQL procedures, and related structures. Although this can be done immediately after CREATE DATABASE (in the same tool session), we chose to run it separately in our examples. CREATE DATABASE is therefore the last statement in this part of the example.

Our example of OS/390 database creation is presented as a batch job that uses SQL*Plus. We have pre-allocated four log files and a single file for the SYSTEM tablespace, but we are going to let the server create the control files (whose names are specified in the init.ora parameter file, which is supplied via the INITORA DD in this job). The SID of the target instance is ORA1. The /NOLOG in the SQL*Plus PARM prevents SQL*Plus prompting for an Oracle userid and password.

```
//ORACRDB  JOB 1,'Create Database'
//PLUS     EXEC PGM=SQLPLUS,PARM='/NOLOG',REGION=4M
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//ORA@ORA1 DD DUMMY
//INITORA  DD DISP=SHR,DSN=ORAN.ORAV.PARMLIB(INITORA1)
//SYSIN    DD *
CONNECT / AS SYSDBA
STARTUP NOMOUNT PFILE=/DD/INITORA
CREATE DATABASE ORADB1 ARCHIVELOG
 MAXLOGFILES 16 MAXLOGMEMBERS 2 MAXDATAFILES 1024
 LOGFILE 'ORAPROD.ORADB1.LOG1' REUSE,
         'ORAPROD.ORADB1.LOG2' REUSE,
         'ORAPROD.ORADB1.LOG3' REUSE,
         'ORAPROD.ORADB1.LOG4' REUSE
 DATAFILE 'ORAPROD.ORADB1.SYSTEM.DBF1' REUSE;
EXIT
/*
```

The job may run for a while depending on the number and sizes of the files that are specified. Oracle formats all of the primary space for all control, log, and data files.

Populating the SYSTEM Tablespace

After the CREATE DATABASE completes successfully, your database is mounted and open. Before you can create application tables, users, and so forth, you must create the Oracle dictionary tables, stored procedures, and other internal structures. SQL scripts for this purpose are placed in a PDS during Oracle installation. As with database creation, you can use SQL*Plus to process these scripts against your new database.

For a new database, members CATALOG and CATPROC from the SQL PDS must be run, in that order. Both can be done in a single tool execution as in our example. Because these scripts contain embedded references to other scripts that are members of the same PDS, you must use FNA to control file name processing in the tool. (FNA is explained in the *Oracle9i Enterprise Edition User's Guide for OS/390*.) As with our previous example, the target database service is identified by an ORA@sid DD statement. We are using the same CONNECT statement as the create job.

```
//ORACATLG JOB 1,'Build Catalog'
//PLUS     EXEC PGM=SQLPLUS,PARM='/NOLOG',REGION=0M
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//ORA$FNA  DD *
  FSA( FTYPE(SQL) FNAME('/DD/LIB(+)') )
  FSA( FTYPE(PLB) FNAME('/DD/LIB(+)') )
/*
//LIB      DD DISP=SHR,DSN=ORAN.ORAV.SQL
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//ORA@ORA1 DD DUMMY
//SYSIN    DD *
CONNECT / AS SYSDBA
@CATALOG
@CATPROC
EXIT
/*
```

The "@" symbols that are used in SYSIN in the example above are SQL*Plus shorthand notation for reading an alternate input file of commands or SQL statements. Following each "@" is a member name in the SQL PDS. The FNA controls (ORA$FNA DD statement) are used to notify the tool that the names following the "@" are members in the PDS that is identified by the LIB DD statement.

These two scripts are large and will therefore run for quite a while. Because the scripts create and load data into tables, log file data is generated as they execute. If you specified ARCHIVELOG in your CREATE DATABASE statement (as in our example), logs may fill and require archiving while CATALOG and CATPROC run. You may want to avoid this complication because you are not likely to attempt a recovery of a brand new database: in the event of problems, both CATALOG and CATPROC can simply be rerun. To avoid log archiving during catalog creation, specify NOARCHIVELOG in your CREATE DATABASE, and then use ALTER DATABASE to switch to ARCHIVELOG mode later.

Be sure to check the output of the catalog build job carefully. Because the scripts are designed to be rerunnable, they contain DROP statements that produce errors the first time they are run. These errors are normal. Other errors must be investigated and resolved to complete initialization of the database.

Note: The database session which executes the catalog build requires up to 64M of session-private memory. If you have limited session memory to less than 64M with the database region MAX_SESSION_MEM parameter, catalog build may fail with an ORA-04030 error, an LE/370 U4088 ABEND, or other errors.

Depending on other Oracle products or features that you may use, you may need to run additional scripts against your new database to enable those products or features. Refer to the product-specific documentation for more details.

After CATALOG and CATPROC have run, your database is ready for use. If you created your database with NOARCHIVELOG to avoid archiving logs during catalog build, then return it to ARCHIVELOG mode by shutting down Oracle, starting it back up with the MOUNT and EXCLUSIVE options, and issuing ALTER DATABASE ARCHIVELOG then ALTER DATABASE OPEN. This also is a good time to change the passwords of the Oracle userids SYS and SYSTEM, which are set up with default passwords during CREATE DATABASE. (This should be done regardless of whether you are changing ARCHIVELOG mode.) Both of these actions are shown in the following example.

```
//ORAALOG  JOB 1,'Database Control'
//PLUS     EXEC PGM=SQLPLUS,PARM='/NOLOG',REGION=4M
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
```

```
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//ORA@ORA1 DD DUMMY
//INITORA  DD DISP=SHR,DSN=ORAN.ORAV.PARMLIB(INITORA1)
//SYSIN    DD *
CONNECT / AS SYSDBA
SHUTDOWN
STARTUP MOUNT EXCLUSIVE PFILE=/DD/INITORA
ALTER DATABASE ARCHIVELOG;
ALTER DATABASE OPEN;
ALTER USER SYS IDENTIFIED BY SECRET1;
ALTER USER SYSTEM IDENTIFIED BY SECRET2;
EXIT
/*
```

You can now proceed with creating ids for Oracle users, adding tablespaces and tables for applications, and so forth.

Configuring a Database Service Using ISPF Panels

Database Customization

The following steps guide you through the process of setting up your user environment to run the Oracle Installation Dialog Facility to configure an Oracle database. The Oracle software for this release must have already been installed.

Step 1: Selecting Oracle ISPF Profile Files

The Oracle installation and customization process uses two ISPF profile files: the tso_userid.ORISPF.O022PROF sequential data set and the O022PROF member of the tso_userid.ISPF.ISPPROF PDS. Both of these files are created automatically if they do not exist when you invoke the Oracle installation and customization process. However, if you are going to configure an Oracle database after someone else performed the product installation, you must use the profile files of the person who performed the product installation.

If you performed the Oracle product installation under the same TSO userid that you are using for the other customization, then proceed to Step 2.

If you did not perform the Oracle product installation under the same TSO userid that you are using for the other customization, then you must:

- Copy the O022PROF member from tso_userid.ISPF.ISPPROF of the TSO userid under which the Oracle product installation was performed to the tso_userid.ISPF.ISPPROF of the TSO userid under which you are performing the other customization.
- Copy or rename sequential data set tso_userid.ORISPF.O022PROF of the TSO userid under which the Oracle product installation was performed to a sequential data set named tso_userid.ORISPF.O022PROF of the TSO userid under which you are performing the other customization.

Step 2: Execute ISPF and Invoke the Oracle Installation and Customization Process

Invoke ORIPO01 to start the customization process.

To invoke the ORIPO01 CLIST, logon to TSO, enter ISPF, and select option 6 (TSO Command Processor) from the main ISPF menu. From option 6, use the EXEC command to invoke the ORIPO01 CLIST from the ISPCLIB data set created in Step 2:

```
 ------------------------ ISPF COMMAND SHELL ------------------------
Enter TSO or Workstation commands  below:
===➔ EXEC ‘oran.orav.ISPCLIB(ORIPO01)’
```

If you are using the correct ISPF profile files, panel OR@PRIM should be displayed immediately. Proceed to "Step 3: Select Product Set to install (Panel OR@PRIM)".

If you see prompts on the screen prior to the display of OR@PRIM (prompts that are asking for data set name qualifiers), then you did not get the correct ISPF profile files. Return to "Step 1: Selecting Oracle ISPF Profile Files".

Step 3: Select Product Set to install (Panel OR@PRIM)

Place an S in front of "Oracle9*i* Server", leaving the high-level and second-level qualifiers that were used during the product installation process as they are.

When you select "Oracle9*i* Server" on panel OR@PRIM and press [Enter], panel OR@INST is displayed. You are led through a series of panels that let you select and customize the Oracle server and tools for your installation.

Step 4: Select Installation/Customization Option to be Performed (Panel OR@INST)

Different installation and customization options may be performed from this panel. These options go in pairs: a definition option which takes you through a series of ISPF panels and a generate option which produces a job stream based on the information provided on the ISPF panels to further populate the Oracle INSTLIB library. (The generate option is typically followed by other tasks to complete the installation or customization that you are performing.)

From the Oracle Primary Option Menu, select option 3 "Define New Oracle Database Parameters".

Step 5: Begin Database Parameter Definition (Panel ORNEWDB)

After selecting the "Define New Oracle Database Parameters" definition option from the Primary Option Menu, you will see panel ORNEWDB. This panel shows a list of the various database customization parameters that must be provided in order to complete the customization process.

To begin the new database definition process, press [Enter] to begin with option 1. The steps that follow describe the procedure and the customization panels that you encounter.

The following rules apply to the customization panels:

- Most customization parameters have default values you can change by typing in a new value in place of the default value. Other required parameters have no default values, and you must supply values for them before proceeding to the next panel.
- You can move forward from one panel to the next by entering C on the command line.
- Most panels allow you to return to the previous panel by pressing the [PF3] key.

Step 6: Specify Oracle Database Name and VSAM Qualifiers (Panel ORDBIP10)

1. Oracle Database Name

Enter the name of the Oracle database that you want to create. The value that is specified here will be used as the database name in the generated initora file and the OSDI database service name. This defaults to ORDB to provide an example. It may be changed to any valid database name of your choosing.

2. Oracle Database System Identifier (SID)

Enter the service identifier for the database that you want to create. The value that is specified here will be used as the OSDI database service identifier. The SID is limited to 4 characters because it is used to generate PDS member names in the INSTLIB and PARMLIB libraries for jobs and parameter files that are used later in the database customization process.

3. Oracle VSAM Qualifiers

Enter the high-level qualifier (up to 8 characters) and second-level qualifier (up to 17 characters, including periods) that you want to use for the database data, control, and redo log files.

Note: Although the Oracle instance alert log file is not a VSAM file, it will be allocated using the same qualifiers as the VSAM files that are specified in this step.

When the information in this panel is correct, enter C to continue to the next panel.

Step 7: Specify Database Region Parameters (Panel ORDBIP15)

1. Maximum Number of Address Spaces

Enter the maximum number of address spaces to be allowed for the database service. Refer to "MAXAS" on page 3-4 for information about the MAXAS database service parameter.

2. Initial Number of Address Spaces

Enter the number of address spaces that you want to start each time the OSDI database service is started. Refer to "INIT_ADR_SPACES | INTADSPC" on page 3-10 for information about the INIT_ADR_SPACES database region parameter.

3. Maximum Number of Sessions per Address Space

Enter the maximum number of sessions that are allowed to run in this database address space. Refer to "MAX_SESSIONS | MAXSESS" on page 3-13 for information about the MAX_SESSIONS database region parameter.

4. Initial Size of the C Stack

Enter the size of the C stack to be allocated initially for each session. Refer to "INIT_STACK_SIZE | INTSTKSZ" on page 3-11 for information about the INIT_STACK_SIZE database region parameter.

When the information in this panel is correct, enter C to continue to the next panel.

Step 8: Specify the Database, Control, and Redo Log Files (Panel ORDBIP20, Panel ORDBIP25, Panel ORDBIP30)

These panels allow you to specify the size and volume of the database, control, and redo log files. If you want to change the default information, then enter the correct information for your installation in each panel. You can also select the space allocation unit in these panels. Records, tracks, or cylinders can be specified.

The database, control, and redo log files are created with the high-level and second-level qualifiers specified in panel ORDBIP10.

The file list on each of these panels is an ISPF table that you can scroll forward and backward using the [PF8] and [PF7] keys. When you press the [Enter] key, the screen scrolls forward, and the first entry is no longer displayed. To display the first entry again, press [PF7] to scroll backward.

Recommendations

There must be at least two redo log files. Oracle Corporation also recommends using a minimum of two files for database information and two files for control information.

You need a minimum of two database files to segregate user tables and data information from the Oracle data dictionary.

Without this segregation, you might not be able to recover user data, because the SYSTEM tablespace can never be taken offline. The installation process defaults to include a user tablespace. If you do not want to use it, then delete the user tablespace row on the panel.

You need a minimum of two control files in case one becomes damaged. Also, Recovery Manager (RMAN) requires a snapshot control file that is separate from the control files that are used by the Oracle9*i* server. Each control file needs to be on a different storage device. Oracle cannot initialize without a control file.

Refer to "Providing a Snapshot Control File" on page 6-8 for more information about using RMAN.

Step 9: Specify File Processing Parameters (Panel ORDBIP35)

This panel allows you to specify the file management parameters for the default file group (DFLT). Refer to "Server File Management Parameters" on page 4-8 for information about these parameters.

When the information in this panel is correct, enter C to continue to the next panel.

Step 10: INSTLIB/ISPSLIB File Tailoring Information (panel ORDBIP40)

1. Oracle ISPSLIB (skeleton) Library member

This item is showing the file tailoring skeleton member that will be used during the generate step. You cannot change this value.

2. Oracle Installation Library

Provide information about the INSTLIB library in which the installation jobs are built. This is normally the same data set that is created when the first file is downloaded from the Oracle distribution tapes (in "Installation Set Up and Initialization" Step 1 in the *Oracle9i Enterprise Edition Installation Guide for OS/390*). However, you may change the Oracle INSTLIB library name if required. Enter NEW or SHR for DATA SET DISPOSITION. The DATA SET DISPOSITION defaults to SHR. If the data set is new, then specify a VOLUME SERIAL and DEVICE TYPE.

When the information in this panel is correct, enter C to continue to the next panel.

Step 11: Define JOB card for database jobs (panel ORDBIP50)

This panel allows you to define the JOB card structure to be used with each of the generated database jobs. The additional lines can be used to add JES subsystem control cards or similar requirements. The JOBNAME and the NOTIFY parameter default to your TSO logon userid. Change the default information as needed.

When the information in this panel is correct, enter C to continue to the next panel.

Step 12: New Database Definition Complete (panel ORDBIP90)

The appearance of this panel indicates that you have completed the New Database Definition process. You may use [PF3] to back up through the previous panels to make any corrections that you need. Or if you are satisfied with all the parameter values in the customization panels, you may use [PF4] to return to the Primary Option Menu panel.

Step 13: Generate Database Job

Select option 4 (Generate Database job) from the Oracle Primary Option Menu panel. The installation CLISTs generate a jobstream in a new member, ORDJA01, in the Oracle INSTLIB library.

The output displays as the generate step is executing. When you see the GENERATION PROCESS COMPLETED message, press [Enter] to return to the Oracle Primary Option Menu. Then press [PF3] to exit the install dialog.

Step 14: Run the Database Job (ORDJA01)

You do not need to edit the generated job, ORDJA01, although you might want to change the jobname. Submit this job after you have made any changes appropriate for your site.

Job ORDJA01 creates multiple members in the INSTLIB library, including a series of installation jobs with member names *sid*JB00 through *sid*JH00 where `sid` is the database service identifier that was specified on panel ORDBIP10.

Examine the output from the ORDJA01 job to confirm its successful execution. The return code from ORDJA01 should be 0, indicating successful execution.

Step 15: Run the Generated Database Jobs

Submit each job in the order shown here in the following sections (*sid*JB00, followed by *sid*JC00, and so on).

Even if the basic JOB card and related data are specified properly in the associated customization panel, you might need to change information such as the jobname in each job before submitting it for execution.

The following list of jobs describes the purpose of each job and any special considerations for its execution.

*sid*JB00

This job uses the IBM IDCAMS utility to define the database, control, and redo log files. The DASD volumes and allocation quantities for these data sets were supplied during the ISPF customization process.

This job also creates the Oracle instance Alert log file.

*sid*JC00

This job copies customized parameter members to the Oracle PARMLIB library. These members include:

`sid``INIT`	database init.ora parameter file
`sid``PARM`	OSDI database region parameter file
`sid``SVC`	OSDI database service parameter file
`sid``FPS`	database file processing parameter file

where `sid` is the database service identifier specified during the database customization process.

These parameter files should be reviewed and modified as necessary for your installation, because some of the default values that are used may not be appropriate for your site.

The *sid*SVC PARMLIB member contains the OSDI DEFINE SERVICE and START commands for the database service. This member is not intended to be used as is. You should copy the contents of this member into your OSDI subsystem parameter file so that the database service will be defined and started every time the OSDI subsystem is initialized.

The *sid*INIT PARMLIB library member has the COMPATIBLE parameter set to the current release number (9.2.0). If you need to run with COMPATIBLE set to an earlier release, then you might not be able to use all Oracle9*i* features. Refer to "Oracle Initialization Parameter Considerations" on page 3-15 for more information about init.ora parameters and Oracle9*i* for OS/390.

The *sid*FPS PARMLIB library member contains only a Default file group entry. If the Default file group parameters are not suitable for all the file groups, you may add to this member the other file groups with the parameters that you need. Refer to "Server File Management Parameters" on page 4-8 for more information.

This job also copies a customized database procedure to the system procedure library that was specified during the product installation process. The procedure is named ORA*`sid`* where *`sid`* is the database service ID that was specified during the database customization process.

*sid*JD00

This job invokes the SQL*Plus utility to create the new database and data dictionary using the VSAM files created by job *sid*B00. No other jobs should be run against this database while this job is running.

Prior to running this job, you must have initialized the OSDI subsystem and defined and started the database service. Refer to Chapter 2, "Configuring and Initializing the Subsystem" and to the earlier part of this chapter for more information.

If you have not already initialized your OSDI subsystem, you may do so now. If you have put the service definitions and start commands in your OSDI subsystem parameter file, then the database and net services that you have customized to this point for the current OSDI subsystem will be defined and started as part of the subsystem initialization.

If you have initialized your OSDI subsystem, then you will need to manually enter OSDI commands to define and start your database service in order to be able to use it.

*sid*JD01

This job will set up the database for running Java.

This job requires that the Oracle9*i* database started task has authority to access OMVS services. For more information about setting up a default OMVS segment, refer to IBM document SC28-1890-8, *USS Planning Guide*.

*sid*JE00 (for SQL*Plus)

This job loads the SQL*Plus help data into the database. The database must be running when this job is run. You can omit this job if you do not want online help or if you have limited database space.

You might receive messages indicating that a synonym, table, or view does not exist. These messages are normal and do not indicate an error condition. Examine the output for other error messages, if any.

*sid*JF00

This job creates the user and demonstration tables used in the Oracle documentation: the SCOTT userid and the EMP, DEPT, SALGRADE, and BONUS tables. The database must be running when this job is run. You can omit this job if you do not want the demonstration tables installed or if you have limited database space.

You might receive messages indicating that a synonym, table, or view does not exist. These messages are normal and do not indicate an error condition. Examine the output for other error messages, if any.

*sid*JH00 (for Oracle9*i* Text)

This job sets up the database for the use of the Oracle9*i* Text feature. It creates the user CTXSYS and the database objects that are required by Oracle9*i* Text. User CTXSYS is created with the password CTXSYS, which you can change after this job completes by using the SQL command ALTER USER. The database must be running when this job is run.

4

Defining OS/390 Data Sets for the Oracle Database

This chapter describes OS/390-specific considerations for the files used by an Oracle9*i* database. In general, what Oracle product literature refers to as "files" will be data sets in your OS/390 system. Some Oracle product features access files that are part of the hierarchical file system of OS/390 Unix System Services (USS) rather than OS/390 data sets. However, the files that make up an Oracle database are all OS/390 data sets.

Refer to *Oracle9i Database Concepts* and the *Oracle9i Database Administrator's Guide* to learn about the uses and relationships among Oracle databases, tablespaces, and operating system files. Refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* for a discussion of the interaction between Oracle database tools and OS/390 data sets.

The following topics are included:

- Oracle Database Files on page 4-2
- Tablespaces and OS/390 Space Management on page 4-4
- Server File Name Syntax on page 4-6
- Server File Management Parameters on page 4-8
- Pre-allocating Database Files on page 4-14
- Oracle Managed Files on OS/390 on page 4-15

Oracle Database Files

A number of files are used by an Oracle database instance. Some of these files are not an intrinsic part of the database. These include input parameter files, trace or diagnostic log files, and other similar files. Most of these will be sequential data sets, PDS members, or JES spool (SYSOUT) files on your OS/390 system.

Of greater interest here are the files that comprise the database and that contain application data and the internal Oracle database structures required to manage this data. Every Oracle database consists of at least four such files: one **control file**, one **database file** for the SYSTEM tablespace, and two **redo log files**. This is an absolute minimum. Normally, a database contains many more files than four. When you first create a database, you will specify an initial group of files. Later, as applications are added and as operating needs change, you may add or remove files according to need.

All of the files comprising an Oracle database are VSAM linear data set (LDS) clusters on OS/390. (For compatibility with past releases, VSAM Entry-Sequenced Data Sets with a control interval size of 4K are also accepted.) The Oracle server can create these data sets for you by internally invoking the OS/390 IDCAMS utility and passing it DEFINE CLUSTER commands. This is done automatically during processing of SQL statements that create or add to the database structure, including CREATE DATABASE, CREATE TABLESPACE, and certain ALTER…ADD statements.

In most cases, it also is possible to pre-allocate Oracle database files by invoking IDCAMS yourself and issuing your own DEFINE CLUSTER commands prior to issuing SQL statements that add the files to the database. The main advantage to pre-allocation is that it gives you more control over physical data set placement.

If you omit file specifications on a CREATE DATABASE request, Oracle normally creates a minimal set of files using default data set names discussed in the following sections. This practice is not recommended for production databases. An optional facility called Oracle Managed Files (OMF) provides a different default naming scheme that is aimed at production database use. OMF is discussed in "Oracle Managed Files on OS/390" on page 4-15.

Control File

The control file is a relatively small file used to record Oracle database instance control information. Control file availability is critical for overall database availability, and the Oracle database will therefore maintain multiple mirror-image copies of the control file if told to do so. Oracle Corporation recommends using at least two control file copies for any production database. The control file copies should be on separate physical storage devices and, if possible, on separate I/O paths in order to minimize the risk of losing both (or all) of them to media or path failure. You specify the data set name(s) of the control files in the init.ora parameter file that you supply to the Oracle database STARTUP command.

If you do not specify a name for the control file and you are not using OMF, the following default file name will be used:

```
"&ORAPREFD..&ORASRVN..DFLCNTL.DBF"
```

The system symbols &ORAPREFD and &ORASRVN are discussed in Appendix C, "Oracle9i for OS/390 System Symbols".

Database Files

The database files contain all the database data, both application tables, indexes and the like, and the Oracle database server's internal dictionary objects. Each tablespace in your database contains at least one database file. The SYSTEM tablespace, where the Oracle database keeps its internal dictionary structures and stored PL/SQL procedures, is created automatically when you issue CREATE DATABASE.

The names of one or more files to be used for the SYSTEM tablespace are supplied in the CREATE DATABASE statement or generated by the Oracle server using Oracle Managed Files. Normally, you will not store application data in the SYSTEM tablespace: after your database is created, you will create one or more additional tablespaces for application data. The CREATE TABLESPACE statement supplies the names of files to be used when adding a new tablespace, or the names can be generated by the Oracle server using Oracle Managed Files.

If you do not specify names for the database files and you are not using OMF, the following default file names will be used:

`"&ORAPREFD..&ORASRVN..DFLDBS1.DBF"` (default database file)

`"&ORAPREFD..&ORASRVN..DFLTEMP.DBF"` (default temporary tablespace file)

The system symbols &ORAPREFD and &ORASRVN are discussed in Appendix C, "Oracle9i for OS/390 System Symbols".

Redo Log Files

The redo log files record changes to database data (both application data and internal control data) and are critical for restart and recovery. An Oracle database instance must have at least two redo log files. Redo data is written to the log more or less sequentially from the beginning of the file to the end. Like control files, the Oracle server can write multiple mirror images of a log to reduce the risk of data loss due to media failure.

The term "log file group" refers to one or more mirror-image copies of a given log. When a log file (or group) is filled, the Oracle server automatically switches log writing to the next available log file or group. The frequency of this switching is a function of the size of the redo log files and the amount of database update activity.

The data set names of the log files can be supplied in the CREATE DATABASE statement, or generated by the Oracle server using Oracle Managed Files. In addition, log files can be added to a database later, using ALTER DATABASE…ADD LOGFILE.

If you do not specify names for the redo log files and you are not using OMF, the following default file names will be used:

`"&ORAPREFD..&ORASRVN..DFLLOG1.DBF"` (default redo log file # 1)

`"&ORAPREFD..&ORASRVN..DFLLOG2.DBF"` (default redo log file # 2)

The system symbols &ORAPREFD and &ORASRVN are discussed in Appendix C, "Oracle9i for OS/390 System Symbols".

Archive Log

When you operate the database in ARCHIVELOG mode (normally the case for all production databases), the Oracle server copies filled logs to another (newly-created) file in order to make the filled log available for reuse. The new file that is created by such an operation is called an **archive log**. The data set names of archive logs are generated using a pattern that you specify in your init.ora parameters. This name includes components that are distinct for each log, ensuring a unique data set name for each archive. This is one case where file pre-allocation is not possible: archive logs are always created by the Oracle server via internal IDCAMS call. Archive logs usually are required (read) during recovery, after restoring database files from backups. For more information, refer to Chapter 6, "Database Backup and Recovery".

Tablespaces and OS/390 Space Management

Tablespaces can be enlarged in two ways. If the file(s) comprising the tablespace were specified with the AUTOEXTEND clause of CREATE or ALTER

TABLESPACE, the Oracle server will attempt to extend one or more files automatically when the tablespace becomes full. If AUTOEXTEND was not specified, or automatic file extension fails, or if you want to enlarge a tablespace manually (in advance of a large addition of data), you can use the SQL statement ALTER TABLESPACE...ADD DATAFILE to add one or more new files to the tablespace.

When you create a tablespace, its initial size is the sum of the sizes of all datafiles you specify on the CREATE TABLESPACE statement. These are the actual sizes of the data sets, if you are using preallocated data sets or reusing existing files, or the amounts specified with the SIZE keyword if the server is creating the files. In the latter case, the Oracle server uses only the SIZE amount, and some unused space (overallocation) may be present due to VSAM allocation rounding (discussed below). The unused space is never more than one cylinder per file, but you should be aware of this rounding when planning your Oracle server disk space needs.

A tablespace can be enlarged in two ways: by extending existing files of the tablespace or by adding new files. Existing files can be extended both automatically, on demand as data is added, and explicitly with a DDL SQL statement. Automatic file extension is enabled when the AUTOEXTEND clause is included in the file specification of CREATE or ALTER TABLESPACE. Automatic file extension is attempted when a database insert or update requires more free space than is currently available in the tablespace. This can be triggered by any user session; no special privileges are required. Explicit extension is requested with an ALTER DATABASE DATAFILE...RESIZE statement using SQL*Plus or a similar interface, and must be done from a session with ALTER DATABASE authority (normally a database administrator).

Adding new files to a tablespace is only done manually, by issuing an ALTER TABLESPACE ADD DATAFILE statement using SQL*Plus or a similar interface.

There are several OS/390-specific considerations with file extension and AUTOEXTEND. When a tablespace consists of multiple datafiles, all specified with AUTOEXTEND, you cannot control which data set the Oracle server extends in a given situation. Second, the amount by which to extend a file is governed by the NEXT amount from the AUTOEXTEND clause, never by a secondary space quantity in the data set's ICF catalog entry. (Secondary space is not included on server-generated DEFINE CLUSTER commands. Secondary space that you specify on the DEFINE CLUSTER for a preallocated database file is ignored except for the influence it exerts on CA size and rounding.)

VSAM space allocation always rounds an allocation amount to a Control Area (CA) multiple, and that can distort the way the AUTOEXTEND NEXT amount is handled. CA size is not specified explicitly but is derived from other DEFINE

CLUSTER parameters: it is the lesser of the primary space quantity, the secondary space quantity (if any), and the device cylinder size, but never less than one whole track. When the Oracle server issues DEFINE CLUSTER it does not include a secondary space quantity, so CA size is one cylinder unless your SIZE is very small (less than a cylinder). You can control CA size on preallocated database files by specifying a secondary space amount that is less than a cylinder; although the secondary amount is not used by the Oracle server, it does determine CA size as described here.

The Oracle server is not aware of VSAM space rounding, so if your NEXT amount is less than one CA, the space added by CA rounding is not used. It does get used, however, when the file is extended again. In some cases, file extension is satisfied entirely with existing allocated but unused space, without invoking VSAM secondary allocation. Be aware of VSAM space rounding behavior when choosing a NEXT amount for a datafile.

One last consideration with file extension concerns Oracle servers that are configured to run in multiple address spaces (where the DEFINE SERVICE command specifies a MAXAS number greater than one). The OS/390-specific logic for file extension requires establishing two concurrent "opens" to a file that is accessed from multiple server address spaces. This in turn requires that the VSAM cluster be defined with cross-region shareoption 3. The cross-system shareoption is irrelevant.

If you preallocate a data set for use as a tablespace file and you want to specify AUTOEXTEND for it, you must specify SHR(3,3) in your DEFINE CLUSTER command. If you let the server create the file for you, you must use the SHAREOPTION file management parameter to specify that cross-region shareoption 3 be included on the server's DEFINE CLUSTER command. (Server file management parameters are discussed in "Server File Management Parameters" on page 4-8.) This is required only for Oracle servers whose MAXAS is greater than one, and is so even if only one address space is actually started. If your Oracle server is defined with MAXAS(1) (which is the default), AUTOEXTEND processing does not require SHR(3,3).

Server File Name Syntax

When specifying files to an Oracle server on OS/390 (in init.ora parameters, SQL statements, and so forth), you use an extended version of the file name syntax used by IBM C/C++ and LE/370. IBM syntax encompasses OS/390 data sets that are specified by data set name or DD name, as well as Hierarchical File System (HFS) files that are specified by full or relative path name. OSDI extends LE/370 file syntax slightly with a specific notation for spool (SYSOUT) output files.

Data Sets

The full OSDI syntax for OS/390 data sets is:

```
//'dsname'
```

where `dsname` is a 1-character to 44-character data set name that conforms to OS/390 rules, except for letter case (lowercase letters are converted to uppercase by OSDI). Simpler forms are also accepted, but they are converted to the full form during processing. Inside the Oracle server, all of the following are equivalent to the full OSDI syntax for OS/390 data sets:

```
//dsname
dsname
'dsname'
```

If you use the file name forms that include apostrophes in a SQL statement such as CREATE DATABASE or ALTER TABLESPACE, then the entire file name string is enclosed in apostrophes, and the embedded apostrophes must be doubled.

Non-VSAM input files (usually parameter files) can include a PDS member name in the file name string as in:

```
//'dsname(member)'
```

The same simpler name formats are also supported. Note that VSAM files and non-VSAM files that are used for output (for example, trace files) cannot include a member name.

Data sets can also be specified by DD name using the full syntax:

```
//DD:name
```

where `name` is the 1-character to 8-character DD statement name. A simpler form without the leading slashes is also permitted.

In general, Oracle Corporation recommends that files be specified to the server using data set names rather than DD names. The use of data set names allows files to be added to the database without requiring changes to the service JCL procedure, which requires stopping and restarting the service. Even when a file is specified using DD:, the Oracle database might retrieve the data set name from the associated DD statement and use that for subsequent accesses to the file. This eliminates any flexibility that might be gained by specifying files using DD names.

The following sample SQL statement adds a new file to an existing tablespace:

```
alter tablespace tbolap3 add
datafile '//''oracle.tbolap3.dbf6''' size 450m;
```

Note: The outer apostrophes surrounding the file name are part of the ALTER TABLESPACE SQL statement syntax. The inner (doubled) apostrophes are part of the file name. The SQL parser converts the doubled apostrophes to single apostrophes.

SYSOUT Support

OSDI extends LE/370 file syntax to provide direct use of SYSOUT files. SYSOUT can be used anywhere that the server requires a sequential (non-VSAM) output-only data set. This is a convenient vehicle for items such as trace files. The syntax for a SYSOUT file is:

```
//SYSOUT:c,form,dest
```

where *c* is a SYSOUT class, *form* is a forms identifier, and *dest* specifies a destination. All three positional parameters are optional and default to whatever defaults are established for the server address space. Note that the default class is equivalent to specifying SYSOUT=*. The leading slashes can be omitted, and the word SYSOUT can be abbreviated to just the letter S.

The following are all examples of valid SYSOUT file specifications:

```
S:
//SYSOUT:*
//S:X,,NPFPRINT
SYSOUT:*,STD
```

Server File Management Parameters

To create a new database or add files to an existing database, you issue a SQL CREATE or ALTER statement to the Oracle server. This statement allows you to specify whether a file is pre-allocated or, if not, the size with which the new file should be created by the server. There is no provision in Oracle SQL for supplying additional OS/390-specific parameters for creating the associated data set. Instead, you can supply **file management parameters,** using the ORA$FPS DD statement in the database service JCL procedure, to control most of the parameters in the

IDCAMS DEFINE CLUSTER command that the Oracle server issues to create a new file.

File management parameters are also used for certain non-VSAM files, including the sequential backup data sets created and read with an RMAN External Data Mover (EDM). In the EDM case the parameters are read and used by the ORAEDM program running in a separate address space instead of the Oracle server.

In the Oracle server, the ORA$FPS file management parameters are meaningful mainly for files for which the Oracle server issues the IDCAMS DEFINE CLUSTER command. As discussed in the section "Oracle Database Files" on page 4-2, you may have the option of pre-allocating database files by issuing your own DEFINE CLUSTER command. The DEFINE CLUSTER command is discussed in "Pre-allocating Database Files" on page 4-14. This command gives you complete control over most of the DEFINE parameters, which may be desirable when creating the permanent parts of a production database.

Oracle server archive log files cannot be pre-allocated, however. Assuming that you are running your database instance in ARCHIVELOG mode, these files are created by the Oracle server whenever an online log fills and is archived. They are created using a generated data set name whose pattern you control with init.ora parameters. Because these files must be created by the server, the file management parameters for the DBAL group are critical for proper operation of the database.

Server file management parameters are read during service startup. You can change the parameters and cause the server to reread them without stopping and restarting the service, by using a MODIFY command. This is discussed in the section "Other Database Service Commands" on page 5-5.

File Group Names

File management parameters are organized by file group, with each group having a distinct 4-character name. The current valid file group names for an Oracle server are:

- DBAL - database archive log file
- DBBA - archive log backup piece (RMAN backup to disk)
- DBBI - datafile incremental backup piece (RMAN backup to disk)
- DBBK - datafile backup piece (RMAN backup to disk)
- DBCP - datafile copy
- DBCT - database control file

- DBDB - database data file (one that is part of a tablespace)
- DBDR - disaster recovery configuration file
- DBOL - database redo log file
- DBMI - miscellaneous file
- DBPM - text (non-VSAM) parameter file, including PFILE
- DBSP - server parameter file (SPFILE)
- DBST - database internal trace file
- DBTR - database trace file
- NTPM - network parameter file
- NTTR - network trace file (non-VSAM)
- Pnnn - RMAN EDM backup file (non-VSAM; "nnn" is POOL number)

In addition to the above, the file group name DFLT can be specified to supply default parameters. Default parameters are used for any group that you do not specify explicitly.

File Management Parameters in an External Data Mover

The External Data Mover (EDM), discussed in Chapter 6, "Database Backup and Recovery", executes in a separate address space from the Oracle server and supports database file backup and restore operations under control of the RMAN utility. EDM uses file management parameters to control creation of non-VSAM sequential backup data sets on disk or tape.

In this case, the ORA$FPS DD statement in the EDM JCL procedure supplies the parameters. The file groups for EDM backups all have names of the form "Pnnn" where "nnn" is the storage pool number from an RMAN BACKUP request. Certain parameters are specific to EDM and are ignored when specified for file groups used by the Oracle server.

File Management Parameters and Syntax

The ORA$FPS parameter file contains file group definitions which are specified using `keyword(value)` syntax. Each definition must start with the keyword `FILE_GROUP(name)` and continues until the next FILE_GROUP keyword is encountered. Comments must start with an asterisk (*) and can begin in any column as long as comments (that are on the same line as keywords) are separated from the last keyword by at least one blank.

Keywords can be coded one per line or strung together on the same line separated by at least one blank, but a keyword (value) pair cannot be split across two lines. No defaults are defined for the parameters. If a keyword is not coded, then it will not be used on the DEFINE CLUSTER or dynamic allocation that is used to create data sets in the associated group. The only parameter that can be overridden or supplied from the SQL command line is SPACE, which Oracle server supports via the SIZE keyword in SQL statements that specify files. The default file group (DFLT) supplies parameters for any file group that is completely omitted from the file management parameters.

`BUFNO(`*`nnn`*`)`	Specifies the number of I/O buffers to be allocated by EDM for use during backup and restore operations, where *nnn* is a decimal number from 1 to 255. Each buffer is equal in size to the block size of the associated backup data set. The entire buffer requirement allocated is above the 16MB line. BUFNO(3) is the default.
`CREATE_MODELDSN(`*`dsn`*`)`	Specifies a data set name to use as a MODEL in IDCAMS DEFINE CLUSTER commands. This value is mutually exclusive with the SMS class name parameters. CREATE_MODELDSN can be abbreviated MODEL.
`DATACLAS(`*`classname`*`)`	Specifies an SMS data class name to be specified on DEFINE CLUSTER or dynamic allocation requests to create new data sets. DATACLAS can be abbreviated DATACL.
`DEFAULT_SPACE(`*`primary secondary`*`)`	Specifies default primary and secondary space quantities for a data set that is being created. The primary quantity applies only in situations where the Oracle server has not indicated the desired file size. The secondary quantity is optional and is ignored at this time. Both values must be numbers and are expressed in kilobyte (1024-byte) units. DEFAULT_SPACE can be abbreviated SPA.

`EXPIRATION_DATE(` *yyyyddd* `|` *yyddd* `)`	Specifies a data set expiration date to be used for allocation during EDM backup data set creation or restoration, where *yyyyddd* is a 4 digit year and 3 digit day of the year, and *yyddd* is a 2 digit year and 3 digit day of the year. EXPIRATION_DATE can be abbreviated EXPDT.
`FILE_GROUP(`*name*`)`	Specifies the file group to which the file management parameters belong, where *name* is one of the allowed 4-letter file group names. This ends any in-progress file group definition and begins a new one. FILE_GROUP can be abbreviated FILE.
`MGMTCLAS(`*classname*`)`	Specifies an SMS management class name to be specified on DEFINE CLUSTER or dynamic allocation requests to create new data sets. MGMTCLAS can be abbreviated MGMTCL.
`RECALL( ALL | NONE )`	Specifies whether migrated data sets may be recalled during backup data set allocation for an EDM restore operation. ALL indicates that the recall of migrated data sets is allowed during backup data set allocation. NONE indicates the recall of migrated data sets is not allowed during backup data set allocation. RECALL(NONE) is the default.
`SHAREOPTION(` *n* `)`	Specifies the VSAM cross-region shareoption to be used on DEFINE CLUSTER. SHAREOPTION(1) is the default. It must be specified as SHAREOPTION(3) if the associated data set is to use the AUTOEXTEND feature in a server configured to run in multiple address spaces. See the discussion of AUTOEXTEND and OS/390 space management under "Tablespaces and OS/390 Space Management" on page 4-4. SHAREOPTION can be abbreviated SHR.
`STORCLAS(`*classname*`)`	Specifies an SMS storage class name to be specified on DEFINE CLUSTER or dynamic allocation requests to create new data sets. STORCLAS can be abbreviated STORCL.

`UNIT(`*`unitname`*`)`	Specifies an allocation unit name to use in dynamic allocation requests that create new non-VSAM data sets. This parameter is intended for future use when non-VSAM files use file management parameters.
`VOLUMES(`*`volser`*`)`	Specifies a volume serial number to use in IDCAMS DEFINE CLUSTER commands for VSAM data sets or in dynamic allocation requests that create non-VSAM data sets. Only a single volume serial can be specified. Because of this limitation, it is recommended that you use storage management class parameters instead of explicit volumes. VOLUMES can be abbreviated VOL.
`VOLUME_COUNT(`*`nnn`*`)`	Specifies a volume count to be used for allocation during EDM backup data set creation, where *`nnn`* is a decimal number between 1 and 255. This parameter is normally used when the backup data set is to reside on tape. VOLUME_COUNT can be abbreviated COUNT.

Example

Storage management parameter example:

```
* Oracle server file management parameters
* Tablespace data files
FILE_GROUP(DBDB)
DEFAULT_SPACE(10000 )               * a comment
CREATE_MODELDSN(ORBL.ORAV8.DB1)
* Archive logs
FILE_GROUP(DBAL)
DEFAULT_SPACE(11100   9000)
DATACLAS(OSDIDC2) MGMTCLAS(OSDIMC2)  * 2 keywords on one line
* Default for groups not specified
FILE_GROUP(DFLT)
DEFAULT_SPACE(10000   5000)
UNIT(SYSDA) VOL(TEMP01)
```

Pre-allocating Database Files

If you choose to pre-allocate Oracle server control, log, or database files, then you execute the OS/390 IDCAMS utility, and you supply one or more DEFINE CLUSTER commands. Alternatively, DEFINE CLUSTER can be issued directly in a TSO session. For details on the DEFINE CLUSTER command, refer to *DFSMS/MVS Access Method Services for ICF*. This section discusses DEFINE CLUSTER requirements specific to Oracle database files.

You can give an Oracle database file any data set name that conforms to your installation's naming standards and/or security requirements. You will specify this name to the Oracle server later (in a SQL statement or, in the case of control files, in the init.ora parameter file) using the file name syntax discussed earlier in this chapter.

Note: Oracle Corporation recommends using a consistent set of qualifiers for the left-hand portion of all data set names associated with a given database. Certain Oracle database features, particularly the standby database features, are usable only when all data sets comprising the database share a common set of leftmost data set name qualifiers.

The amount of space to allocate to a file depends on how the file is used and on your requirements. Refer to the *Oracle9i Database Administrator's Guide* for discussion of database file sizing for each type of file. The IDCAMS DEFINE command can specify space in any of several different units. Choose the unit that is easiest for you. The Oracle server has no preference for space allocation units. Space can be specified in tracks, cylinders, megabytes, kilobytes, or records. Any secondary space quantity in your DEFINE is ignored by the Oracle server.

When you specify the pre-allocated file to the Oracle server in a SQL CREATE or ALTER statement, you must omit the SIZE keyword and specify REUSE, indicating that the file already exists. Except in the case of control files, the Oracle server will use all of the primary space that you pre-allocated. (With control files, the Oracle server uses exactly the amount of space it needs to contain the internal control structures, whose size depends on some of the other parameters of CREATE DATABASE. This might be less than the space that you pre-allocated.)

Multivolume VSAM clusters are not supported in an Oracle database. If you need to add space to a database in multiple volume increments, or if you want to "stripe" a tablespace across volumes, then create a separate cluster on each volume and specify them as multiple files in the SQL CREATE or ALTER statement.

The DEFINE command must specify the LINEAR keyword, indicating that a VSAM linear data set (LDS) is being created. An LDS always has a control interval size of 4K and does not contain VSAM logical record structures. DEFINE parameters such as CONTROLINTERVALSIZE and RECORDSIZE are therefore not used. (If space is specified using RECORDS, then IDCAMS assumes that each record equates to one 4K CI.) The VSAM SHAREOPTIONS default of SHR(1,3) is recommended for all database files except when the server is configured for multiple address space (MAXAS is greater than one). In this case, if automatic file extension is desired (using the AUTOEXTEND clause of the CREATE/ALTER TABLESPACE command), SHR(3,3) is required.

Depending on the standards of your installation, you may need to specify VOLUMES or one or more of the SMS parameters (STORAGECLASS, MANAGEMENTCLASS, and DATACLASS) in the DEFINE command.

The following is an example DEFINE command for an Oracle database file:

```
DEFINE CLUSTER( -
NAME(VSAM.QUALS.SYSTEM.DBF2)-
LINEAR -
STORAGECLASS(OSCM3A) -
MANAGEMENTCLASS(OMCM3A) -
MEGABYTES(150))
```

No other preparation, loading, or formatting is required before a pre-allocated file is added to the database. When you specify the new file in a SQL statement (such as ALTER DATABASE or CREATE TABLESPACE), the server will format all of the primary space of the data set. Adding a large file to the database will therefore incur a noticeable delay while formatting is done. (This is true whether files are pre-allocated or created by the server.)

Oracle Managed Files on OS/390

The Oracle Managed Files (OMF) feature of Oracle9*i* simplifies database administration by eliminating the need to specify the names of database files (control, log, and tablespace files) and to delete underlying files when the owning database element is logically dropped.

When you use OMF, you can omit the single-quoted filenames in the CREATE/ALTER DATABASE and CREATE/ALTER TABLESPACE statements because the Oracle server generates unique names for each file. When you drop an OMF log file or a tablespace comprising OMF files, the Oracle server deletes the files. In the case of DROP TABLESPACE, you can omit the INCLUDING CONTENTS AND DATAFILES clause.

To use OMF, you must do the following:

- Specify certain init.ora parameters involved in name generation
- Omit the single-quoted filename(s) from CREATE or ALTER SQL statements

The init.ora parameters for OMF are DB_CREATE_FILE_DEST and DB_CREATE_ONLINE_LOG_DEST_n. On OS/390, these parameters supply the left-hand portion of a data set name (high-level qualifier and possibly other qualifiers) and they must end with a period. The OMF parameters can be reset or changed without shutting down, using ALTER SYSTEM or ALTER SESSION. For OS/390-specific details on the OMF parameters, see "Oracle Initialization Parameter Considerations" on page 3-15. For general information on OMF, refer to the *Oracle9i Database Administrator's Guide*.

Even when you specify OMF parameters you can continue to specify explicit file names in CREATE and ALTER statements. In fact, it is necessary to do so when you want to use a preallocated file or reuse an existing file.

Oracle tablespace names can be up to 30 characters long. If you want to be able to associate an OMF-created data set with its owning tablespace, you must use tablespace names that are distinct in the first 8 characters.

The right-hand portion of an OMF-generated filename depends on the type of file and includes an encoded timestamp value for uniqueness. The complete data set name format for OMF files is shown in the following example:

```
control files:              destOMC.Attttttt
log files:                  destOMLnnn.Attttttt
permanent tablespace files: destOMD.tsn.Attttttt
temporary tablespace files: destOMT.tsn.Attttttt
```

where:

dest	is the destination string (_DEST) in the OMF parameter
nnn	is a 3-digit log group number
tsn	is up to 8 characters of the tablespace name
ttttttt	is the encoded timestamp (which looks like a random mix of letters and numerals)

Oracle allows underscores ("_") in a tablespace name, and any that are present are changed to "@" for use in the generated data set name.

Given the 44-character limit on OS/390 data set names, the above data set name formats impose a limit of 29 characters on DB_CREATE_ONLINE_LOG_DEST_n and 23 characters on DB_CREATE_FILE_DEST (assuming a tablespace name of 8 characters or more).

You can use Oracle-specific and OS/390 system symbols in the OMF parameters. The destination string must end with a period after any symbol substitutions have been performed.

SQL statements that exploit OMF are generally the same as their non-OMF counterparts except that single-quoted filenames are missing. REUSE is not recognized for OMF and you can omit SIZE, which defaults to 100M for all types of files.

The following is an example of a CREATE DATABASE command that uses OMF for both log files and for the SYSTEM tablespace:

```
CREATE DATABASE W1O9
 MAXINSTANCES 1
 MAXDATAFILES 1000
 MAXLOGFILES 10
 MAXLOGMEMBERS 1
 MAXLOGHISTORY 100
 LOGFILE SIZE 40M, SIZE 40M, SIZE 40M
 DATAFILE SIZE 128M AUTOEXTEND ON NEXT 32M MAXSIZE 256M;
```

The repeated "SIZE 40M" results in 3 log files of 40 megabytes each. You can leave the LOGFILE clause off completely and the Oracle server will create two log files (or log file groups) of the default 100M size. Similarly, the DATAFILE clause can be omitted if the 100M size is acceptable and no autoextension is required for the SYSTEM tablespace. In fact, if you accept all the defaults, it is possible to specify the CREATE DATABASE command as follows:

```
 CREATE DATABASE W1O9;
```

Likewise, you can do the same with the CREATE TABLESPACE command.

OMF files are distinguished internally by the presence of ".OMC", ".OML", ".OMT", or ".OMD" in the data set name. To avoid conflict with OMF, avoid using data set names containing these qualifiers in non-OMF operations.

5 Operating a Database Service

This chapter describes the OS/390-specific details of how to start and stop an Oracle database instance on OS/390. Use this chapter in conjunction with the information in the *Oracle9i Database Administrator's Guide* to establish Oracle database server operating practices in your installation.

The following topics are included:

Starting and Stopping the Database Service

As discussed in the *Oracle9i Database Administrator's Guide* and in the following section, you make an Oracle database instance available for use by issuing the Oracle database STARTUP command via Oracle SQL*Plus or a comparable Oracle utility. Before you can do this on OS/390, the OSDI-defined database service must be started. You start a database service using the OSDI START command documented in Appendix A, "OSDI Subsystem Command Reference". This will create one or more OS/390 address spaces according to the INIT_ADR_SPACES value in the server region parameters. These address spaces are what OS/390 calls "system address spaces", and they are similar to OS/390 started tasks (STCs). Each address space executes the JCL procedure that you specified via the PROC parameter of DEFINE SERVICE. The OSDI START command can be included in the subsystem parameter file so that the service is always started during IPL.

After the service is started, additional OSDI START commands can be issued to create additional address spaces for the service, up to the MAXAS limit that was specified in DEFINE SERVICE. Each additional START command adds one address space to the service. Added address spaces increase the amount of virtual memory available for database application sessions. Address spaces can be added before or after the Oracle database STARTUP command is issued. No Oracle database-specific action (such as the STARTUP command) is needed when adding address spaces.

Other than stopping the database service, which terminates all of the service address spaces, there is no way to reduce the number of address spaces of a running service. Database service address spaces cannot be stopped individually.

When you perform an Oracle database shutdown (by issuing the SHUTDOWN command via Oracle SQL*Plus), the associated service address spaces continue to run. You can startup and shutdown an Oracle database instance as many times as you want using the same set of service address spaces. The only situations that dictate stopping the service (terminating its address spaces) are:

- to effect a change to one of the service parameter files that is read only at service start: ORAFPS, ORAENV, or the main service parameters (data set specified using the OSDI PARM string);
- to effect a fix or upgrade to software modules (Oracle software, or IBM software that is fetched into the address space such as LE/370);
- to resolve a problem that has rendered the address spaces unusable in some way.

With this in mind, it may be best to think of the database service address spaces as more or less permanent fixtures. In fact, the OSDI START command can be included in the subsystem parameter file so that the service is always started during OSDI subsystem initialization (normally during system IPL). This will help to ensure that the service is always ready and available for Oracle database startup processing.

You can stop a database service with the OSDI STOP command (described in Appendix A, "OSDI Subsystem Command Reference") or the native OS/390 STOP (or P) command. Stopping a service terminates all of its address spaces. The command takes effect immediately regardless of the operating state of the associated Oracle database instances. If you stop a database service without first performing an Oracle database shutdown via SQL*Plus, then active client requests may be abnormally terminated. The subsequent Oracle database server startup of a database stopped in this fashion will take longer because of the requirement to read log files and perform recovery for transactions that were in progress at the time of termination.

Oracle Database Instance Startup and Shutdown

After the database service is successfully started, you can issue the Oracle database STARTUP command to make the database instance available to applications. Any of several different Oracle database utilities, including SQL*Plus and Recovery Manager, can be used to issue this command. You can execute the utility on the same OS/390 system as the database that you are starting, or you can execute it on a different system, even one that is not OS/390. In the latter case, you must have configured and started the Oracle Net network service, and some special security considerations come into play. These special security considerations are discussed in Chapter 8, "Security Considerations". For simplicity, the balance of this section assumes that you are running SQL*Plus on the same OS/390 image as the database instance that you are managing.

On OS/390, all of the Oracle database utilities can be executed as TSO commands or noninteractively as batch jobs or started task address spaces. Most installations will find it convenient to set up started task procedures or operator-startable jobs for executing Oracle database STARTUP and SHUTDOWN commands so that these functions are accessible to the system operator. For detailed information on executing SQL*Plus on OS/390, refer to the *Oracle9i Enterprise Edition User's Guide for OS/390*.

Before the utility can issue STARTUP, it must establish a connection to the target instance with a CONNECT statement. Special rules apply to this CONNECT because it is processed before the Oracle database data dictionary (where Oracle database userid information resides) is open. Prior to Oracle9*i*, the statement "CONNECT INTERNAL" could be used to make this connection. In Oracle9*i*, the new form "CONNECT / AS SYSDBA" must be used.

Special security processing is performed on OS/390 to authenticate the OS/390 job or the TSO user who makes the connection. This processing uses a SAF-based (RACROUTE) test, discussed in Chapter 8, "Security Considerations". If your installation has enabled this processing, then the userid that runs SQL*Plus must have been granted the proper authority; otherwise, CONNECT / AS SYSDBA will fail, as will the following STARTUP command. Refer to Chapter 8 for more details.

You must ensure that the utility CONNECT statement connects to the correct database service. When you are running the utility on OS/390, several different methods exist to indicate the target service for a CONNECT, all of which utilize the SID associated with the service. These methods are documented in the *Oracle9i Enterprise Edition User's Guide for OS/390*. In most of our examples here, we use the ORA@*sid* DD statement to specify the target service.

When using multiple server address spaces, the STARTUP command must be issued from a session that is connected to the first service address space. There is no explicit mechanism for requesting connection to the first address space, but this condition is always met when there are no other users connected to the service, which is normally the case when you are running STARTUP.

When the STARTUP command is processed, Oracle database initialization takes place in the server as described in *Oracle9i Database Concepts* and in the *Oracle9i Database Administrator's Guide*. The init.ora parameters are read by the utility and are passed to the server, the Oracle server System Global Area (SGA) is allocated in the server address space(s), and the Oracle server background processes are started. (These run as OS/390 tasks in the first server address space.) The Oracle server does not normally issue any messages to the OS/390 system log or console during this process. The utilities generally display progress messages during STARTUP processing. After a normal STARTUP command completes successfully, the Oracle database is available for application connections and processing. The utility that was used to issue the STARTUP command can exit or terminate at this point, leaving the database up.

The following is an example Oracle database startup that has been set up as an OS/390 batch job using SQL*Plus. The target service has the SID "ORA1", which has been specified using the ORA@ dummy DD statement in the job. The init.ora file is a member of a PDS. The STARTUP command has defaulted the MOUNT and OPEN options, and the database will therefore be opened and made usable to applications.

```
//ORASTART JOB 1,'Oracle ORA1 Startup'
//PLUS EXEC PGM=SQLPLUS,PARM='/NOLOG'
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SQLLOGIN DD DUMMY
//ORA@ORA1 DD DUMMY
//INITORA  DD DISP=SHR,DSN=ORAN.ORAV.PARMLIB(ORA1INIT)
//SYSIN    DD *
CONNECT / AS SYSDBA
STARTUP PFILE=/DD/INITORA
EXIT
/*
```

The database is shut down in the normal Oracle database fashion, by connecting with SQL*Plus and issuing the SHUTDOWN command. Shutting down the database does not terminate service address spaces.

Oracle Corporation does not recommend using SHUTDOWN ABORT or STARTUP FORCE with OS/390 database instances. These commands attempt to forcibly terminate all processes that are accessing the instance, which usually is not a desirable action. In situations where other forms of Oracle database shutdown do not appear to be working, or when you are unable to connect to the server to issue a shutdown command, the best course of action is to stop and restart the database service, then proceed with STARTUP.

Other Database Service Commands

Aside from a stop command, three other commands currently can be issued to a running database service: DISPLAY SESSION, DUMP SESSION, and REFRESH FPS. These commands use the OS/390 MODIFY (or F) command mechanism. To issue these commands, enter the OS/390 system command:

F *id*,*command*

where *id* is the service jobname or identifier, and *command* is the command image as described below.

DISPLAY SESSION

The `DISPLAY SESSION` command displays information for active sessions within an ORA8 service. All keywords and values are required.

```
DISPLAY SESSION JOBNAME(job_filter)
```

Abbreviations: `D, SESS, JOB`

`job_filter:`

Specify up to 8 characters. The value may be suffixed with an '*' or may consist of only an '*' to indicate wildcarding.

Examples:

```
F MYORA8,DISPLAY SESSION JOBNAME(JAOTT3)
```

The above command displays information for sessions initiated by clients with a job name of 'JAOTT3'.

```
F MYORA8,D SESS JOB(JAOTT*)
```

The above command displays information for sessions initiated by clients with a job name beginning with 'JAOTT'.

DUMP SESSION

The `DUMP SESSION` command creates a machine-readable dump of the address space, or spaces, that are associated with a given session within an ORA8 service. All keywords and values are required.

```
DUMP SESSION(sessid) DSN(dataset_name)
```

Abbreviations: `SESS`, `DA`

`sessid:`

The Session ID is an identifier of the relevant session in the form of 8 hexadecimal characters. Session IDs may be discovered with the DISPLAY SESSION command.

`dataset_name:`

This identifier is the fully-qualified name of the data set that will contain the dump, and it may be up to 44 characters in length. The data set cannot exist at the time that the DUMP command is entered.

Example:

```
F MYORA8,DUMP SESS(00010010) DSN(OSDI.SESSION.DUMP)
```

The above command dumps the address spaces associated with session '00010010' to a data set of the name 'OSDI.SESSION.DUMP'.

REFRESH FPS

The REFRESH FPS command is used to reload the server file management parameters defined on page 4-8. As during database service startup, the file specified by the ORA$FPS DD statement is read to obtain the parameters. Unlike service startup, any error encountered while processing the file contents will cancel the attempted refresh.

```
REFRESH FPS
```

Abbreviations: `REFR`

Example:

```
F MYORA8,REFR FPS
```

6 Database Backup and Recovery

This chapter provides OS/390-specific details on Oracle features and techniques that are used to ensure database availability and correctness.

The following topics are included:

Overview

Before planning or attempting database backup or recovery, you should be familiar with the organization of an Oracle database as described in *Oracle9i Database Concepts*, with common database administration procedures covered in the *Oracle9i Database Administrator's Guide*, and with basic Oracle backup and recovery practices documented in *Oracle9i User-Managed Backup and Recovery Guide*.

If you decide to use Oracle Recovery Manager (RMAN) for database backup and recovery, which is highly recommended, read the *Oracle9i Recovery Manager User's Guide*. Finally, if you plan to implement a standby database, refer to *Oracle9i Data Guard Concepts and Administration* before reading the related material in this chapter.

Logging and Recovery

Recovery is the process of applying or reapplying database changes to reflect transactions which have been committed. Critical to the recovery process is the Oracle database redo log (or online log) of database changes that is written by the server as transactions are processed. As each log file or log file group is filled, the Oracle database switches to the next file or group in turn. When you run the database in NOARCHIVELOG mode, the process eventually wraps back around to the first log file or group, overlaying the data in that file. Most production Oracle databases run in ARCHIVELOG mode, which requires that filled logs be archived (copied to another file, called an **archive log**) before they can be reused. The archive logs are opened and read by the server in certain recovery situations.

Two circumstances can occur in which an Oracle database performs recovery operations to update the database to a correct state. One circumstance occurs during Oracle database server normal startup and occurs automatically if the database instance was not shut down cleanly with an Oracle SHUTDOWN command. (This includes cases in which the entire system failed and cases in which you bring an Oracle database down by stopping the OSDI service without performing an Oracle database shutdown.) Except in very unusual circumstances, the recovery obtains all required data from the online logs and rollback segments. It does not involve backups or archive logs and does not require any special action on your part.

The other circumstance occurs when media recovery is required after restoring all or part of the database from backups. This usually is a consequence of losing all or part of a database to physical (hardware) failure or to some kind of logical (software-induced) corruption. This situation involves action on your part, including taking regular backups of your database. During media recovery, the

Oracle server must read all log data written since the backup (which you restored) was taken. In most media recovery situations, archived redo logs, as well as online redo logs, will be required to complete the recovery.

Note: This means that media recovery is generally available only for databases that run in ARCHIVELOG mode.

The work of archiving logs is always done in the Oracle database server, which means that the database instance must be started for logs to be archived. You have two choices for scheduling log archive operations: manual or automatic. If you choose manual archiving, then you run SQL*Plus and issue an ARCHIVE LOG command to cause the server to copy a filled log to an archive. Manual archiving gives you direct control over when archiving occurs, but it has this drawback: if you fail to archive your filled logs and all logs fill up, then the Oracle server suspends update transactions until an archive occurs.

The alternative to manual archiving is automatic archiving, usually indicated by specifying LOG_ARCHIVE_START=TRUE in the init.ora parameter file. With this option, the Oracle server runs an additional background process named ARCH, and that process automatically archives logs as soon as they are filled. As you might expect, most production Oracle databases utilize automatic archiving.

On OS/390, archive logs are VSAM linear data sets (LDS) similar to the other database files. Unlike control, database, and online log files, they cannot be pre-allocated and are always created by the server via invocation of the IDCAMS utility. This means that the server file parameters governing creation of archive logs (the DBAL file group specified in the ORA$FPS parameter file) are particularly important: these parameters must be specified so that archive logs can be created readily as needed. For additional information on the ORA$FPS parameter file, refer to "Server File Management Parameters" on page 4-8.

The naming of archive log data sets is controlled by several init.ora parameters whose OS/390-specific details are discussed in "Oracle Initialization Parameter Considerations" on page 3-15 in Chapter 3, "Configuring a Database Service and Creating a New Database". You can specify that multiple archive copies are to be created in order to reduce the chances of losing an archive to media failure. The parameter conventions require you to supply one or more leading (left-hand) data set name prefixes, one for each copy. When you elect to create multiple copies, the respective prefixes must differ to avoid duplicate data set names, for example, "ORA1.ARCHLOG1." and "ORA1.ARCHLOG2.". A single additional parameter supplies a template for a data set name suffix that is used on all copies. Normally this suffix contains substitution symbols that are replaced with the logical log thread and sequence number, ensuring a unique data set name for each archive.

In addition to normal archiving to a data set, Oracle9*i* will let you archive logs to a standby Oracle database on the same (or a different) OS/390 system. These archives are not used for normal recovery. They are used to update the standby database. For further information refer to *Oracle9i Data Guard Concepts and Administration*.

Backup and Recovery without Recovery Manager

As discussed in the *Oracle9i User-Managed Backup and Recovery Guide*, there are several approaches to backing up and restoring all or parts of an Oracle database independent of the Oracle server, using platform-specific utilities of your choice. On OS/390, this can be accomplished using a fast data mover such as IBM DFSMSdss or a comparable product. The Oracle database server requirements are met by any software that can backup and restore a VSAM linear data set without disturbing the contents or organization of the data. The restore can be to a different disk volume and physical location than was originally backed up, but it must create a valid ICF catalog entry for the cluster with the same high-allocated and high-used RBA characteristics as the original data set.

The key to performing independent backup of Oracle database data is making sure that the correct group of VSAM linear data sets is backed up as a logical set. Which data sets should be copied during backup depends on the type of backup you are doing. The various types of backups and the Oracle database files which comprise them are discussed in the *Oracle9i User-Managed Backup and Recovery Guide*.

One of the most common forms of independent backup is a tablespace backup, in which all of the files that make up a given tablespace are backed up as a set. This kind of backup can be performed while the Oracle server is running and has the tablespace online and in use. It therefore does not interfere with database availability and has little impact on application performance. When done in this fashion (called a "hot backup"), you must notify the Oracle database server of your actions immediately before and after the backup. This notification is done using SQL ALTER TABLESPACE statements. The easiest way to accomplish this is to implement the backup as a 3-step batch job in which the first and last steps execute SQL*Plus to issue the requisite SQL, and the middle step executes the data move. The data move step should be conditioned on successful execution of the first step, as shown in the following example:

```
//ORABKTS1 JOB 1,'Oracle Backup'
//PLUS1    EXEC PGM=SQLPLUS,PARM='/NOLOG',REGION=4M
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//ORA@ORA1 DD DUMMY
//SYSIN    DD *
WHENEVER SQLERROR EXIT FAILURE
WHENEVER OSERROR EXIT FAILURE
CONNECT / AS SYSDBA
ALTER TABLESPACE ARREQ1 BEGIN BACKUP;
EXIT
/*
//BACKUP   EXEC PGM=ADRDSSU,COND=(0,NE,PLUS1)
//SYSPRINT DD SYSOUT=*
//BKUPDS   DD DISP=(,CATLG,DELETE),
//  DSN=ORAPROD.ORADB1.ARREQ1.BKF,
//  UNIT=(TAPE,,DEFER)
//SYSIN    DD *
 DUMP DATASET(INCLUDE(ORAPROD.ORADB1.ARREQ1.DBF*)) -
  OUTDD(BKUPDS)
/*
//PLUS2    EXEC PGM=SQLPLUS,PARM='/NOLOG',REGION=4M,COND=EVEN
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//ORA@ORA1 DD DUMMY
//SYSIN    DD *
CONNECT / AS SYSDBA
ALTER TABLESPACE ARREQ1 END BACKUP;
EXIT
/*
```

In a recovery situation (following a media failure, for example), the Oracle server must not be accessing the logical structure that is being recovered. In the case of tablespace or individual datafile recovery not involving the SYSTEM tablespace, the server can be up and running, but the affected tablespace must be offline. For recovery of the SYSTEM tablespace or for full database recovery, the Oracle database instance must be shut down. You should restore all data sets comprising the entity that you are recovering, for example: all tablespace files for a tablespace that is to be recovered. Normally, you will restore using the original data set names of the files. If you change the data set names during restore, you will have to start the Oracle database server without opening the database (using STARTUP MOUNT) and issue ALTER DATABASE RENAME statements to update the Oracle database control file with the new names.

When all restores are completed and any required renaming has been done, you are ready to begin recovery as described in the *Oracle9i User-Managed Backup and Recovery Guide*. During recovery, the Oracle server will open and read all log data that has been written since the backup (that you restored) was taken. Usually this involves both archive logs and online logs. All of the required archive logs must be available during recovery. If you allow archive logs to be migrated by a product such as IBM DFSMShsm, then you must recall them to disk before Oracle attempts to open them.

Recovery Manager on OS/390

The Recovery Manager (RMAN) utility automates or facilitates various aspects of both backup and recovery and reduces the chances of problems due to human error. It also provides capabilities not available with independent utilities, such as block-level incremental backup.

With RMAN, you create scripts which direct backup and restore and/or recovery processing. RMAN can optionally store scripts and data to support backup and/or recovery processing in an Oracle database. The set of tables that RMAN uses are referred to as the **recovery catalog** and should reside in a different database than the one being backed up and recovered. If you do not wish to maintain a separate database to support RMAN, it can operate without a recovery catalog. In this case, it records some of the data that it requires in the control files of the database that is backed up and recovered.

Although RMAN runs as a separate utility (in TSO or batch, or even remotely on another platform), it connects to the database instance that is backed up or restored (called the **target instance**), and the actual data movement is performed by that instance. The RMAN utility is the same on all Oracle database server platforms, but the data movement particulars in an OS/390 server are quite different from other platforms. For a complete description of RMAN, refer to the *Oracle9i Recovery Manager User's Guide*. The following sections discuss OS/390 particulars of RMAN.

External Data Mover

OS/390 provides an External Data Mover (EDM) implementation for performing RMAN-initiated data movement and backup maintenance activity. EDM runs as a separate OS/390 address space, freeing the Oracle server address space from system resource conflicts (such as tape device allocation and mounting) and from the processing demands of bulk data movement that are associated with backup and restore operations. Priorities or WLM (Workload Manger) goals of the EDM address spaces can be set as desired, separate from those of Oracle server address spaces.

Starting and stopping EDM is done automatically by the Oracle server that hosts an RMAN session. Each RMAN ALLOCATE CHANNEL command starts a separate EDM address space on the same OS/390 system as the host server. The EDM address spaces are system address spaces (similar to started tasks). Each EDM address space executes the Oracle EDM program ORAEDM.

RMAN backup, restore, and maintenance requests that are directed to the server are routed to EDM for processing by using cross-memory communication. Each EDM channel can read or write a single copy of a single backup piece at one time. If multiple channels are allocated in one RMAN script, then multiple EDM address spaces are started when the script runs. If the backup or restore involves multiple backup pieces or copies, then RMAN distributes the operations over the channels, which improves concurrency. The EDM address spaces persist until RMAN deallocates the corresponding channel, until the RMAN session terminates, or until the Oracle server is shut down. It also is possible to terminate an EDM address space with an OS/390 CANCEL command, if necessary.

RMAN ALLOCATE CHANNEL commands must specify "EDM0" as the channel type to start an EDM address space. The only other RMAN channel type allowed with an OS/390 server is DISK. Allocating a DISK channel does not start an EDM address space. A DISK channel can create and read only VSAM Linear Data Set (LDS) backups on direct access storage devices, and the processing occurs entirely within the Oracle server address space. An EDM channel can process backups that are physical sequential (DSORG=PS) data sets on either tape or direct access storage.

The RMAN channel type "SBT_TAPE" that is used on other platforms is not supported on OS/390.

Preparing to Use RMAN with an OS/390 Server

Before you can use RMAN to perform EDM-type operations on an OS/390 system, you must set up a JCL procedure for running EDM address spaces. This procedure must be installed in a system procedure library. The JCL EXEC and DD statement requirements for this procedure are discussed in the section "EDM JCL and Parameters" below. You can give the procedure any name that you desire, as long as it does not conflict with another procedure or with any subsystem that is defined to OS/390. You will need to know the name when you code an RMAN ALLOCATE CHANNEL command.

You might need to discuss EDM security requirements with your system security administrator. The EDM address spaces must be able to create and open backup data sets using data set names that you specify as part of an RMAN script. This might dictate taking actions to associate an OS/390 authorization id with the EDM procedure.

Providing a Snapshot Control File

Certain RMAN synchronization functions require a snapshot control file, which is basically a copy of the Oracle database control file that you create with an ALTER DATABASE statement. By default, an OS/390 Oracle server expects this file to be specified via a SNAPCF DD statement in the server region JCL. Of course, you cannot supply this DD statement until the associated OS/390 data set exists, so if you want to rely on the default snapshot identification, then you must go through a 2-step process as described here. (Alternatively, you can use the SET SNAPSHOT CONTROLFILE NAME command in an RMAN session to specify the snapshot file by its data set name rather than by a DD name.)

To create the snapshot control file, start up the Oracle server and issue an ALTER DATABASE command similar to the following:

```
ALTER DATABASE BACKUP CONTROL FILE TO 'oran.orav.SNAP.CTL';
```

You can use any valid data set name (inside the single quotes) that conforms to the standards of your installation. In most cases, you will want to use the same high-level qualifiers that you use for other files that are part of this database. After this statement completes successfully, you will have created a snapshot control file with the specified data set name. Note that the file management parameters for the DBCT group (specified via the ORA$FPS DD statement) affect the way that this data set is created by the server. If you wish, you can pre-allocate the snapshot control file with your own invocation of IDCAMS. If you do so, make it the same size as your existing control file and include the word REUSE after the quoted name in the ALTER DATABASE statement.

Now shut down the Oracle database server, stop the associated OSDI service, and add a SNAPCF DD statement (similar to the following) to the service JCL procedure:

```
//SNAPCF DD DISP=SHR,DSN=oran.orav.SNAP.CTL
```

Start the OSDI service and startup the Oracle database server. Now your instance has a snapshot control file with the expected default identification.

Identifying Backups

Backups created with RMAN, using an EDM channel, are cataloged OS/390 physical sequential data sets. A separate data set is produced for each copy of each backup piece. You specify the data set name for each backup using the FORMAT parameter of the RMAN BACKUP command.

Although you can specify a fixed OS/390 data set name for FORMAT, the parameter is designed to be used as a template: various substitution variables (identified by a "%" prefix) can be included in the FORMAT string, and these are replaced with specific values each time the RMAN script executes.

You can use any of the substitution variables shown in Table 6–1 as portions of a template data set name in order to form distinct, valid OS/390 data set names:

Table 6–1 FORMAT Parameter Substitution Variables

VARIABLE	DESCRIPTION
%c	specifies the copy number of the backup piece within a set of duplexed backup pieces. If you did not issue the set duplex command, then this variable will be 1. If you issued set duplex, the variable identifies the copy number: 1, 2, 3, or 4.

Table 6–1 FORMAT Parameter Substitution Variables

VARIABLE	DESCRIPTION
%d	specifies the database name. Note: The database name substitution variables (%d and %n) use the database name specified in the CREATE DATABASE command (or in the init.ora file). The database name can be up to 8 characters long and must start with a letter (or #, $, or @) and only contain letters, numbers, the special characters #, $, @ or the hyphen ('-') in the second and subsequent positions to be valid as a standalone part of a data set name (as in 'ORACLE.BACKUP.%d'). If the database name starts with a number but is otherwise valid and is less than 8 characters long, a letter (or #, $, or @) can be placed in the pattern before the %d (as in 'ORACLE.BACKUP.X%d'). Thus, if the %d will be used in the FORMAT string, care should be taken in choosing the database name in the CREATE DATABASE command to ensure that it is valid in an OS/390 data set name.
%n	specifies the database name, padded on the right with 'X' characters to a total length of 8 characters. For example, if PROD1 is the database name, then PROD1XXX is the padded database name. (See the note under %d, above.)
%p	specifies the backup piece number within the backup set. This value starts at 1 for each backup set and is incremented by 1 as each backup piece is created. The number will never exceed 4 digits (9999 maximum).
%s	specifies the backup set number. This number is a counter in the control file that is incremented for each backup set. The counter value starts at 1 and is unique for the lifetime of the control file. If you restore a backup control file, then duplicate values can result. Also, CREATE CONTROLFILE initializes the counter back to 1. If this value exceeds 9,999,999, only the right 7 digits of the number will be used. The number will be from 1 to 7 digits long.

Table 6–1 FORMAT Parameter Substitution Variables

VARIABLE	DESCRIPTION
%t	specifies the backup set timestamp, which is a 4-byte binary unsigned integer value derived as the number of seconds elapsed since a fixed reference time. The combination of %s and %t can be used to form a unique name for the backup set. The unsigned binary integer is formatted into a string in this format: Tsssssss. The sssssss is a base-32 representation of the number using the letters A-V and the numbers 0-9. The generated value will always be 8 characters in length.
%u	specifies an 8-character name constituted by compressed representations of the backup set number and the time the backup set was created. This will always be 8 characters in length and it will start with a letter so it will be valid within an OS/390 data set name. Note: In Oracle8*i*, the variable was '.*%u' which generated '.Axxxx.Axxxx'. The '.*%u' is still accepted, but now generates '.xxxxxxxx' (just as if '.%u' were specified).
%D	specifies the current day of the month from the Gregorian Calendar in DD format.
%F	specifies that dbid (a numeric value that identifies an Oracle database), year, month, day and sequence be combined so that the generated name is unique and repeatable. The generated format is: Iiiiiiii.Dyymmdd.Sss. The iiiiii is a base-32 representation of the dbid using the letters A-V and the numbers 0-9 and will always be 7 characters. The yy, mm, and dd are the year, month and day, respectively. The ss is a hexadecimal representation of the sequence.
%M	specifies the current month from the Gregorian Calendar in MM format.
%T	stands for year, month and day in the format YYMMDD. This will always be 6 characters in length. The first two characters of year are dropped (2008 becomes 08).
%U	specifies a convenient shorthand for &ORAPREFD..&ORASRVN..ORABKUP.%u.P%p.C%c that guarantees uniqueness in generated backup filenames. If you do not specify a format, RMAN uses %U by default. Note: For more information on &ORAPREFD and &ORASRVN, see Appendix C, "Oracle9i for OS/390 System Symbols".

Table 6–1 FORMAT Parameter Substitution Variables

VARIABLE	DESCRIPTION
%Y	specifies the current year from the Gregorian Calendar in YYYY format.
%%	stands for % (i.e. %%Y is actually the string %Y).

Note: Several of the substitution variables generate numbers only, and should not be used immediately after a period in the FORMAT string, or an invalid data set name will result. For example, %s generates a 1 to 7 digit number. In a FORMAT string, a period and a letter (or #, $, or @) have to precede the %s to make it valid in a data set name (for example, 'ORACLE.BACKUP.S%s').

You can also use Oracle9*i* for OS/390 system symbols (identified by a "&" prefix) to form parts of the data set name. This allows a single FORMAT string to produce distinct backup data set names over multiple pieces and copies, and over multiple uses of the same RMAN script. Refer to Appendix C, "Oracle9i for OS/390 System Symbols", for more information.

When you code the FORMAT parameter, you should combine appropriate high-level data set name qualifiers, %u, other RMAN substitution variables, and Oracle9*i* for OS/390 system symbols to form a complete, meaningful data set name. Both RMAN substitution variables and Oracle9*i* for OS/390 system symbols are translated into their current values as the BACKUP command is processed at the server. The Oracle server saves the names of backups in the RMAN catalog or in the database control file, so you do not need to supply, or even know, the names of backups in order to perform an RMAN RESTORE.

The following example is a more complex FORMAT parameter for an OS/390 server:

```
FORMAT '&ORAPREFD..BKUP.%d.%u.P%p.C%c'
```

where:

&ORAPREFD.	is replaced with the default data set name prefix (a server region parameter). Note that the system symbol, "&ORAPREFD.", has the required period terminator.
%d	is replaced with the Oracle database name.

%u	is replaced by a unique, 8-byte string.
%p	is replaced by the backup piece number.
%c	is replaced by the backup copy number.

A data set name that is produced from this format specification might look as follows:

```
ORA3ADBF.BKUP.ORA3DB.G6ENPJ03.P3.C1
```

The data set name of a successfully-created RMAN backup is always cataloged in the OS/390 system catalog structure. When RMAN is told to delete a backup that is no longer needed, EDM invokes the OS/390 IDCAMS utility and passes it a DELETE command for the data set. If the backup is a disk data set, then DELETE uncatalogs the data set and scratches it from the disk volume. If it is a tape data set, then DELETE only uncatalogs the data set. If the backup was a disk data set, and if it has been migrated by DFSMShsm, then the DELETE command logically deletes the migrated copy without recalling it to disk.

When backups are taken directly to tape, the tape volumes must not be reused as long as the backup remains known to RMAN, in other words, as long as the backup is cataloged. The easiest way to ensure this, if you use OS/390 tape management software, is to specify "**catalog retention**" for tape backups. In this case, the tape management system will not recycle the tape volumes as long as they are associated with a cataloged data set name. How you specify catalog retention for a tape data set depends on the tape management software that you are using. If you are using IBM's DFSMSrmm, catalog retention is specified as a **retention policy**. Such policies can be imposed based on the data set name or on a DFSMS management class. For further information, refer to IBM DFSMSrmm documentation or to the documentation for the tape management software that your system uses.

Backup Allocation Parameters

OS/390-specific parameters for backup data sets are specified indirectly in an RMAN BACKUP command using the POOL parameter. The actual parameters are supplied in the EDM procedure in a file identified by an ORA$FPS DD statement. These parameters are coded the same as those for the database server region, discussed in "Server File Management Parameters" on page 4-8. The ORA$FPS parameter file that you use with EDM will contain different file groups with different parameter settings than those used in the database server.

The file group identifier (in the EDM case) consists of the letter "P" followed by a 3-digit decimal pool number. For example, if your backup script specifies

"POOL 6", EDM searches the ORA$FPS file for parameters associated with file group P006. The POOL parameter defaults to zero, and EDM will therefore search for file group P000 if no POOL is coded in the script.

As with the database server, if the matching group cannot be found, then EDM will use parameters specified for the DFLT (default) group if they are included in the EDM's ORA$FPS file. If no DFLT group exists, then EDM will attempt to create the backup file with only a data set name, disposition, and (possibly) space parameters. This is likely to fail unless your installation uses automatic classification (ACL) logic to set suitable allocation parameters.

Additional details and an example ORA$FPS file are included in the section "EDM JCL and Parameters" below. The full description of ORA$FPS keywords and syntax is in "Database Region JCL" in Chapter 3, "Configuring a Database Service and Creating a New Database".

ALLOCATE CHANNEL Considerations

For an OS/390 target server, the ALLOCATE CHANNEL command must specify the channel TYPE as either DISK or "EDM0". The full quotes must be included when the EDM0 type is used. A DISK channel can be used to create and read only backups or copies that are VSAM Linear Data Sets (LDS). This processing takes place within the target database server address space and does not involve EDM. Allocating a TYPE "EDM0" channel starts an EDM address space that can create and read backups that are physical sequential (DSORG=PS) data sets on disk or tape.

The EDM JCL procedure is identified using the PARMS parameter of ALLOCATE CHANNEL. At minimum, PARMS must specify the 1-character to 8-character JCL procedure name for your EDM JCL procedure. The PARMS string can also include other fields that are valid for an OS/390 START command, including a job identifier, JOBNAME or SUBSYS keywords, and procedure-specific JCL symbolic keywords. This string is case-sensitive and generally must be supplied in all uppercase letters. If apostrophes are required around a JCL symbolic parameter, then they must be doubled inside the outer apostrophes that are part of RMAN's PARMS parameter syntax.

The following example is an ALLOCATE CHANNEL command for an EDM whose JCL procedure, ORAEDM1, includes a procedure-specific keyword parameter FPS. A job identifier (EDMC1) is also included:

```
ALLOCATE CHANNEL C1 TYPE "EDM0" PARMS 'ORAEDM1.EDMC1,FPS=''FPS1''';
```

The PARMS parameter defaults to an empty string. If you omit PARMS on an EDM channel allocation for an OS/390 server, then the command will fail.

If your backup script creates multiple backup pieces or copies, or if a restore script is going to call for multiple pieces or copies, then consider allocating multiple EDM channels to improve concurrency. Each ALLOCATE command can specify the same EDM procedure name. If you specify a job identifier (or the JOBNAME parameter), Oracle Corporation recommends using a distinct identifier or jobname for each EDM channel in the script.

BACKUP Considerations

The BACKUP command parameters with OS/390-specific considerations are FORMAT and POOL. As discussed earlier, FORMAT must specify a string that will produce a valid, distinct OS/390 data set name for each backup piece and copy after RMAN metasymbol and OS/390 system symbol substitutions have been performed. FORMAT can also be specified in the ALLOCATE CHANNEL command, in which case the specified string applies to all backups that are created using the associated channel.

The POOL parameter is used to select backup file creation parameters that are supplied via the ORA$FPS DD statement in the EDM JCL procedure. The 3-digit pool number (padded on the left with zeroes if necessary) is appended to the letter "P" to form the 4-character file group identifier that is looked up in the EDM's ORA$FPS file. If POOL is omitted, it defaults to zero, which means EDM will attempt to locate parameters for file group P000.

Example RMAN Backup Script

```
run {
  allocate channel c1 type "EDM0" parms 'ORAEDM.EDM1';
  allocate channel c2 type "EDM0" parms 'ORAEDM.EDM2';
  backup database format 'ORA1.BKUPDB.%u.P%p.C%c' pool 6;
    }
```

EDM JCL and Parameters

To use EDM, you must have a JCL procedure in a system procedure library.

For example:

```
//EDMPROC  PROC,FPS=EDMFPS
//EDMPROC  EXEC PGM=ORAEDM,REGION=0M
//STEPLIB   DD DISP=SHR,DSN=ORAN.ORAV.AUTHLOAD
//ORA$FPS   DD DISP=SHR,DSN=ORAN.ORAV.PARMLIB(&FPS)
```

```
//SYSPRINT  DD SYSOUT=*
//PXYPRINT  DD SYSOUT-*
```

where:

`PGM=ORAEDM`	identifies the External Data Mover program.
`//STEPLIB`	is normally included. It identifies the Oracle AUTHLOAD data set containing ORAEDM.
`//ORA$FPS`	is recommended. It specifies parameters that allow control over the creation, attributes, and processing of backup data sets by EDM. If omitted, internal and system defaults are used.
`//SYSPRINT`	is required. Normally a JES spool data set, it receives informational and error messages generated by EDM.

Tablespace Point-in-Time Recovery

This section supplements tablespace point-in-time recovery (TSPITR) information in the *Oracle9i User-Managed Backup and Recovery Guide*. The most complicated task involved in TSPITR is configuring the parameters for the auxiliary instance, which is set up as a separate OSDI database service in the same or a different OSDI subsystem.

Assuming that ORA1 is the target database service and that ORA2 is the auxiliary database service, the following are examples of init.ora parameters for the auxiliary instance:

CONTROL_FILES refers to the name of the control file for the auxiliary instance. Because the auxiliary instance does not really contain any user data, one control file is used for simplicity.

DB_FILE_NAME_CONVERT This parameter is used to update the auxiliary instance control file with the location of the auxiliary instance files.

DB_NAME must be set to the same value as the target database name.

LOCK_NAME_SPACE must be set to a unique value such as underscore, followed by the target database name. It allows the auxiliary instance to start up even though it has the same name as the primary database.

LOG_FILE_NAME_CONVERT This parameter is used to update the auxiliary instance control file with the location of the log files.

The other parameters are to be the same as those for the target database.

The control files, database, data files, and online log files for the auxiliary instance can be pre-allocated, or you can rely on the ORA$FPS parameters of the auxiliary service to govern file creation. The auxiliary instance is to be started, but unmounted: (START NOMOUNT).

The init.ora example is as follows:

```
CONTROL_FILES = "//'ORA2.CONTROL1'"
SHARED_POOL_SIZE = 4000000
DB_BLOCK_BUFFERS = 500
DB_FILES = 256
DB_NAME = ORA1
LOG_BUFFER = 65536
LOG_CHECKPOINT_INTERVAL = 3000
OPEN_CURSORS = 120
TRANSACTIONS = 55
SESSIONS = 55
PROCESSES = 50
DML_LOCKS = 220
COMPATIBLE = 8.0.0
NLS_DATE_FORMAT='MON DD YYYY HH24:MI:SS'
LOCK_NAME_SPACE = _ORA1
DB_FILE_NAME_CONVERT=("//'ORA1","//'ORA2")
LOG_FILE_NAME_CONVERT=("//'ORA1","//'ORA2")
```

Note: In order to do meaningful time-based recovery, the init.ora of the target instance also must contain the parameter NLS_DATE_FORMAT='MON DD YYYY HH24:MI:SS'.

Tablespace Point-in-Time Processing Using Recovery Manager

Tablespace point-in-time recovery involves using the Recovery Manager HOST command to run EXP (Oracle Export) and IMP (Oracle Import) modules. EXP and IMP require the opening of SYSIN DDname, and you therefore must use the Recovery Manager CMDFILE parameter to designate Recovery Manager commands instead of SYSIN. For example:

```
//ORARMN    EXEC PGM=RMAN,REGION=6M,PARM='++/DD/SYSPARM'
//STEPLIB   DD DSN=...
//ORA$LIB   DD DSN=...
//BSQ       DD DISP=SHR,DSN=oran.orav.SQL
//SYSERR    DD SYSOUT=*,DCB=(LRECL=132,BLKSIZE=1320,RECFM=VB)
//SYSOUT    DD SYSOUT=*
//ORAPRINT  DD SYSOUT=*
//TNSNAMES  DD DSN=...
//SYSIN     DD DUMMY
//SYSPARM   DD *
TARGET INTERNAL/X@ORA1 CATALOG RMAN/RMAN@ORAR
CMDFILE "MYGROUP.RMANCMD(TBPITSQL)"
/*
```

This case assumes that ORAR is the recovery catalog database tnsname alias, ORA1 is the target database tnsname, and MYGROUP.RMANCMD(TBPITSQL) contains the following:

```
CONNECT AUXILIARY INTERNAL/X@ORA2;
RUN {
ALLOCATE AUXILIARY CHANNEL C1 TYPE DISK;
ALLOCATE CHANNEL C2 TYPE DISK;
RECOVER TABLESPACE USER2  UNTIL TIME 'DEC 29 1998 13:35:00';
}
```

This example is created with the assumption that ORA2 is the auxiliary instance tnsname alias, and that USER2 is the tablespace that you want to recover to a point-in-time. For details about tablespace point-in-time recovery, refer to the *Oracle9i User-Managed Backup and Recovery Guide*.

Oracle9*i* Data Guard

Data Guard and DMON processes use the Oracle database instance name for some operations. Oracle9*i* for OS/390 uses the OSDI database service name as the Oracle database instance name. The OSDI service name is restricted to 6 characters or less.

Note: The term "Oracle Net service name," as it is used in the *Oracle9i Data Guard Concepts and Administration* guide, refers to the Oracle Net TNSNAMES alias name and not the OSDI Net service name.

One way to create a standby database is to use an operating system utility to copy Oracle database files. Two different methods are described below.

Example 1 This method uses DFSMSdss to copy two tablespaces, system and rollback. It works only for standby when the two systems share DASD.

```
//STEP2    EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DASD     DD UNIT=SYSDA, SPACE=(CYL, (300,100)),
//            DISP=(NEW,DELETE,DELETE),DSN=&&TMPNAME
//SYSIN    DD *
    COPY TOL(ENQF) CATALOG -
      OUTDDNAME (DASD) -
      DS(INCLUDE( -
       ORAF.V900.SYSTEM.DB1 -
       ORAF.V900.SYSTEM.RBS -
      )) -
      RENAMEU( -
        (ORAF.V900.SYSTEM.DB1,ORBN.V900.SYSTEM.DB1) -
        (ORAF.V900.SYSTEM.RBS,ORBN.V900.SYSTEM.RBS) -
      )
//*
```

Example 2 This method uses the IDCAMS Export/Import utility to copy the system tablespace. It is useful when the file needs to be transmitted to another system. You can use FTP to send the output file from Export to the remote system and then read it with Import to recreate the file. This method can be used for any database file (control files, log files, and tablespaces).

```
//EXPORTCL EXEC PGM=IDCAMS,REGION=1024K
//SYSPRINT DD SYSOUT=*
//DISKOUT  DD DSN=EXPORT.VSAM,DISP=(,CATLG),
           UNIT=SYSDA,SPACE=(CYL,(50,50),RLSE)
//SYSIN    DD  *
  EXPORT                        -
    ORAF.V900.SYSTEM.DB1        -
     OUTFILE(DISKOUT)          -
     TEMPORARY
/*
```

Now transmit to the remote system the EXPORT.VSAM data set, using a method such as FTP.

```
//STEP01 EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  IMPORT IDS(EXPORT.VSAM) -
  ODS (ORBN.V900.SYSTEM.DB1) -
  OBJECTS -
  ((ORAF.V900.SYSTEM.DB1               -
    NEWNAME(ORBN.V900.SYSTEM.DB1)      -
   ) -
   (ORAF.V900.SYSTEM.DB1.DATA          -
    NEWNANE(ORBN.V900.SYSTEM.DB1.DATA) -
   ) -
  )
```

7

Oracle9*i* Utilities

This chapter provides OS/390-specific information for running Oracle utilities that are specific to database administration and are therefore not covered in the *Oracle9i Enterprise Edition User's Guide for OS/390*.

The following topics are included:

Overview

The Oracle utilities that are specific to database administration include Recovery Manager (RMAN), the Oracle Password utility (ORAPWD), and the Offline Database Verification utility (DBV). In Oracle9*i*, the functions of Server Manager (SVRMGRL) have been incorporated into SQL*Plus. Refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* for OS/390-specific details on running SQL*Plus, Oracle Export and Import, and other utilities that are used by non-administrative users. Certain OS/390 facilities that are common to all Oracle utilities, such as PARM field processing and FNA, are also covered in the *Oracle9i Enterprise Edition User's Guide for OS/390*.

General Considerations

Each of the Oracle utilities can be invoked as a TSO command, as a batch or started task jobstep, or via TSO CALL. When invoked as a TSO command, parameters are specified on the command line in the usual manner for TSO command processors. For batch or CALL invocation, parameters are supplied within the PARM string. When usage requires more parameter data than will fit in a 100-character PARM, the "++" notation that is described in the *Oracle9i Enterprise Edition User's Guide for OS/390* can be used to specify that PARM data be read from a data set.

The syntax with which OS/390 data sets are specified to Oracle utilities differs from the syntax that is used in the Oracle server (described in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database"). The primary differences include the use of "/DD/" and "/DSN/" as prefixes to indicate whether a DD name or data set name is used and the availability in utilities of FNA. FNA is a facility for manipulating simple (single-level) file names so that they are treated as PDS (Partitioned Data Set) member names or as other OS/390-specific identifiers. Refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* for a complete discussion of file name syntax and FNA processing.

The Oracle installation process will, optionally, install JCL procedures for commonly-used utilities. To facilitate having multiple different release levels of Oracle software, the installation allows you to specify a distinct 2-character suffix that is appended to the basic name of each such JCL procedure. The detailed utility descriptions on the following pages mention the procedure name for each utility that has one. The name is given with lower-case "xx" on the right-hand end, signifying the optional suffix that is specified at installation time. Refer to your installation records to determine whether these procedures were installed and to determine their exact names.

The following DD statement considerations are common to all Oracle utilities on OS/390:

STEPLIB

Specify a STEPLIB DD in a batch job or TSO command invocation if the Oracle utility that is being invoked is not in a linklist library or OS/390 LPA. Usually STEPLIB designates the Oracle CMDLOAD or AUTHLOAD data set that was created during installation.

ORA$LIB

Specify an ORA$LIB DD statement if your STEPLIB or linklist does not contain the modules that were dynamically loaded by Oracle code, including the Oracle program interface and message and NLS data modules that are used by Oracle localization features.

SYSERR

Required in a non-TSO environment, SYSERR is a sequential output file to which runtime environment error messages are written. Usually this is specified as a SYSOUT data set. In a TSO environment, if no SYSERR DD is allocated to the session, runtime error messages are displayed at the terminal.

SYSOUT

Required in a non-TSO environment, SYSOUT is a sequential output file to which normal tool or utility messages and other output are written. Usually this is specified as a SYSOUT data set. In a TSO environment, if no SYSOUT DD is allocated to the session, then normal output is displayed at the terminal.

SYSIN

Required in a non-TSO environment, SYSIN is a sequential input file which the tool or utility reads for input commands or data. Usually this is specified as an instream (DD *) data set or as a member of a PDS. When creating or editing a PDS member that will be used as SYSIN for an Oracle tool or utility, do not allow the editor to put sequence numbers or other nonblank data into the rightmost positions of each record. In a TSO environment, if no SYSIN DD is allocated to the session, tool, or utility, then input is read from the terminal.

Some utilities receive all of their input from the command line or JCL PARM field. A batch execution of such utilities must include a SYSIN DD statement, but it can be coded as DUMMY.

SQL*Plus on OS/390

In Oracle9*i*, SQL*Plus is used for most database administration tasks, including Oracle startup and shutdown. All of the functions that were previously provided in line-mode Server Manager are available in SQL*Plus.

A version of SQL*Plus for the OS/390 Unix System Services (USS) environment is also supplied. You can invoke this version of SQL*Plus in a USS shell environment and utilize input and output files that are in the USS Hierarchical File System (HFS).

Considerations for using Oracle tools and utilities, including SQL*Plus, are discussed in the *Oracle9i Enterprise Edition User's Guide for OS/390*.

Recovery Manager (RMAN) on OS/390

Oracle9*i* Recovery Manager can be invoked in batch or TSO using the name RMAN. If you installed the Oracle JCL procedures, then you can execute the ORARMNxx procedure to invoke RMAN.

RMAN has some special processing requirements. First, it must be able to read the recover.bsq script during its initialization. (This script is member RECOVER of the SQL data set created during Oracle installation.) RMAN expects a BSQ DD statement that specifies the SQL data set but no member name, as in the following:

```
//BSQ DD DISP=SHR,DSN=ORAN.ORAV.SQL
```

If you use the ORARMNxx procedure, this DD statement is already included.

Depending on how it is used, RMAN may need to connect to as many as three distinct Oracle database instances: one for its catalog (the "catalog instance"), one for the database that is being backed up or recovered (the "target instance"), and, during certain types of point-in-time recovery, an "auxiliary instance" that participates in recovery processing.

The requirement to connect to multiple instances indicates that you cannot rely entirely on one of the singular mechanisms (the ORA@*sid* DD statement or the ORACLE_SID or TWO_TASK environment variables) to specify the instance. You can use one of those mechanisms for any one of your RMAN connections, but the other connection(s), if required, must use a tnsnames.ora file or explicit Oracle Net address strings. Oracle Corporation recommends using a tnsnames.ora file, which is specified via a TNSNAMES DD statement on OS/390. Refer to the *Oracle9i Net Services Book Set* and to Chapter 10, "Oracle Net", for a discussion of this file.

The RMAN CONNECT statements that do not rely on ORA@*sid*, ORACLE_SID, or TWO_TASK will need to supply the tnsnames.ora identifier for the instance. In the example batch RMAN job which follows, we have used ORA@*sid* to access the catalog instance at SID 'ORMC' and have used a tnsnames.ora identifier to access the target instance at SID 'ORA1'. Only the RMAN CONNECT statements are shown.

```
//ORARMAN  JOB 1,'Oracle Recovery Mgr'
//RMAN     EXEC PGM=RMAN
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//BSQ      DD DISP=SHR,DSN=ORAN.ORAV.SQL
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//ORA@ORMC DD DUMMY
//TNSNAMES DD *
DBORA1=(DESCRIPTION=(ADDRESS=(PROTOCOL=XM)(SID=ORA1)))
/*
//SYSIN    DD *
connect catalog rman/rman
connect target /@DBORA1
...
/*
```

On OS/390, RMAN sets return code zero if all input statements are processed without error. If any errors occur, a return code of 8 is produced.

Oracle Password Utility (ORAPWD) on OS/390

The Oracle password utility, ORAPWD, is used to initialize a password file that the database server uses to validate certain types of Oracle logon. Usage considerations for a password file (which is optional) are discussed in Chapter 8, "Security Considerations" and in the *Oracle9i Database Administrator's Guide*.

ORAPWD can be run in batch or as a TSO command. Because it is used infrequently, no JCL procedure is provided. The password file must be pre-allocated as a VSAM LDS prior to executing ORAPWD. The IDCAMS DEFINE CLUSTER considerations for this file are exactly the same as those for Oracle database files, discussed in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database". Refer to the *Oracle9i Database Administrator's Guide* for information on sizing this file.

All input to ORAPWD comes from the PARM field or TSO command line parameters. A SYSIN DD statement must be supplied, but it can be coded as DUMMY. When you specify the FILE= parameter to ORAPWD, use only the data set name of the VSAM LDS. Do not include apostrophes or any "/DSN/" prefix. The OS/390 userid that is associated with the batch job or TSO session must have update authority on the data set. Following is an example batch job that creates the password data set using IDCAMS and then initializes the password data set using ORAPWD.

```
//ORAPW    JOB 1,'Oracle Administration'
//AMS      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
 DEFINE CLUSTER(NAME(ORAPROD.ORADB1.PWD) LINEAR -
  RECORDS(16) STORAGECLASS(PRODSC1) MANAGEMENTCLASS(PRODMC1))
/*
//OPW      EXEC PGM=ORAPWD,COND=(0,NE),
//  PARM='FILE=ORAPROD.ORADB1.PWD PASSWORD=SECRET3 ENTRIES=32'
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//
```

On OS/390, ORAPWD sets return code zero if the file is initialized without errors. If any errors occur, a nonzero return code is produced.

Offline Database Verification Utility (DBV) on OS/390

DBV (Database Verification Utility) examines the physical and logical structure of an offline Oracle database file or a (backup) copy of a database file. General considerations for using DBV are discussed in the *Oracle9i User-Managed Backup and Recovery Guide*.

DBV can be run as a TSO command or as a batch job. Because it is used infrequently, no JCL procedure is provided. All input to the utility is via command line parameters or the PARM field. A SYSIN DD statement is required, but it can be coded as DUMMY. The FILE= parameter for DBV must be specified as a DD name with a "/DD/" prefix, as shown in the following example. Do not use the BLOCKSIZE and FEEDBACK parameters on OS/390.

```
//ORADBV  JOB 1,'Oracle DBVerify'
//DBV     EXEC PGM=DBV,
//  PARM='FILE=/DD/DBFILE START=1 END=50'
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//DBFILE   DD DISP=SHR,DSN=ORAPROD.ORADB1.SYSTEM.BKUP.DBF1
//SYSERR   DD SYSOUT=*
//SYSOUT   DD SYSOUT=*
//SYSIN    DD DUMMY
//
```

On OS/390, DBV produces a zero return code if processing was successful and if no logical or physical errors were detected. Otherwise, return code 8 is produced.

8 Security Considerations

This chapter describes post-installation and operational OS/390 security issues in Oracle9*i*, Oracle Net, and associated products and features on OS/390. The information is presented with the assumption that IBM OS/390 Security Server (RACF) is being used, but any OS/390 product that fully implements IBM's System Authorization Facility (SAF) can be used. If your installation does not use RACF, please refer to your security product's documentation for matters presented here in RACF terms.

The following topics are included:

Overview

Oracle products and OSDI interface in a number of places with native OS/390 security features. Some of the resulting interactions are discussed as installation topics in the *Oracle9i Enterprise Edition Installation Guide for OS/390*. These topics include RACF resource class considerations, Program Properties and APF authorization requirements for Oracle product modules, and requirements for associating OS/390 userids with OSDI-defined services.

Controlling Access to OSDI Subsystem Commands

OSDI subsystem command processing includes an authorization check to confirm that the console or the user is allowed to issue the command. You control access to commands by defining resource profiles to the security subsystem and then by granting access (for specific consoles and users) to those resources. If you do not define resource profiles, then the authorization check returns a "resource unknown" indication to OSDI, and OSDI then allows the command to be processed. Thus, the default behavior (in the absence of any profile definitions) is that any command is allowed from any source. This is not chaotic, as it may sound, because access to command-issuing mechanisms themselves (such as consoles) is usually controlled in most OS/390 installations. Base your decision to define profiles and to activate the authorization mechanism upon the security standards and procedures at your installation.

The resource profiles that are used to protect commands should be defined in the resource class that you chose when installing Oracle software, as discussed in *Oracle9i Enterprise Edition Installation Guide for OS/390*. If you elected to accept the default, then this will be the FACILITY class. Otherwise it will be a class name that you chose and configured for your SAF-compliant security software.

The command authorization resource names are of the form:

```
ssn.cmdverb
```

where *`ssn`* is the OSDI subsystem name, and *`cmdverb`* is the full-length OSDI command verb. (Verb abbreviations, such as DEF for DEFINE, must not be used in the resource name.)

The level of authorization that must be granted to users or to consoles in order to enable commands depends on the command. The following table lists all of the command verbs and the authorization level required for each:

Verb	Authorization Level
DEFINE	Update
ALTER	Control
SHOW	Read
START	Read
DISPLAY	Read
DRAIN	Read
RESUME	Read
STOP	Read

Controlling Access to OSDI Services

OSDI bind processing, which establishes connections between OS/390 address spaces, performs an authorization check to confirm that the binding address space (the "client") is allowed to access the target service. The target service can be an Oracle9*i* instance or an Oracle Net network service running on OS/390. The possible client address spaces are:

- a TSO or batch address space running an Oracle tool or an Oracle utility or a customer-written Oracle application
- a CICS or IMS region running transactions which access an Oracle database
- a local Oracle database instance that is accessing another local or remote Oracle instance via a database link
- the Oracle network service accessing a local Oracle instance on behalf of remote (inbound) client applications or database links

The bind authorization check applies in all of these cases. In order for the check for a given target service to be meaningful, resource profiles must be defined to a SAF-compliant security server such as RACF. The profile names incorporate the OSDI service name so that access to each service is separately controlled. When profiles are defined, the OS/390 userid that is associated with the client address space must have READ authorization on the target service's profile in order for the bind to be allowed. Two profiles are defined for each service: one for normal application binds (used by TSO and batch Oracle tools or applications) and one for managed binds (used by CICS and IMS, and by the Oracle server and Oracle Net when operating as **clients** as described above).

If you do not define resource profiles for a service, then all binds from all address spaces are permitted. Oracle Corporation recommends that you define resource profiles for all services so that bind access is controlled via standard OS/390 security mechanisms.

The resource profiles that are used to protect binds should be defined in the resource class that you chose when installing Oracle software, as discussed in *Oracle9i Enterprise Edition Installation Guide for OS/390*. If you elected to accept the default, then this will be the FACILITY class. Otherwise it will be a class name that you chose and configured for your SAF-compliant security software.

- The name structure for the normal application bind resource profile is:

  ```
  ssn.service.UBIND
  ```

  where *`ssn`* is the OSDI subsystem name, *`service`* is the target service name, and `UBIND` is a constant indicating application binds.

- The name structure for the managed binds used by CICS, IMS, Oracle database links, and Oracle Net is:

  ```
  ssn.service.ABIND
  ```

  where *`ssn`* is the OSDI subsystem name, *`service`* is the target service name, and `ABIND` is a constant indicating managed binds.

Note: In addition to subsystem-level authentication of binds, an OSDI database instance uses SAF to control access to the Oracle database server's SYSOPER and SYSDBA system privileges. This mechanism is discussed in the following section.

Controlling Access to Database SYSDBA and SYSOPER Privileges

SYSOPER and SYSDBA are access privileges that are associated with an Oracle database instance. A database session with these privileges can perform operating functions such as starting up and shutting down the database and can perform DBA activities such as managing database and tablespace definitions. The privileges can be requested by including "AS SYSOPER" or "AS SYSDBA" in a CONNECT statement issued to a tool such as SQL*Plus.

For local clients connecting from TSO, batch, or USS address spaces via cross-memory, access to the SYSDBA and SYSOPER privileges is controlled using SAF-defined resources on OS/390. These must be defined in the same resource class that you use for binds, as discussed in the previous section. You chose this class when installing Oracle software, as discussed in the *Oracle9i Enterprise Edition Installation Guide for OS/390*. If you elected to accept the default during installation, then this will be the FACILITY class. Otherwise, it will be a class name that you chose and configured for your SAF-compliant security software. The form of the resource names is:

```
ssn.service.OPER
ssn.service.DBA
```

where:

ssn	is the OSDI subsystem name
service	is the database service name
`OPER`	is a suffix to be entered exactly as shown
`DBA`	is a suffix to be entered exactly as shown

After the resources are defined, granting "read" authorization on a resource to a given OS/390 userid allows a job or TSO session with that userid to connect to the database with the given privilege. You must grant read authority on the OPER resource to the OS/390 userids that will startup and shutdown the database.

If you do not define these resource names for a given database instance, then any userid is allowed to connect with SYSOPER or SYSDBA privileges. Oracle Corporation recommends that you define these resources so that the use of Oracle database system privileges is controlled.

If you want remote clients to have access to SYSDBA and SYSOPER privileges, you must create an Oracle password file using the ORAPWD utility described in Chapter 7, "Oracle9i Utilities". SAF authorization cannot be used for remote clients because the user's OS userid cannot be verified and might not even be compatible with SAF expectations.

A separate file is used to provide password verification for remote users because the database may not be mounted or open when the connection is requested. (This occurs when the remote user connects in order to issue STARTUP, for example.)

To use an Oracle password file to control remote client access to SYSOPER and SYSDBA privileges, follow these steps:

1. Create and initialize a password file as described in the section "Oracle Password Utility (ORAPWD) on OS/390" on page 7-6.
2. Add an ORAPASSW DD statement to the database service region JCL procedure. Specify the dsname of the VSAM linear data set created in step 1, and specify DISP=SHR.
3. Shut down the Oracle instance and stop the associated service.
4. Set the REMOTE_LOGIN_PASSWORD_FILE parameter to EXCLUSIVE in the instance's init.ora file.
5. Restart the database service (with the updated JCL procedure) and startup the Oracle instance.

Database Service Actions Subject to OS/390 Authorization

When a database service runs on OS/390, some of its interactions with the operating system may be subject to authorization checks. These checks are performed by the operating system, not by Oracle software, and generally are based on the OS/390 userid associated with the service address space. The *Oracle9i Enterprise Edition Installation Guide for OS/390* describes the OS/390 mechanisms that are used to associate a particular userid with a service address space. This section describes the actions that the Oracle server and the OSDI infrastructure take that might be subject to authorization checks on your system. It is your responsibility to make sure that an OS/390 userid is associated with a database service, if necessary, and that it has the correct authorizations for the functions it must perform.

Data Set Creation and Deletion

The Oracle server can invoke the OS/390 IDCAMS utility to create and to delete VSAM linear data set (LDS) files. Details regarding when this occurs and how you control it are provided in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database". If your server will be creating or deleting files, make sure that the associated OS/390 userid has the necessary authorization to do so with the data set name structure that you are using.

Data Set Open

The Oracle server performs an update-type open on the VSAM LDS files comprising the database. It also opens the alert log and other diagnostic logs for output and opens various parameter and SQL files (such as the SQLBSQ and

ORA$FPS DDs) for input. The associated OS/390 userid must have the required authorization for these opens.

OSDI Bind Authorization

Database links are Oracle's mechanism for distributed database access. When an Oracle application uses a database link, a connection is made from one Oracle database instance to another. When this happens on OS/390, an OSDI bind is issued from the first instance to the second (if the second instance is on the same OS/390 system) or from the first instance to an Oracle Net network service (if the second Oracle instance is remote). If you are using OSDI bind authorization checking as described previously in the section "Controlling Access to OSDI Services", the OS/390 userid that is associated with the first instance must be authorized to bind to the second instance or to Oracle Net.

Unix System Services Access

Certain Oracle9*i* features (such as the UTL_FILE and UTL_HTTP packages, Oracle9*i* JVM, and the Oracle external table feature) use OS/390 Unix System Services (USS). To use USS, the database service must have an associated OS/390 userid, and that userid must have a default OMVS segment defined to the security system. If either of these conditions is not met, then message MIR0110W is issued during service address space initialization, and the features which rely on USS are inoperative in that Oracle instance.

External Data Mover Actions Subject to OS/390 Authorization

When you use Oracle9*i* Recovery Manager (RMAN) to perform backup or recovery actions on an OS/390 server, one or more separate External Data Mover (EDM) address spaces may be started to perform data movement and backup management tasks. The details of this activity are in Chapter 6, "Database Backup and Recovery". Although it is not defined as an OSDI service, the EDM runs as a system address space and can have an associated OS/390 userid via the same mechanisms as OSDI services.

The EDM address space must have the necessary authorization to open sequential backup file data sets for output (during backup) or for input (during recovery). Certain RMAN backup maintenance activities cause the EDM to delete backup data sets via an IDCAMS DELETE command, so the EDM that is used during backup maintenance should have that authorization as well.

Note: Be aware that for backup and restore operations, the EDM address space does **not** open or access the associated Oracle database (VSAM LDS) files. Those files are accessed only in the Oracle server address space.

Oracle Net Actions Subject to OS/390 Authorization

The Oracle Net network service JCL procedure name must have an associated OS/390 userid. The associated OS/390 userid must have an OMVS RACF segment (or equivalent, if a product other than RACF is used) if the installation is not using a default OMVS segment.

Authorizing Oracle Logon

When an Oracle userid is defined as IDENTIFIED EXTERNALLY it means the user is authenticated by operating system facilities rather than by Oracle. On OS/390 this mechanism works in one of two ways:

- The user logs on to the operating system and is authenticated by normal operating system security. When the user runs an Oracle tool or application, it logs on to Oracle with "/" (a single forward slash) instead of specifying an Oracle userid and password. Oracle takes the user's OS/390 userid as the Oracle userid (possibly with a prefix, specified as the OS_AUTHENT_PREFIX init.ora parameter). This means the user can access Oracle only with OS/390 userids for which the password is known. This technique is normally used only when the user is running on the same OS/390 system as the Oracle server being accessed. Although it can be enabled for use over Oracle Net, doing so is not considered secure because the server cannot guarantee that the associated userid was authenticated.
- The user runs an Oracle tool or application that logs on to the Oracle server with an explicit userid/password. The verification of the userid and password is controlled by the LOGON_AUTH database region parameter, which can specify one of three verification choices:

No SAF check If a user defined to Oracle as IDENTIFIED EXTERNALLY attempts to logon with an explicit userid/password, the logon is rejected.

Built-in SAF check This built-in SAF check verifies that the userid and password that are provided on the logon are valid. The user need not be logged on to OS/390 with the same userid and in fact may be running on a non-OS/390 platform (connecting via Oracle Net). The Oracle userid must be defined to the OS/390 security subsystem and the given password must be correct for the logon to succeed. A `RACROUTE VERIFY` function is performed by the OSDI code, but no logon exit is called in this case. SAF interfaces with any security manager that you have installed (IBM's RACF, CA's Top Secret or ACF2, an internally developed security manager, or another third-party security manager). This SAF check is designed to work with virtually any security manager and eliminates the need for an external logon exit, unless you want to modify the normal `RACROUTE VERIFY` of a userid and password.

External logon exit This method calls an external, dynamically loaded logon exit. Oracle Corporation supplies a sample logon exit (which does what the built-in SAF check does), or you can write your own. This exit performs the actions that you specify and then returns success or failure of the validation.

> **Note:** The supplied sample logon exit and the built-in SAF check are functionally equivalent. Unless site-specific changes to the sample logon exit are needed, the built-in SAF check should be used because it is more efficient.

The "Built-in SAF check" and the "External logon exit" are called only for users that are `IDENTIFIED EXTERNALLY`. These checks are not performed for a user who is defined to Oracle (`IDENTIFIED BY <`*`password`*`>`), because these users can be resolved internally within Oracle.

The type of validation done for users that are `IDENTIFIED EXTERNALLY` is indicated in the OSDI service parameters. A single parameter controls this validation. The syntax is:

`LOGON_AUTH (`*`auth`*`)`

where *auth* is:

`NONE` - IDENTIFIED EXTERNALLY not allowed

`SAF` - perform built-in SAF check

`exitname` - call logon exit

The default (if nothing is specified) is `NONE`.

For example:

```
LOGON_AUTH(NONE)
LOGON_AUTH(SAF)
LOGON_AUTH(RACFSMPO)
```

The logon exit must reside in the STEPLIB or JOBLIB concatenation or in the linklist, and it must be in an authorized library.

A sample user logon exit is provided in Oracle SRCLIB library member RACFSMPO. It uses the OS/390 SAF interface to invoke the OS/390 security manager. If the OS/390 security manager does not support the SAF interface, then the calls to RACROUTE in the exit must be replaced with the equivalent calls appropriate for the OS/390 security manager.

The calling sequence for the logon exit uses standard OS/390 assembler calling conventions. R15 is the entry point, R14 is the return address, R13 points to a standard 72 byte save area, and R1 is the address of a parameter list. The parameter list consists of a list of addresses of each parameter (all values are passed by reference, not by value), and the last parameter has the high-order bit set.

When returning, R15 should be set to 0 to indicate a successful verification of the userid and password that were supplied, and should be set to any nonzero value to indicate any type of failure (4 would be an appropriate value).

The exit is called in 31 bit addressing mode, supervisor state, storage protection key 7, and in an authorized address space. The exit will be running in TCB mode with no locks held and with no ARRs, FRRs, or EUT-style FRRs set. The exit is called in primary addressing mode with HASN=PASN=SASN (home, not cross memory mode).

The logon exit should be fully reentrant code.

The parameter list that is passed contains pointers to the following parameters, all of which are input only:

Table 8–1 Input-Only Parameters

Field	Type/Length	Description
work area	char/4k	set to all x’00’ before every call to the exit
userid	char/1+	userid to be validated (number of bytes varies)
userid length	bin/2	length of userid
password	char/1+	password to be validated (number of bytes varies)

Table 8–1 Input-Only Parameters

Field	Type/Length	Description
password length	bin/2	length of password
OS/390 jobname	char/8	OS/390 jobname from JOB card of client
ASID	bin/2	address space id of client address space
OSDI session id	bin/4	a unique OSDI session id
OS username	char/8	the operating system username (batch, TSO, CICS, and IMS only)
terminal name	char/8	terminal name
program name	char/8	program name
RACF group name	char/8	RACF group name
connection type	char/8	connection type (BATCH, TSO, CICS, IMS, TCP/IP, VTAM) to indicate environment of client
JES jobid	char/8	JES job identifier (such as JOB08237)
job card entry time	bin/4	entry (submission) time of job. Binary hundredths of a second since midnight
job card entry date	packed/4	entry (submission) date of job. Packed decimal 0CYYDDDF, where C=0 FOR 19, C=1 FOR 20, YY=year, DDD=day number within the year (Jan 1=1)
job card accounting info	char/145	from jobcard
network data (high bit set in parameter list)	char/2+	variable length NIV data (refer to Chapter 9, "Oracle SMF Data")

The only output of the logon exit is the R15 return code. No other value that is passed in the parameter list should be modified except the first one.

The first parameter is a 4096-byte work area that is set to all `X'00'` before every call to the logon exit. The logon exit can use this storage for anything that it needs. It should not be freed.

The logon exit can do any of the following:

- call a security manager (SAF calls, RACF, Top Secret, ACF2, and so forth)
- get and free storage using the STORAGE macro (the exit must keep track of all acquired storage and must make certain to free it before exiting)
- call SMF to write SMF records
- call WTO to write messages to the console

The logon exit should not do anything that would cause it to wait for any significant period of time (more than one tenth of a second, for example). Avoid opening data sets, writing to the operator with a reply (WTOR), and creating enqueues.

Any resources that are acquired in the logon exit must be freed before it returns. There is no cleanup call made to the logon exit, so any resources that are not released will accumulate in the address space and could eventually cause resource shortages.

9 Oracle SMF Data

The IBM System Management Facility (SMF) provides a facility for users to collect and record a variety of system and job-related information. SMF formats the information into a number of different records. By creating analysis and report routines, installations can use the information in SMF records to track system usage.

The Oracle server uses the standard SMF interface to write user records to the SMF data sets. These user records contain Oracle server accounting and Oracle auditing information allowing Oracle installation sites to charge individual users for the resources they use.

The following topics are included:

Preparing to Record SMF Information

SMF recording is activated by updating the SMFPRMxx member of SYS1.PARMLIB using the SYS or SUBSYS option to allow recording of the Oracle user record type. If the SUBSYS option is used, then the SUBSYS name must match the OSDI subsystem name that hosts the database service. Refer to the IBM guide on System Management Facilities for information about implementing SMF.

Specifying the Oracle Record Type

The default Oracle user record type is 0 (zero). A zero for this parameter indicates that no SMF statistics record is to be written. You can override the default to any value between 128 and 255 by adding the SMF_STAT_RECNO (abbreviation is SMFSTRCN) to the OSDI database region parameter file. The SMF record number that is chosen must not be the same as the number that is used by any other OS/390 software.

Oracle Corporation recommends using SMF record number 204, but any available record number between 128 and 255 may be used.

If this parameter is not specified, or if zero is specified, then no SMF statistics collection or recording is done. This saves some CPU overhead and saves the overhead of the SMF write itself (which is mostly asynchronous work done by the SMF address space, and the in-line overhead is mostly just moving data into SMF buffers).

Using the OSDI SMF_STAT_RECNO Parameter

SMF_STAT_RECNO | SMFSTRCN

SMF_STAT_RECNO can be added as an OSDI parameter to the OSDI database region parameter data set to override the default record number 0. In the following example, 204 is the new SMF record type:

```
SMF_STAT_RECNO(204)
```

Starting SMF Recording of Oracle Records

SMF recording of Oracle accounting information starts automatically at startup if SMF is activated and if the Oracle record number is specified.

Because the standard system default record types activated for SMF are 128 through 255, and because the recommended number for the Oracle server (204) is within this range, many sites automatically begin SMF recording of Oracle records when Oracle is installed, and the SMF_STAT_RECNO parameter is supplied.

The service must be stopped and restarted for this parameter to take effect.

Stopping SMF Recording of Oracle Records

The OSDI SMF_STAT_RECNO parameter can be used to stop SMF recording for Oracle. To stop SMF recording for Oracle regardless of what your system tables specify, use:

```
SMF_STAT_RECNO (0)
```

or take the default of 0. The service must be stopped and restarted for this parameter to take effect.

Events that Generate SMF Records

After SMF recording is turned on, an SMF record is written each time a user logs off (normal termination or SMFINV=SMFNORM), provided SMF was activated when the user logged on.

SMF records are also written on an abend or cancellation of a job if SMFINV=SMFABORT

If the OS/390 system crashes, then SMF records are not written, and the information is lost.

SMF Recording under CICS

When Oracle Access Manager for CICS transactions are used to access data on your local Oracle9*i* for OS/390 server, a single thread can be shared by many CICS users. When SMF recording is activated, an SMF record is written for a single thread when the thread is dropped. However, SMF recording is not supported when these transactions are used to access data on a remote Oracle server. If a thread is defined with PROTECT set to NO, then the thread is dropped after being idle for 30 seconds.

If a thread is defined with PROTECT set to YES, then the thread is dropped when Oracle Access Manager for CICS is stopped with the STOP command. Refer to Chapter 11, "Oracle Access Manager for CICS", for more information on the STOP command.

For Oracle Access Manager for CICS, SMF accounting information is based on the Oracle userid for the CICS transaction.

The following is a sample thread definition table:

```
ORACICS TYPE=THREAD,
AUTH=TRANSID,
PROTECT=NO,
TRANSAC=(PGM1,PGM2)
```

If the above thread definition is used, then a sample session is as follows:

```
PGM1
PGM2
PGM1
```

The thread is dropped after being idle for 30 seconds, and two SMF records are written. One of the records summarizes statistics for all PGM1 transactions, and the other record summarizes statistics for all PGM2 transactions.

Refer to Chapter 11, "Oracle Access Manager for CICS" for instructions for configuring thread definition tables in Oracle Access Manager for CICS.

Interpreting an Oracle SMF Record

To interpret an Oracle SMF record, you first need to dump the SMF data set to a sequential data set. You can then write a program that does all of the following:

- Reads the sequential data set
- Selects only records with the Oracle record number
- Accesses the Oracle SMF record fields using the provided DSECT
- Prints the statistics

A sample program named ORAFMTO is provided in the SRCLIB library that you can customize for your installation. Refer to "Sample Formatting Program for SMF Records" on page 9-10.

The Assembler copy file, ORASMFO, contains DSECTS that map and document the Oracle SMF record fields. The ORASMFO data set member resides in the Oracle SRCLIB library.

The ORASMFO file is divided into these sections:

- The standard record header section, which contains offsets and lengths of the Net and accounting sections

- The correlation section
- The OSDI section
- The database engine section
- The Net section (if applicable), which contains information about the network origin of clients on an Oracle Net for OS/390 TCP/IP protocol network (IBM or SNS/TCPaccess)
- The OS/390 accounting section (if applicable)

Not all sections are present in all SMF records. For example, the OS/390 accounting section is present only in SMF records for batch and TSO users. When a section is present, the SMF record header contains the correct length for that section, which might be release dependent. The length field for non-existent sections contains 0 (zero).

Contents of the SMF Header Section

Table 9–1 contains brief descriptions for the labels in the SMF header section. For a complete layout of the contents of the SMF header section, refer to the DSECT.

Table 9–1 Contents of the SMF Header Section

ORASMF0 Label	Description
SMFHDR	Standard SMF header
SMFHLEN	Total length of SMF record
SMFHSEG	Segment descriptor = 0
SMFHSIN	SYS IND = X’80’ Subsystem info to follow SYS IND = x’40’ Subtype format record
SMFHREC	Record type recommended = 204 (decimal)
SMFHTIM	Timestamp, time binary (0.01 seconds since midnight)
SMFHDAT	Timestamp, date (0cyyydddf) c=0 for 19xx. c=1 for 20xx
SMFHSYS	System id
SMFHSSI	OSDI Subsystem id
SMFHSUB	Record subtype; 1 = accounting record
SMFSRVC	OSDI service name

Table 9–1 Contents of the SMF Header Section (Cont.)

ORASMF0 Label	Description
SMFSESID	OSDI session id
SMFHRSV1	Reserved
SMFNETO	Offset to Net section
SMFACTO	Offset to OS/390 accounting section
SMFHRV2	Reserved
SMFNETL	Length of Net section
SMFACTL	Length of OS/390 accounting section
SMFHRV3	Reserved

Contents of the SMF Correlation Section

Table 9–2 contains brief descriptions for the labels in the SMF correlation section.

Table 9–2 Contents of the SMF Correlation Section

ORASMF0 Label	Description
SMFAUTH	Authorization id = TSO logon id Batch user id on jobcard CICS USERID, TERM-ID,TRANS-ID, PROGRAM-ID, or OPID
SMFCORI	Correlation id = TSO logon id Batch jobname CICS jobname Not valid for Oracle Net
SMFCONN	Connection type (TSO,BATCH,CICS,VTAM,TCP/IP, IMS)
SMFASID	Users address space id (not valid for Oracle Net)
SMFOUSR	Oracle logon id
SMFTNAME	Originating terminal id (if available)
SMFPNAME	Originating program name (if available)
SMFGRPN	RACF group name (if available)
SMFJBID	JES job identifier

Table 9–2 Contents of the SMF Correlation Section (Cont.)

ORASMF0 Label	Description
SMFENTRY	RDR jobcard entry time (batch and TSO only). This field is equivalent to the SMF5RST field in the SMF job termination (type 5) record.
SMFEDATE	RDR jobcard entry date (batch and TSO only). This field is equivalent to the SMF5RSD field in the SMF job termination (type 5) record.

Contents of the SMF OSDI Data Section

Table 9–3 contains brief descriptions for the labels in the SMF OSDI data section.

Table 9–3 Contents of the SMF OSDI Data Section

ORASMF0 Label	Description
SMFTIM	Beginning timestamp, time binary (0.01 second since midnight)
SMFDAT	Beginning timestamp, date (0cyyydddf), ending time and date in header c=0 for 19xx. c=1 for 20xx
SMFDTAI	Data in
SMFDTAO	Data out
SMFXMCPU	Cross memory CPU time (TOD format)
SMFRPCS	RPC count
SMFHWST	High-water mark of storage used
SMFINV	Reason for invocation
SMFNORM	Normal termination
SMFABORT	Clean up done

Contents of the SMF Database Engine Data Section

Table 9–4 contains brief descriptions for the labels in the SMF database engine section.

Table 9–4 Contents of the SMF Database Engine Section

ORASMF0 Label	Description
SMFLRC	Logical read count
SMFPRC	Physical read count
SMFLWC	Logical writes
SMFDMC	DML COMMITs
SMFDMR	DML ROLLBACKs
SMFDED	DEADLOCKs
SMFHDLN	Length of SMF header

Contents of the SMF Net Data Section

Table 9–5 contains brief descriptions for the labels in the SMF Net data section.

Table 9–5 Contents of the SMF Oracle Net Data Section

ORASMF0 Label	Description
SMFNET	Net section header
SMFNETL	Length of Net NIV information. The information contained in this section is specific to the Net driver in use. This information is variable length.
SMFNETA	Start of variable length information. Refer to next section, "Oracle Net Network Information Vector Overview".

Oracle Net Network Information Vector Overview

Oracle Net constructs a Network Information Vector (NIV) list containing information about the network origin of an incoming client connection. This information is available to the Logon User Exit and is also written out in the SMF record for each user.

The vector list is preceded by a 2-byte length field indicating the length of the entire list including the length field itself. The individual vectors in the list consist of a 1-byte length field indicating the length of the vector, a 1-byte vector ID field identifying the vector, and a variable number of vector-specific data bytes. The first NIV in the list is always a protocol identification NIV, which will identify the Net protocol being used as well as the network location of the client.

Currently, only the protocol identification NIV is built. Other NIVs may be added in the future as required. The following tables describe the NIV list and the individual NIV formats.

NIV List Format

The following table describes the NIV list:

Byte	Contents
0 through 1	total length of NIV list including bytes 0 and 1
2 through m	first NIV
m+1 through n	second NIV
n+1 through ...	......
.. .-x	......
x+1 through y	last NIV

General NIV Format

The following table describes the general NIV format:

Byte	Contents
0	total length of NIV including byte 0
1	ID of NIV
2 through p	NIV data

Protocol Identification NIV

The following table describes the protocol identification NIV:

Oracle Net TCP/IP Identification NIV

The following table describes the Oracle Net TCP/IP identification NIV format:

Byte	Contents
0	NIV length = x'08'

Byte	Contents
1	NIV ID = x'03'
2 through 3	TCP port number from which client originated
4 through 7	Internet address on which client resides (hex format)

Contents of the SMF OS/390 Accounting Data Section

Table 9–6 contains brief descriptions for the labels in the SMF OS/390 accounting data section.

Table 9–6 Contents of the SMF OS/390 Accounting Data Section

ORASMF0 Label	Description
SMFACT	OS/390 accounting section header.
SMACTNF	Number of accounting fields. This field is equivalent to the SMF5ACTF field in the SMF job termination (type 5) record.
SMFACTA	OS/390 accounting information. This field is equivalent to the SMF5JSAF field in the SMF job termination (type 5) record.
SMFACLN	Length of OS/390 accounting section.

Sample Formatting Program for SMF Records

A sample program, ORAFMTO, is provided with the Oracle SMF interface to format Oracle SMF records. ORAFMTO is an Assembler program that reads and formats SMF accounting records with the default Oracle type of 204. It reads records from a variable-blocked sequential data set and writes the formatted records to a fixed-block sequential data set with a logical record length of 133.

The sample ORAFMTO program is in the Oracle SRCLIB library. These members are included:

ORAFMTCL	contains sample JCL to assemble and link ORAFMTO.
ORAFMTGO	contains sample JCL to run ORAFMTO.
ORAFMTO	extracts and prints values from the Oracle SMF records.

Table 9–7 SMF Record Values

Value	Description
SSN	OSDI Subsystem name
SERVICE	OSDI Service name
SMFAUTH	Authorization id = TSO logon id Batch user id on jobcard CICS USERID, TERM-ID,TRANS-ID, PROGRAM-ID, or OPID
SMFCONN	Connection type (TSO,BATCH,CICS,VTAM,TCP/IP, IMS)
Oracle ID	Oracle user id. (This field is blank if the connection is not associated with a user id.)
DATE	Start date of Oracle session
TIME	Start time of Oracle session
CPU SECONDS	Total CPU seconds used in the Oracle address space (SMFXMCPU)
LOG READS	Count of logical reads
PHY READS	Count of physical reads
LOG WRITES	Count of logical writes
DMC	Data Manipulation Language Commits
DMR	Data Manipulation Language Rollbacks
DED	Deadlocks
HI STG	High-water mark of main storage used by the session

If any of the values are too large for the precision their column allows, then they are shown as a series of asterisks.

Sample output from the ORAFMTO program:

```
SSN  SERVICE  SMFAUTH  SMFCONN  ORACLE ID      DATE   TIME         CPU SECONDS   LOG READS  PHY READS LOG WRITES DMC DMR DED HI STG
---- -------- -------- -------- -------------- ------ ----------- -------------- ---------- ---------- ---------- --- --- --- ------
ORA1 ORA108   MJJONES  BATCH    SYS            00.259 08:06:25.82       1.258847       5396        220         33   0   0   0   955K
ORA1 ORA108   MJJONES  TSO      SCOTT          00.259 08:07:16.43       1.604269       1611       1037          8   0   0   0   673K
ORA1 ORA108   MJJONES  TSO      SYSTEM         00.259 08:10:35.49        .318189        614         17         97   3   1   0   675K
ORA1 ORA108   MJJONES  BATCH    SCOTT          00.259 10:12:33.04        .580421       3006         96         23   0   0   0   772K
```

Auditing Database Use

Oracle9*i* allows system-wide audit records to be written to operating system audit trails. Oracle9*i* for OS/390 uses the System Management Facility (SMF) as its operating system audit trail. This section describes OS/390-specific considerations for defining and using an operating system audit trail. Refer to *Oracle9i Database Concepts* and *Oracle9i Database Administrator's Guide* for additional information on auditing.

Preparing To Record Oracle9*i* Audit information

Two steps must be performed before SMF audit recording can take place:

1. Two INITORA parameters must be specified.
2. SMF recording by the Oracle9*i* instance must be enabled.

Until both steps are completed no SMF auditing will occur.

The first step is to specify INITORA parameters AUDIT_TRAIL and AUDIT_FILE_DEST. AUDIT_TRAIL=OS is required to inform Oracle9*i* that operating system auditing is desired. AUDIT_FILE_DEST is required to indicate the desired SMF record number.

When specifying a record type, you must select a user SMF record type not conflicting with any other user record types. This includes the Oracle9*i* record type described in "Specifying the Oracle Record Type", earlier in this chapter. For example, specifying AUDIT_FILE_DEST=205 causes Oracle9*i* audit records to be written to SMF record type 205.

SMF recording of Oracle accounting information starts automatically at startup if SMF is activated and if the Oracle record number is specified.

Because the standard system default record types activated for SMF are 128 through 255, and because the recommended number for the Oracle server (204) is within this range, many sites automatically begin SMF recording of Oracle records when Oracle is installed, and the SMF_STAT_RECNO parameter is supplied.

The service must be stopped and restarted for this parameter to take effect.

The second step is to enable SMF recording by the Oracle9*i* instance. You have two ways to accomplish this:

1. You can use the SMFPRMxx member of SYS1.PARMLIB; or
2. You can use the OS/390 SETSMF command.

For example, issuing the following SETSMF command would cause subsystem ORA1 to begin writing SMF records to user type 205.

```
SETSMF SUBSYS(ORA1,TYPE(205))
```

Continuing the example, if you decide also to write Oracle9*i* accounting records to SMF and use the recommended SMF record type of 204 for those records, then the following SETSMF command activates SMF recording for both record types.

```
SETSMF SUBSYS(ORA1,TYPE(204,205))
```

Interpreting Oracle9*i* Audit Records

To interpret Oracle audit SMF records, you first need to dump the SMF data set to a sequential data set. You can then write a program to read the sequential data set and extract the desired records from it. A sample program to extract and print the audit records is provided in member ORAFMTAO of the installed Oracle9*i* SRCLIB data set. It refers to the copy file ORASMFAO, containing DSECTS that map and document the Oracle audit SMF record fields. The contents of the Oracle audit data are defined by Oracle.

You can customize ORAFMTAO for use in your installation. If you select an SMF record type other than 205, then update the value of the ORAREC constant to match your chosen SMF record type and reassemble the program. SRCLIB member ORAFTACL contains sample JCL to assemble and link ORAFMTAO. Sample JCL to run ORAFMTAO is provided in SRCLIB member ORAFTAGO.

10

Oracle Net

Oracle Net for OS/390 supports network communications between Oracle applications and Oracle database systems across different OS/390 systems and different operating systems. Oracle provides two listeners on OS/390, an OSDI listener (ORANET) and a generic listener (TNSLSNR). This chapter describes the two listeners and how to configure them. For more information on Oracle Net, refer to the *Oracle9i Net Services Book Set*.

The following topics are included:

Overview

Oracle provides two listeners on OS/390, an OSDI listener and a generic listener.

The OSDI listener (ORANET), also referred to as the Net service, runs as a service under an OSDI subsystem. In Oracle8*i*, Release 8.1.7 and Oracle9*i*, Release 1, all TCP and LU62 connections by Oracle applications, both client and server, were performed through the Net or Net8 service. Now, all Oracle clients on OS/390 open their own sockets.

The OSDI listener's primary function is to listen for inbound remote connections to an Oracle instance. For compatibility purposes, the OSDI listener still provides outbound connectivity services for Oracle9*i*, R1 and Oracle8*i*, 8.1.7 Oracle clients.

The generic listener is the Oracle listener (TNSLSNR) that runs on USS. It provides additional functionality that is not present in the OSDI listener. In particular, it provides support for external routines and shared servers. Although shared servers is not the preferred method for connecting to an Oracle instance running on OS/390, certain services such as XML DB, FTP, and HTTP require a shared servers connection.

OSDI Listener Architecture

On OS/390, Oracle Net is implemented as an OS/390 OSDI service running in its own address space separate from the Oracle service. The OSDI service acts as a listener for the Oracle instances. All protocol-specific code runs inside the OSDI listener.

Remote clients that access an OSDI server through an OSDI listener are dispatched on a lightweight unit of work called an enclave SRB. An enclave is created either once per session or for each SRB depending on the ENCLAVE keyword (described under "PARM" on page 10-4). An SRB is scheduled each time work is required to be done by the kernel. The enclave is deleted when the SRB completes. The OS/390 Workload Manager component may be used to control the execution characteristics of these enclave SRBs. Refer to "OS/390 Tuning" on page 16-21 of Chapter 16, "Oracle9i Performance", for further details.

For client and server support, OSDI listener TCP/IP uses the IBM macro implementation and a TCP/IP network to support network communications between the Oracle server and any remote OSDI listener TCP/IP client or server. For more information, refer to "TCP/IP Network Considerations" on page 10-7.

OSDI Listener Filenames

The product documentation, *Oracle9i Net Services Book Set*, refers to files in the following form:

`basename.extension`

where:

`basename`	is the product name.
`extension`	is the extension.

An example of this form is SQLNET.ORA.

These files are then converted to DDnames. The following DDnames are implemented under OS/390:

SQLNET	defines a data set containing any SQLNET.ORA diagnostic, ASO, or Oracle names parameters. It is not necessary to allocate this DD unless these features are desired. Refer to *Oracle9i Net Services Book Set* or the *Oracle Advanced Security Administrator's Guide* for more information.
SQLNETTC	defines a data set into which trace output is written. It is recommended that this be defined as a SYSOUT data set in a held output class.
SQLNETLG	defines a data set into which any logging output is written. It is recommended that this be defined as a SYSOUT data set in a held output class.
TNSNAMES	defines a data set containing all the TNS connect descriptors and aliases for your installation. For further information on TNS connect descriptors, refer to the *Oracle9i Net Services Book Set*. This DDname is not necessary on server JCL unless DBLINKS originates from the server.
LDAP	defines the location of the LDAP server.
TNSNAV	TNS client navigation. (Generally not used on OS/390.)
INTCHG	Interchange. (Generally not used on OS/390.)

Configuring the OSDI Listener

To create a listener under OSDI, you must first define the OSDI listener as a service using the OSDI DEFINE SERVICE command. In addition to defining the service, two other items that must be set up before the service can be started are: a JCL procedure, and network protocol-specific (TCP/IP) configuration. After you have defined OSDI listener as a service and have set up the additional items, you can start the service, which creates OS/390 address spaces based on controls that you have specified.

Network Service Definition

The OSDI DEFINE SERVICE command is described completely in Appendix A, "OSDI Subsystem Command Reference". Here, we describe DEFINE parameter considerations that are specific to the OSDI listener.

Service Name

The service name for OSDI listener can be anything that you want within the content limitations described in Appendix A.

TYPE

The TYPE parameter for a database service must be specified as Net.

PROC

This procedure specifies the name of a service JCL procedure that you will place in one of your system procedure libraries. The procedure need not exist when DEFINE SERVICE is issued, but it must be in place before the service is started. The procedure name can be anything that you choose or that the naming standards of your installation require. The requirements for this procedure are discussed in section "OSDI Listener Region JCL" on page 10-5.

PARM

The PARM string is used to specify additional initialization parameters that are specific to the OSDI listener. These parameters are in the form of keywords and determine which protocols are initialized at OSDI listener startup as well as configuration and debugging features.

A description of the OSDI listener keywords follows:

`HPNS`	specifies support for the TCP/IP protocol.
`ENCLAVE(SESS|CALL)`	specifies the duration of the enclave. When SESS is specified the enclave is created at logon and deleted at logoff. When CALL is specified the enclave is created when the server is sent a request, and is deleted when the server waits for a receive.
`PORT(`*`nnnn`*`)`	specifies the TCP/IP port number `(`*`nnnn`*`)` on which to listen for incoming connections. The default is 1521.
`GTF`	may be specified at the request of Oracle Support Services. This allows the OSDI listener internal trace to be captured to the OS/390 Generalized Trace Facility.
`DUMP(`*`nodename`*`)`	specifies the high level node, or nodes, of transaction dump data set names. The character string can be up to 26 characters in length, must follow the rules for OS/390 data set names, and must not end with a period. When an OSDI listener transaction dump occurs, then the value defined here will be prefixed to a string that includes a time and date stamp to generate a unique data set name. The default is `ORACLE.TRANDMP`.

Example of OSDI Listener Definition

```
DEFINE SERVICE NET TYPE(NET) PROC(NET) -
DESC('Oracle Network Service') -
SID(NET) -
PARM('HPNS GTF PORT(1521) DUMP(ORACLE.TRANDMP)')
```

Note: The entire **`PARM()`** string must be on one line.

OSDI Listener Region JCL

As with a database service, a JCL procedure must be placed in a system procedure library prior to attempting a start of the service. The EXEC card of the JCL must be equivalent to the following:

```
//NET     EXEC PGM=ORANET,REGION=0M
```

`REGION=0M` is specified to ensure that the service can allocate as much private virtual memory as it needs. Some OS/390 systems may prohibit or alter a REGION parameter such as this, so you might want to check with your systems programmer to determine if any changes must be made to allow the system to accept your REGION parameter. In addition, the following DD statements are required:

STEPLIB: This DD statement should be the same as specified for the database service. Refer to "Database Region JCL" on page 3-6.

NET8LOG: Connection-related informational messages, warning messages, and error messages are written to this sequential output file. Oracle Corporation recommends that it also be assigned to a JES spool file.

Note: If the IBM TCP/IP protocol is used, the OSDI listener JCL procedure name must have an associated OS/390 userid. Refer to the next topic, "TCP/IP Network Considerations", for details.

Example of OSDI Listener Procedure JCL

```
//NET EXEC PGM=ORANET,REGION=0M
//STEPLIB DD DSN= ORAN.ORAV.AUTHLOAD,DISP=SHR
//NET8LOG DD SYSOUT=X
```

Example of NET8LOG output

```
2000034 09:50:35.0 MIN0017I message service subtask initialized
2000034 09:50:35.0 MIN0016I command service subtask initialized
2000034 09:50:35.1 MIN0018I bind/unbind service subtask initialized
2000034 09:50:35.2 MIN0026I timer service subtask initialized
2000034 09:50:35.2 MIN0002I networking service NETC     initialization complete
2000034 09:50:35.2 MIN0005I global vector is at 19F0A000
2000034 09:50:35.2 MIN0024I connected to WLM subsystem OSDI
2000034 09:50:50.4 MIN0700I HPNS INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:50.5 MIN0724I HPNS GHBY INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.1 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.1 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.1 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0728I HPNS KID INITAPI call performed.  RC=0000, EC=00000
2000034 09:50:51.2 MIN0713I I am listening on port 01522 socket 00000
2000034 10:05:58.8 MIN0733I Socket 0000 connected Subtask Kid1, IP 144.025.040.217, Port 01129.
2000034 10:05:58.8 MIN0733I Socket 0000 connected Subtask Kid2, IP 144.025.040.217, Port 01130.
2000034 12:00:13.9 MIN0098I networking service NETC     termination in progress
```

```
2000034 12:00:18.9 MIN0722I HPNS Kid #003 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #001 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #006 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #002 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #005 shut down.
2000034 12:00:18.9 MIN0722I HPNS Kid #004 shut down.
2000034 12:00:18.9 MIN0723I HPNS Gethostbyname subtask ended.
2000034 12:00:18.9 MIN0721I HPNS shut down, GoodBye.
2000034 12:00:18.9 MIN0091I timer service subtask terminated
2000034 12:00:18.9 MIN0095I bind/unbind service subtask terminated
2000034 12:00:18.9 MIN0093I command service subtask terminated
2000034 12:00:18.9 MIN0094I message service subtask terminated
MIN0000I End of Net8 Log.
```

TCP/IP Network Considerations

The OSDI listener uses the MACRO API interface for TCP/IP, and distributes the communications processing workload across multiple tasks in the OSDI listener address space.

If the IBM stack is being used, then particular attention must be paid to the MAXFILEPROC and MAXSOCKETS parameters (under AF_INET) in the BPXPRM*xx* member of SYS1.PARMLIB. These parameters must be set high enough to support the expected connection load. Both of these parameters can limit the number of connections that the OSDI listener will be able to open. Also, the OSDI listener JCL procedure name must have an associated OS/390 userid in order to use TCP/IP, which is controlled by OS/390 UNIX System Services (USS). The userid must have an OMVS RACF segment (or equivalent, if a product other than RACF is used) if the installation is not using a default OMVS segment.

In addition, the interface resolves names through the standard GETHOSTBYNAME API. Thus the resolution depends on how IBM TCP/IP is configured. If a DNS is defined to TCP/IP, then it will be used. Otherwise, TCP/IP will default the processing to its SITEINFO file. Also, IBM's Language Environment runtime library (LE) must be available through a STEPLIB DD or linklist to the OSDI listener address space in order for GETHOSTBYNAME to work. This is an IBM requirement. TNS does a GETHOSTBYNAME call at startup to test the function. This call may take minutes to complete if a busy name server is involved. The interface is not ready for work until the MIN0713I message is displayed on the system console. For more information on the GETHOSTBYNAME API, refer to the relevant IBM documentation on TCP/IP.

Client-Server Access Using the OSDI Listener

Remote Clients

Remote (inbound) clients access Oracle database instances through the OSDI listener, as follows:

1. The OSDI listener listens on a single endpoint (network address) for each protocol. All remote clients that go through a particular OSDI listener with a particular protocol use the same network address regardless of which database instance they want to access. All TCP/IP clients specify the same hostname (or IP) and port number.
2. Clients indicate the target database instance that they want with the `'(CONNECT_DATA=(SID=ssss))'` clause in the OSDI listener address string.

Oracle clients on OS/390 are also able to use an Oracle Names or LDAP server running on another platform to resolve connection requests. The following samples of the OSDI listener configuration file are required to make use of this service.

Name Server

SQLNET DD or SQLNET.ORA Definitions

```
################
# Names .........:  (CONNECT_TIMEOUT = 0) -MUST- be specified
################
NAMES.DEFAULT_DOMAIN = world
NAMES.DEFAULT_ZONE = my.domain.com
NAMES.DIRECTORY_PATH = (TNSNAMES,ONAMES,LDAP)
NAMES.PREFERRED_SERVERS =
     (ADDRESS_LIST =
       (DESCRIPTION =
         (ADDRESS =
           (PROTOCOL = TCP)
           (HOST = names_host)
           (Port = 1575)
         )
         (CONNECT_TIMEOUT = 0)
       )
     )
##
```

LDAP Server

LDAP DD or LDAP.ORA Definitions

A sample LDAP.ORA file:

```
DEFAULT_ADMIN_CONTEXT = "c=us"
DIRECTORY_SERVERS = (hostname:389:636)
DIRECTORY_SERVER_TYPE = OID
```

LDAP.ORA can be generated using the NETCA utility.

Operating the OSDI Listener

The OSDI listener is started by the OSDI subsystem start command, for example:

```
ORSS START NET
```

This command would start the OSDI listener defined in the earlier example for "Example of OSDI Listener Definition" on page 10-5 if the subsystem were named 'ORSS'. You should then see the OSDI listener PROC start up followed by the following messages from the OSDI listener address space:

```
MIN0001I networking service initializing
MIN0002I networking service NET8     initialization complete
MIN0713I I am listening on port 01521 socket 00000
```

Additional messages are written to the NET8LOG DD, but message traffic to the console is limited to error and warning messages.

Several commands are available for communicating with a running Net service. Commands are issued using the OS/390 MODIFY (or F) system operator command with the general format:

F *name,cccc pppppp*

where:

name	is the jobname or identifier of the OSDI listener
cccc	is a command verb from the table below
pppppp	represents an appropriate parameter for that command

Table 10–1 Command Verbs for OS/390 MODIFY (or F) System Operator Command

`start`	`hpns`	Starts support for the specified protocol in the OSDI listener.
`stop`	`hpns`	Stops support for the specified protocol.
`dis`	`tcp | all | pool`	Display information about existing connections for the specified protocol or storage pool statistics.

The OSDI listener can be stopped with the OS/390 stop command (STOP or P), as in '`p net`', or via the OSDI subsystem stop command, as in '`ORSS STOP NET`'. In either case, the following messages will be seen on the console, assuming both protocols were active:

```
MIN0098I networking service NET termination in progress
MIN0721I HPNS shut down, GoodBye.
MIN0099I networking service termination complete
```

The OSDI listener will also respond to the OSDI subsystem '`display`' and '`display long`' commands with appropriate information from the address space. Finally, the OSDI subsystem '`drain`' command will prevent any new connections on either protocol. Existing connections will not be affected. The OSDI subsystem '`RESUME`' command will restore the ability of clients to establish new connections through the OSDI listener.

Formatting OSDI Listener Trace Files

The OSDI listener provides a utility program called TRCASST that formats the trace files the OSDI listener can produce. You may be asked to run TRCASST to help gather diagnostic information required by Oracle Support Services. Sample JCL for TRCASST is provided in *oran.orav*.SRCLIB(TRCASST).

Before you use TRCASST, ensure that the trace files have not been created with carriage control. TRCASST will be unable to process such files.

When TRCASST runs, the TNSUSMSG DDname must point to a PDS containing a TNSUS message file. This file was placed into *oran.orav*`.`MESG(TNSUS) during OSDI listener installation.

Oracle Advanced Security Option Encryption

The OSDI listener supports CHECKSUM and encryption algorithms. The following sections describe a basic method of verifying this feature, if it is to be used by your site. The easiest way to tell if Oracle Advanced Security Option (ASO) encryption is attempting to work is to deliberately set wrong configuration parameters and attempt a connection between the server and client. Incorrect parameters cause the connection to fail.

After receiving the expected failure message, set the configuration parameters to the correct settings and try the connection again. ASO encryption is working properly if no further error messages are received.

The following procedures test ASO encryption by this method. The incorrect parameter settings produce error 12660.

Setting Up ASO Encryption for Test

Checklist for Setting Up ASO Encryption

1. Set ASO encryption parameters for the server
2. Set ASO encryption parameters for the client

Step 1: Set ASO Encryption Parameters for the Server

Use ISPF to edit the OSDI listener configuration file on the OS/390 system (server system) to add the following parameters and values. If the server is remote (not OS/390), then use the appropriate editor for the server platform to change SQLNET.ORA.

```
SQLNET.CRYPTO_CHECKSUM_SERVER = REJECTED
SQLNET.ENCRYPTION_SERVER = REJECTED
SQLNET.CRYPTO_CHECKSUM_TYPES_SERVER = (MD5)
SQLNET.ENCRYPTION_TYPES_SERVER = (DES40,RC4_40)
SQLNET.CRYPTO_SEED = "abcdefg"
```

The value shown for `SQLNET.CRYPTO_SEED` is only an example. Set it to the value you want. Refer to the *Oracle Advanced Security Administrator's Guide* for more information.

Step 2: Set ASO Encryption Parameters for the Client

Edit the OSDI listener configuration file on the client system to add the following parameters:

```
SQLNET.CRYPTO_CHECKSUM_CLIENT = REQUIRED
SQLNET.ENCRYPTION_CLIENT = REQUIRED
SQLNET.CRYPTO_CHECKSUM_TYPES_CLIENT = (MD5)
SLQNET.ENCRYPTION_TYPES_CLIENT = (DES40,RC4_40)
SQLNET.CRYPTO_SEED = "abcdefg"
```

The value shown for `SQLNET.CRYPTO_SEED` is only an example. Set it to the same value used on the server system.

Testing ASO Encryption

After completing Steps 1 and 2 of the configuration procedure, you are ready to test the operation of the ASO encryption.

Checklist for Testing ASO Encryption

1. Connect client and server
2. Reset configuration parameters on server

Step 1: Connect Client and Server

Attempt a connection between the server and client systems. You receive the following error message:

```
ORA-12660: Encryption or crypto-checksumming parameters incompatible
```

Step 2: Reset Configuration Parameters on Server

Change the ASO encryption parameters on the server to:

```
SQLNET.CRYPTO_CHECKSUM_SERVER = REQUIRED
SQLNET.ENCRYPTION_SERVER = REQUIRED
```

Attempt the connection between the client and server again. If no error message is returned and the connection completes, then ASO encryption is working properly.

Generic Listener Architecture

The generic listener runs under OS/390 Unix System Services (USS). It listens for incoming client connection requests and manages traffic to the server. When a client requests a network session with a database server, the generic listener receives the actual request. If the client information matches the generic listener information, then the generic listener grants a connection to the database server.

Connections to the database are only supported through shared servers (formerly multi-threaded servers, or MTS). The Oracle instance must register its shared servers with the generic listener. For more information about the generic listener, refer to the *Oracle9i Net Services Administrator's Guide*.

Oracle External Routines

Oracle external routines (previously known as external procedures) are functions or procedures written in a third-generation language that can be called from PL/SQL code. The supported languages are C and Java. The user-written applications must be invoked under OS/390 Unix System Services (USS) and must be in DLL (dynamic loadable library) form. The use of external routines requires the generic listener to listen for external routine calls. For information on configuring the generic listener for external routines, refer to "Generic Listener Configuration Steps" on page 10-14. For more information about external routines, refer to the *Oracle9i Application Developer's Guide - Fundamentals*.

Oracle Shared Servers

A shared server is a database server that is configured to allow many client processes to share very few server processes. With shared server architectures, client processes ultimately connect to a dispatcher, which handles multiple incoming network session requests to shared server processes. The process monitor (PMON) process registers the location and load of the dispatchers with the generic listener, enabling the generic listener to forward requests to the least loaded dispatcher. Certain Oracle functionality (XML DB, FTP, HTTP) is only provided using shared servers. Oracle shared servers are supported on OS/390 through the generic listener running on USS. For information on configuring the generic listener for shared servers, refer to "Generic Listener Configuration Steps" on page 10-14. For more information about shared servers, refer to the *Oracle9i Net Services Administrator's Guid*e.

Generic Listener Configuration Steps

The following steps describe how to configure the generic listener for external routines and shared servers under USS.

The command examples in this section were created with the assumption that you are connected under TSO. If you are connected via rlogin, then you may be using native UNIX commands, and you will need to modify the statements that have been provided in this chapter.

The user-written applications stored in the DLL are invoked by the Oracle server by issuing a TCP connection to the generic listener. The generic listener then executes the user-written application from the DLL.

Note: As with other USS components, you will need to set your environment variables correctly, including $PATH, $LIBPATH, and $TNS_ADMIN. Refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* for more information.

The generic listener configuration steps are as follows:

- Step 1: Add the Init.ora Parameter on page 10-14
- Step 2: Create and Modify the Tnsnames.ora File on page 10-15
- Step 3: Create and Modify the Listener.ora File on page 10-16
- Step 4: Start the Generic Listener on page 10-17

Step 1: Add the Init.ora Parameter

This step is required for shared servers, only. If you are configuring the generic listener for external routines, skip to the next step.

For shared servers, the Oracle instance must be configured with the following init.ora parameter:

```
DISPATCHERS                                      ="(ADDRESS=
(PROTOCOL=TCP) (PORT=port_nurnber_l))
                     (LISTENER=tnsname_entry))"
DISPATCHERS        ="(ADDRESS= (PROTOCOL=TCP) (PORT=port_number_2))
                     (LISTENER=tnsname_entry))"
MAX_DISPATCHERS    =xx
MAX_SHARED_SERVERS =yy
```

where

port_number_n	is the port number that the dispatcher will listen on
tnsnames_entry	is the connect string name used to locate the appropriate tnsnames.ora entry
xx	is the number of dispatchers
yy	is the number of servers

Step 2: Create and Modify the Tnsnames.ora File

Edit the tsnames.ora file for the Oracle server for either external routines or shared servers, as described in the following sections.

For External Routines

Add to the tnsnames.ora file an entry similar to the following:

```
EXTPROC_CONNECTION_DATA =
  (DESCRIPTION=
(ADDRESS=(PROTOCOL=TCP)(HOST=hostname)(PORT=nnnn))
      (CONNECT_DATA=(SID=EXTPROC))
  )
```

where:

hostname	is the name of the OS/390 machine where the OSDI RDBMS is running
port=*nnnn*	is the TCP port number on which the network listener will listen (refer to "Step 3: Create and Modify the Listener.ora File" in the following section.)

For Shared Servers

Add to the tnsnames.ora file an entry similar to the following:

```
TNSNAME_ENTRY =
 (DESCRIPTION=
  (ADDRESS=(PROTOCOL=TCP) (HOST=hostname) (PORT=listener_port))
 )
```

where:

tnsnaxne_entry	`is the tnsname_entry specified in the init.ora file`
hostname	`is the node name of the listener`
listener_port	is the port number specified in listener.ora for the listener

Step 3: Create and Modify the Listener.ora File

Create the listener.ora file by issuing the following edit statement:

```
oedit $ORACLE_HOME/network/admin/listener.ora
```

Modify the listener.ora file for either external routines or shared servers, as described in the following sections.

For External Routines

Add to the listener.ora file an entry similar to the following:

```
listener =
  (address_list =
    (address=(protocol=tcp) (host=hostname) (port=nnnn) )
  )

sid_list_listener =
  (sid_list =
    (sid_desc =
      (sid_name = extproc)
      (oracle_home = $ORACLE_HOME)
      (program = extproc)
    )
  )
```

where:

hostname	is the name of the OS/390 machine where the OSDI RDBMS is running
port=*nnnn*	is the TCP port number on which the generic listener will listen

---

Note: The `oracle_home` entry in the example above must be given explicitly, not as `$ORACLE_HOME`. It must, however, match the value of `$ORACLE_HOME`.

At this point in the configuration process, make sure that your database and network services are running.

For Shared Servers

Add to the listener.ora file an entry similar to the following:

```
listener = (description=
            (address_list=
             (address=(protocol=tcp)(host=hostname)(port=listener_port))
             )
            )
LOGGING_LISTENER         = ON
LOG_DIRECTORY_LISTENER   = /oracle_home/ log!
TRACE_LEVEL_LISTENER     = 16
TRACE_DIRECTORY_LISTENER = /oracle_home/ trace!
DIRECT_HANDOFF_LISTENER  = NO
```

where:

`hostname`	is the node name of the generic listener
`listener_port`	is port on which the generic listener is listening

Note: On OS/390 "DIRECT_HANDOFF_LISTENER=NO" is a required parameter.

At this point in the configuration process, make sure that your database and network services are running.

Step 4: Start the Generic Listener

Perform the following steps to start the generic listener:

a. Verify that the ORACLE_HOME environment variable is set to the same value that was used when you extracted the contents of the Oracle9*i* Text tar file.

b. Run the generic listener control program by entering "lsnrctl" from the USS environment.

c. When prompted with the LSNRCTL> prompt, enter "start" to start the generic listener.

d. After you see the messages about the generic listener starting, enter "quit" to exit from the generic listener control program.

The following sample dialogue is typical of what you should expect when you run lsnrctl. Output is prefixed with the character ">". The character ">" will not appear on your screen. Everything else, such as "listen" and "start", is typed at the keyboard.

```
lsnrctl
> LSNRCTL for OE390: Version 9.0.1.0.0 - Production on dd-Mon-yyyy hh:mm:ss
> (c) Copyright 1991, 2001, Oracle Corporation.  All rights reserved.
> Welcome to LSNRCTL, type "help" for information.
> LSNRCTL>
start
> Starting $ORACLE_HOME/bin/tnslsnr: please wait...
> TNSLSNR for OE390: Version 9.0.1.0.0 - Production
> System parameter file is $ORACLE_HOME/network/admin/listener.ora
> Log messagss written to $ORACLE_HOME/network/log/listener.log
> Trace inforamtion written to $ORACLE_HOME/network/trace/listener.trc
> Listening on: (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=hostname)(PORT=nnnn)))
> Connecting to (address=(protocol=tcp)(host=mvs02)(port=nnnn))
> STATUS of the LISTENER
> -----------------------
> Alias                    LISTENER
> Version                  TNSLSNR for OE390: Version 9.0.1.0.0 - Production
> Start Date               DD-Mon-yyyy hh:mm:ss
> Uptime                   0 days 0 hr. 0 min. 1 sec
> Trace Level              support
> Security                 OFF
> SNMP                     OFF
> Listener Parameter File $ORACLE_HOME/network/admin/listener.ora
> Listener Log File       $ORACLE_HOME/network/log/listener.log
> Listener Trace File     $ORACLE_HOME/network/trace/listener.trc
> Listening Endpoints Summary...
>   (DESCRIPTION=(ADDRESS=(PROTOCOL=tcp)(HOST=hostname)(PORT=nnnn)))
> Services Summary
> Service "extproc" has 1 instance(s).
    Instance "extproc", status UNKNOWN, has 1 handler(s) for this service.
>..
```

```
> The command completed successfully
> LSNRCTL>
quit
```

The listener should now be running.

11 Oracle Access Manager for CICS

This chapter discusses how to configure and operate the Oracle Access Manager for CICS. The following topics are included:

Overview

Oracle Access Manager for CICS communicates with an OSDI local database server (but not an MPM database server), or it communicates with a remote Oracle database (but not an MPM database). The target service or remote database is defined in a TNSNAMES entry that is used as input when the thread table is created. Each instance of Oracle Access Manager for CICS communicates directly with one OSDI service or one remote database.

All OSDI clients connect to a service address space using the bind mechanism. Oracle Access Manager for CICS connects to a service address space at startup time using an OSDI special purpose bind that is specific to multi-connection environments. This provides efficiencies when subsequent connections are created to service any requests from Oracle transactions in the CICS region.

Connections for Oracle CICS transactions, whether to a local or remote database, are created and reused based on criteria defined in the thread table, and they are assigned to a CICS transaction the first time that it accesses Oracle.

When the CICS transaction request requires work to be done in the Oracle database, processing is switched to a CICS subtask which contains an IBM Language Environment (LE) environment prepared for executing the Oracle client code (LIBCLNTS). The Oracle client code communicates with a local Oracle server via cross-memory services or with a remote server via Oracle networking socket calls to satisfy the request.

Oracle Access Manager for CICS Applications

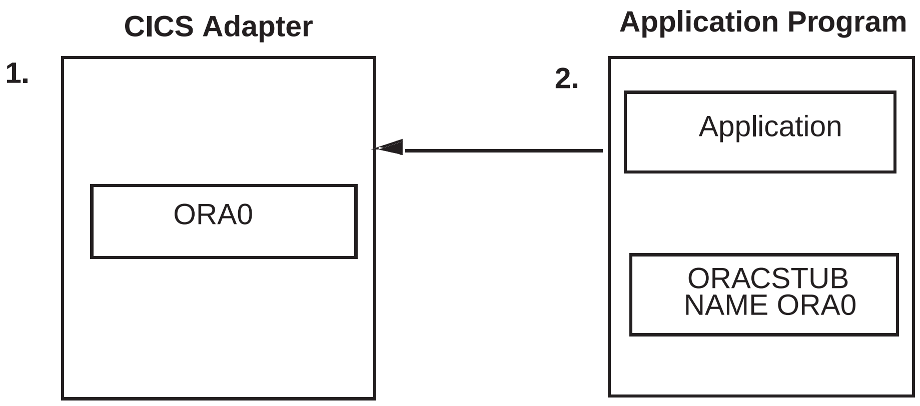
You can run applications built with these Oracle products for OS/390 under Oracle Access Manager for CICS:

- Pro*COBOL
- Pro*C
- Pro*PL/I

Oracle Access Manager for CICS Configuration

The following figure shows the relationship between an application program and Oracle Access Manager for CICS.

Figure 11–1 CICS Adapter for an Application Program



In the figure, ORA0 is the example adapter name. It is assigned on the CICS side when Oracle Access Manager for CICS is started. Refer to "Step 8: Start Oracle Access Manager for CICS Adapter" on page 11-17 and the START command details on page 11-31 for more information on the adapter name. An adapter name gets associated with an application program when it is linked with an ORACSTUB stub. Refer to "Step 6: Generate the ORACSTUB Stub for CICS" on page 11-16 for details on generating an ORACSTUB stub.

Configuration Checklist

Oracle Access Manager for CICS must be installed successfully before you can configure it. Refer to the *Oracle9i Enterprise Edition Installation Guide for OS/390*, for more information.

Configuration Steps

- ❏ Step 1: Define and Assemble Thread Definition Table
- ❏ Step 2: Define the MESG Library to CICS
- ❏ Step 3: Copy Access Manager for CICS Modules to CICS Libraries
- ❏ Step 4: Define CICS to Oracle and Grant Privileges
- ❏ Step 5: Set INITORA Parameter and Prepare Host
- ❏ Step 6: Generate the ORACSTUB Stub for CICS
- ❏ Step 7: Update CICS Tables to Include Oracle Access Manager for CICS
- ❏ Step 8: Start Oracle Access Manager for CICS Adapter
- ❏ Step 9: Set Up Automatic Initialization for Oracle Access Manager for CICS

Post-Configuration Steps

- ❏ Step 1: Modify the Sample Compilation Procedures
- ❏ Step 2: Use the SRCLIB Member OSAMPLE

Configuration Steps

Step 1: Define and Assemble Thread Definition Table

A table, defined and assembled as a load module, defines threads and their characteristics to Oracle Access Manager for CICS. When you create the thread definition table, configure the threads to match the expected workload and priorities for Oracle Access Manager for CICS users and applications.

A thread represents a connection to the database. It is maintained by Oracle Access Manager for CICS. All active CICS transactions that communicate with an Oracle server database use a thread.

Note: The thread table must be reassembled for the current release of Oracle Access Manager for CICS.

Before defining your table, select a table name appropriate for your installation. The table name is specified to Oracle Access Manager for CICS at startup in the Oracle Access Manager for CICS START command.

Note: You can also use the thread definition table to create automatic startup and shutdown (PLTPI and PLTSD) programs.

Complete these steps to build the thread definition table.

Step 1.1: Determine the Requirements for Each Transaction

Consider these criteria in determining the requirements for each transaction:

- The authorization assignment specified in the AUTH parameter
- The peak and average rates of the transaction
- Whether the transaction requires dedicated threads
- Whether Oracle COMMIT processing or CICS SYNCPOINT is to be used for commit and rollback functions. For more information, refer to "Recovery Considerations" on page 11-23.

Step 1.2: Define the Thread Requirements in the Thread Definition Table

When you define a thread table:

- Use the TRANSAC parameter to group transactions according to similar characteristics. If a transaction requires dedicated threads, then you should be sure to give it a separate definition.
- Use the COPIES parameter to specify the number of threads you need to define for each group.
- Base the thread definition on the peak and average rates of the transaction. The specified value must be large enough to handle a workload for the number of transactions specified in the TRANSAC parameter.
- Use the PROTECT parameter to specify how to protect the threads defined in the COPIES parameter. The session is not dropped, even if the IDLE TIME has expired.
- Use the WAIT parameter to specify the action to take if no thread is available.

Thread Definition Table Parameters

This list describes the thread definition table parameters:

AM4COID	is used, if specified, as the Oracle user id for the control thread. It must be a constant character string of 30 characters or less, and requires the AM4CAUTH parameter. If omitted, then the CICS applid is used for autologon. This parameter is specified on the TYPE=START statement and requires that at least one thread be defined using the OID and AUTH parameters.
AM4CAUTH	is the authentication string (password) for the Oracle user id specified in the AM4COID parameter. It must be a constant character string of 30 characters or less. This parameter is specified on the TYPE=START statement.
AUTH	determines how the Oracle user id is derived without an explicit connect statement. The Oracle userid is also used for SMF accounting. A maximum of three values can be specified for AUTH. Valid values are:

OPID	based on the operator id.
PROGRAM	causes the Oracle user id to be derived from the program name.

TERMID	causes the Oracle user id to be derived from the terminal identification.
TRANSID	causes the Oracle user id to be derived from the transaction id.
USERID	causes the Oracle user id to be derived from the CICS user id.
authorization string	causes the Oracle user id to be the specified constant character string. The implied Oracle user id is derived by prefixing the string OPS$ to the value specified by this AUTH parameter. If the OID parameter is specified, then this is the authentication string (password) used for that Oracle user id. OPS$ will not be prefixed to this string. For example, by specifying AUTH='SCOTT', OPS$SCOTT is used as the Oracle user id.

CINTERVL	specifies the control transaction time interval. This interval is used to determine how frequently unprotected idle threads are dropped. Idle threads defined with PROTECT=YES are not affected by the setting of CINTERVL. The interval also affects the length of time it can take Oracle Access Manager for CICS to invoke emergency shutdown when the Oracle server cannot be contacted. The value is expressed in minutes and seconds, and valid values range from 0000 to 5959. The default value is 0030 (00 minutes, 30 seconds).
COMMIT	specifies whether to use Oracle COMMIT processing or CICS SYNCPOINT for commit and rollback functions. Valid values are CICS and ORACLE. The default is ORACLE. This parameter can also be specified on the TYPE set to START statement or the Oracle Access Manager for CICS START command.
COPIES	indicates the number of threads this definition generates.
DESC	defines the OS/390 descriptor code used for the console message. Valid values are numeric. The default is 11. DESC is an optional parameter.

ENAME	is used in conjunction with TYPE=ENV to specify environment variables for Oracle Access Manager for CICS. It is useful for setting the NLS environment variables to values appropriate for your environment. The syntax is: ENAME=(var=val, ...) where `var` is the name of the environment variable you wish to set, and `val` is the value you wish to give to `var`. You can specify several `var=val` pairs separated by commas in a single ENAME parameter, or a separate TYPE=ENV can be coded for each variable. Do not enclose the names or values in quotes. The following example shows a single ENAME parameter being used to specify NLS_LANG and NLS_DATE_FORMAT:

```
ORACICS TYPE=START,SSN=ORA0
ORACICS TYPE=THREAD,COPIES=30,PROTECT=NO,          X
  TRANSAC=*
ORACICS TYPE=ENV,                                  X
  ENAME=(NLS_LANG=AMERICAN_AMERICA.WE8EBCDIC37C,   X
  NLS_DATE_FORMAT=DD-MON-RR)
```

FREESPC	defines the amount of free space to be allocated for later additions. This parameter is not implemented.
MAXCRSR	defines the maximum number of cursors used. The MAXCRSR parameter is patterned after the precompiler option MAXOPENCURSORS. Refer to the *Pro*C/C++ Precompiler Programmer's Guide* for more information about the MAXOPENCURSORS option. The default is 10.
MAXTHRD	defines the maximum number of threads allocated for this table definition. If the total threads for the table exceed this value, then the MAXTHRD value is adjusted. The default is 10.
NAME	identifies the adapter you want started. If NAME is not specified, then the started adapter name defaults to the thread table name. This parameter only applies if TYPE is set to PLTPI.
OID	is used, if specified, as the Oracle user id for this thread definition, including all threads generated with the COPIES parameter. It must be a constant character string of 30 characters or less. When specified, the value in the AUTH parameter is used as the associated authentication string.
PRIORITY	specifies whether Oracle Access Manager for CICS subtasks are run at a higher or lower priority than the current dispatching priority when subtasking is used. Valid values are HIGH and LOW. If HIGH is specified, then Oracle Access Manager for CICS subtasks are run at a higher dispatching priority. If LOW is specified, then Oracle Access Manager for CICS subtasks run at a lower dispatching priority. LOW is the default.

PROGRAM	specifies the name of the program used with the Oracle Access Manager for CICS transaction. ORACICS is the default. This parameter only applies if TYPE is set to PLTSD.
PROTECT	specifies whether a thread is protected. Valid values are YES, NO, and nn, where nn is the number of protected threads when used with the COPIES parameter. A value of YES indicates the thread must be protected. (The session is not dropped, even if IDLE TIME has expired.) A value of NO indicates the thread does not need to be protected. NO is the default.
ROUTCDE	defines the OS/390 route code to use when writing messages to the OS/390 operator console. The default is 11.
SSN	specifies the alias name used for the Oracle Net connection (as specified in the TNSNAMES entry used as input to the CIN step of the TBLJCLN job in SRCLIB) for database access. The alias must be four characters or fewer.
START	specifies the name of the program for CICS PLTPI processing when TYPE is set to PLTPI.
STOP	specifies the name of the adapter stopped during PLTSD shutdown processing. This name corresponds to the value of the NAME parameter used if shutdown processing is performed manually with the STOP command.
TRANSAC	specifies the transaction codes eligible to use the thread. A value of * indicates the thread is to be used as a default and no other definitions apply.

TYPE — is a parameter specifying the type of entry being defined. Valid values are:

CONTINUE	indicates the previous definition is being continued.
ENV	is used in conjunction with the ENAME parameter to specify environment variables for Oracle Access Manager for CICS.
END	indicates the end of a table.
START	indicates the beginning of the table and sets default values.
THREAD	indicates a thread is being defined.

PLTPI	indicates PLTPI generation. For PLTPI generation, the only valid value is PLTPI. A value of PLTPI indicates this definition generates the PLTPI program for Oracle Access Manager for CICS. Refer to "Step 9.1: Generate the PLTPI Program" for an example of a PLTPI generation statement. You must perform PLTPI generation separately from the thread table and PLTSD generation.
PLTSD	indicates PLTSD generation. For PLTSD generation, the only valid value is PLTSD. A value of PLTSD indicates this definition generates the PLTSD program for Oracle Access Manager for CICS. Refer to "Step 9.2: Generate the PLTSD Program" for an example of a PLTSD generation statement. You must perform PLTSD generation separately from the thread table and PLTPI generation.

WAIT	specifies the action to be taken when all threads specified for a transaction are in use. Valid values are YES, NO, and POOL. YES indicates the transaction is placed in a CICS WAIT state until a thread is available. NO indicates a return code is set indicating no threads are available. POOL indicates the general thread pool (TRANSAC=*) is used. YES is the default.

Sample Thread Definition Table

This sample thread definition table resides in the SRCLIB library in the ORACICSD member:

```
ORACICS TYPE=START,                             X
   SSN=ORA0,                                    X
   MAXTHRD=30,                                  X
   MAXCRSR=5
ORACICS TYPE=THREAD,                            X
   COPIES=4,                                    X
   PROTECT=2,                                   X
   TRANSAC=(TTRN,TRN2,TRN3)
ORACICS TYPE=CONTINUE,                          X
   TRANSAC=(TRN9,TRNA)
ORACICS TYPE=THREAD,                            X
   COPIES=4,                                    X
   PROTECT=NO,                                  X
   TRANSAC=(TRN1,TRNT,TRN4),                          X
   AUTH=(USERID,'STRING')
ORACICS TYPE=THREAD,                                  X
   COPIES=4,                                          X
   PROTECT=NO,                                        X
   TRANSAC=(TRN5,TRN6,TRN7),                          X
   AUTH='AUTHORIZATION_STRING'
ORACICS TYPE=THREAD,                                  X
   COPIES=4,                                          X
   PROTECT=NO,                                        X
   TRANSAC=TRN8
ORACICS TYPE=THREAD,                                  X
   COPIES=4,                                          X
   PROTECT=NO,                                        X
   TRANSAC=*
ORACICS TYPE=END
   END
```

Note: To use an explicit Oracle id for the control thread, use the AM4COID and AM4CAUTH parameters on the TYPE=START. To define threads to use an explicit Oracle id, use the OID and AUTH parameters on the TYPE=THREAD entry.

Special Considerations

These conditions apply to thread table definitions:

- If a transaction code does not have an associated thread table definition, then the general thread pool definitions (TRANSAC=*) are used.
- If all threads for a transaction are in use and the WAIT parameter specifies POOL, then the thread used has all the characteristics defined in the TRANSAC=* definition. The transaction thread characteristic is not carried over to the pool thread.
- Use AUTH='`authorization_string`' to define threads using a constant character string. AUTH='`authorization_string`' derives the Oracle user id and provides the highest level of thread sharing and throughput. AUTH set to TRANSID or AUTH set to PROGRAM performs equally only if one TRANSID or PROGRAM uses a thread definition.

Step 1.3: Assemble and Link Thread Table

1. Assemble and link the table using one of the following methods.

   > **Note:** If you are configuring Oracle Access Manager for both local and remote access, then you must assemble the remote thread table with a different name than the local thread table.

2. Assemble the table for database access. The sample JCL in SRCLIB library member TBLJCLN assembles and links the input table.

   Oracle Net allows variable block TNSNAMES files. Oracle Access Manager requires fixed block of 80. In addition, Oracle Net allows free format. Oracle Access Manager must have its entry as the only entry and cannot have any extra spaces.

   In the INFILE DD statement in the first step (CIN), specify your TNSNAMES data set containing your TNS connect descriptor and alias for CICS. The alias name for CICS must be four characters or fewer and must match the SSN parameter specified in the thread table definition or the SSN override parameter on the START command. There must be only one entry used by CICS. You might need to create a separate PDS member from one of the TNSNAMES entries for the CICS. The service name or alias must be on a separate line from the rest of the connect descriptor. A TNSNAMES specification for Oracle Access Manager might be similar to:

   ```
   ORA0=
        (DESCRIPTION=
         (ADDRESS=
         (PROTOCOL=TCP)
         (HOST=HQUNIX)
         (PORT=1533)
         (CONNECT_DATA=
         (SID=O803)))
   ```

   For a local database, the specification might be similar to:

   ```
   ORA0=
        (DESCRIPTION=
         (ADDRESS=
         (PROTOCOL=xm)
         (SID=ORA0)))
   ```

"OSDI Listener Filenames" on page 10-3 also contains a description of TNSNAMES.

In the SYSLMOD DD statement, specify the name of the data set where the table load module resides.

Step 1.4: Installing a Revised Thread Table with CICS Executing

If you want to install a revised thread table while CICS is executing, then perform these steps:

1. Ensure that you have reassembled the table as described in "Step 1.3: Assemble and Link Thread Table".
2. Issue the STOP command for the adapter.
3. Define the thread definition table to CICS using the CEDA command:

   ```
   CEDA DEFINE PROGRAM(tablename) LANG(ASSEMBLER) GROUP(groupname)
   ```

   where:

tablename	is the name of your thread definition table.
groupname	is the name of the group, typically ORACLE.

4. Issue the CICS master terminal command to install the revised thread table where tablename is the name of the revised thread definition table:

   ```
   CEMT SET PRO(tablename) NEW
   ```

5. Issue the START command for the adapter.

Step 2: Define the MESG Library to CICS

You can make the MESG Library available to CICS in one of two ways:

- Add an ORA$LIB DD statement to the CICS JCL specifying the MESG library data set name.
- Concatenate the MESG library data set name to the CICS STEPLIB DD concatenation.

Note: This library must be authorized to be in CICS STEPLIB.

Step 3: Copy Access Manager for CICS Modules to CICS Libraries

- Copy the ORACICSC and LIBCLNTS modules from the Oracle CMDLOAD library to an authorized library in the CICS STEPLIB concatenation. This library must be a PDSE library.
- Copy the ORACICS and CICADPX modules from the Oracle CMDLOAD library to a library in the CICS DFHRPL concatenation or alternatively, add the Oracle CMDLOAD library to the CICS DFHRPL concatenation.

Step 4: Define CICS to Oracle and Grant Privileges

Oracle Access Manager for CICS establishes a database connection at startup

- to perform recovery when CICS syncpoint is used,
- to manage idle connections,
- and to verify that Oracle services are available during execution.

You must define Oracle Access Manager for CICS as an Oracle user by one of the following methods:

- If you are using AM4COID and AM4CAUTH in the thread table, then the userid is the value specified in the AM4COID parameter. For example, if AM4COID=AM4CICS and AM4CAUTH=AMTHREAD, then the following statements would be used to define Oracle Access Manager for CICS:

  ```
  CREATE USER AM4CICS IDENTIFIED BY AMTHREAD;
  GRANT CREATE SESSION TO AM4CICS;
  ```

- If you are not using AM4COID and AM4CAUTH, then the Oracle Access Manager control thread uses OS authenticated logon based on the CICS applid. Refer to the *Oracle9i Database Administrator's Guide* for more information about OS authenticated logins. The OS_AUTHENT_PREFIX initora parameter is prefixed to the CICS applid to create the Oracle userid. For example, if OS_AUTHENT_PREFIX = "" (null prefix), and if the CICS applid is CICSAPPL, then the following statements would be used to define Oracle Access Manager for CICS:

  ```
  CREATE USER CICSAPPL IDENTIFIED EXTERNALLY;
  GRANT CREATE SESSION TO CICSAPPL; ''''''''''''
  ```

Note: If a local database is accessed and the CICS applid is used, it is important to review the LOGON_AUTH setting in the Database Service definition.

To enable FORCE and SELECT privileges for recovery processing if COMMIT is set to CICS (that is, if CICS SYNCPOINT processing is used instead of Oracle COMMIT), then you must issue the following statements, where *`userid`* is the Oracle Access Manager for CICS userid from above:

```
GRANT FORCE ANY TRANSACTION TO userid;
GRANT SELECT ON SYS.PENDING_TRANS$ TO userid;
GRANT SELECT ON SYS.PENDING_SESSIONS$ TO userid;
```

Step 5: Set INITORA Parameter and Prepare Host

You must complete the following steps on the server for Oracle Access Manager for CICS. For additional information, refer to the Oracle server installation documentation.

Step 5.1: Set DISTRIBUTED_TRANSACTIONS

Set the INITORA parameter DISTRIBUTED_TRANSACTIONS to a value equal to or greater than the number of concurrently executing Oracle transactions under CICS. Consider other distributed transactions that are executing concurrently under the Oracle server when setting this value.

If you are using the default Oracle user id for the control thread (the CICS applid), then you must perform the following:

Step 5.2: Set REMOTE_OS_AUTHENT_TRUE

If you are preparing a remote host for Oracle Access Manager for CICS, and if OS authenticated logon is being used, then set the INITORA parameter REMOTE_OS_AUTHENT to TRUE on the remote database.

Step 5.3: Set OS_AUTHENT_PREFIX

If OS authenticated logon is being used, then ensure that the value of the INITORA parameter OS_AUTHENT_PREFIX on the remote database is used in the GRANT statements in "Step 4: Define CICS to Oracle and Grant Privileges".

Step 6: Generate the ORACSTUB Stub for CICS

To generate the ORACSTUB stub for CICS:

1. Determine your CICS adapter name.
2. Create the generation statement. Refer to SRCLIB library member ORACICSD for an example.
3. Generate ORACSTUB. Refer to SRCLIB library member ORACJCL for an example.
4. Relink your Oracle Precompiler and OCI programs with the generated ORACSTUB. Place the library containing ORACSTUB in the SYSLIB concatenation of the link JCL and insure the SYSLIN DD contains an INCLUDE SYSLIB (ORACSTUB) statement.

Note: You must regenerate the stub for CICS and relink applications with the new stub. Use of an older stub will result in an ORAP application abend.

Step 7: Update CICS Tables to Include Oracle Access Manager for CICS

If you are running CICS release 2.2 or less, then complete Steps 7.1 through 7.3.

If you are running CICS release 3.0 or higher, then do not perform Step 7. Instead, use the RDO definition statements in member ORACSD of the Oracle SRCLIB library as SYSIN for JCL to invoke DFHCSDUP. Before you submit the job, remove all comments. Change the appropriate data set names in the DD statements as appropriate for your site.

After successful completion, issue this RDO command to install the CICS tables:

```
CEDA INSTALL GROUP (ORACLE)
```

Proceed to "Step 8: Start Oracle Access Manager for CICS Adapter".

Step 7.1: Define Oracle Access Manager for CICS Programs to CICS

Before you can use Oracle Access Manager for CICS, you must define several resources to CICS by specifying CICS CEDA transactions.

Normally definitions for resources used by CICS, such as programs and transactions, are contained in GROUPS containing logically related resources. A GROUP is created when it is first specified in a CEDA DEFINE command. In these examples, the assigned GROUP name is ORACLE.

Issue these commands to define the programs used by Oracle Access Manager for CICS to CICS:

```
CEDA DEFINE PROGRAM(ORACICS) GROUP(ORACLE) LANGUAGE(ASSEMBLER)
CEDA DEFINE PROGRAM(CICADPX) GROUP(ORACLE) LANGUAGE(ASSEMBLER)
```

After executing each CEDA DEFINE command, CICS displays the message DEFINE SUCCESSFUL at the bottom of a full screen panel.

To end a current CEDA transaction before entering a new command, press the [PF3] and [Clear] keys.

Step 7.2: Define Oracle Access Manager for CICS Transactions to CICS

Once you have specified the programs, define the Oracle Access Manager for CICS control transaction with this command. Enter the command string on one line:

```
CEDA DEFINE TRANSACTION(xxx) GROUP(ORACLE) TWASIZE(0) PROGRAM(ORACICS)
```

where `xxx` is the name of the transaction you use as the Oracle Access Manager for CICS control transaction.

Note: The Oracle Access Manager control transaction also requires you to specify the TASKDATAKEY(CICS) parameter.

Step 7.3: Install Oracle Access Manager for CICS Resources

After you have defined the programs and transactions, install the newly created Oracle group. In this example, the group name is `ORACLE`. Use this command to install the new group:

```
CEDA INSTALL GROUP (ORACLE)
```

Step 8: Start Oracle Access Manager for CICS Adapter

To make Oracle Access Manager for CICS available to CICS transactions, you must start the Oracle Access Manager for CICS adapter. An adapter is a CICS program that interfaces between transactions and a system such as Oracle. To start the adapter, you can use:

```
transaction START MOD(module)
```

where:

`module`	is the name of the assembled thread definition table containing the Oracle Access Manager for CICS control transaction.
`transaction`	is the Oracle Access Manager for CICS control transaction.

For example, if the control transaction is ORA2 and the assembled thread definition table name is ORA0, you issue the command:

```
ORA2 START MOD(ORA0)
```

If the START command is successfully executed, the connection between CICS and Oracle is successfully established.

Step 9: Set Up Automatic Initialization for Oracle Access Manager for CICS

If you want to initialize Oracle Access Manager for CICS automatically at CICS initialization time and automatically shutdown Oracle Access Manager for CICS at CICS termination, then the Oracle Access Manager for CICS control program must be invoked during CICS startup and shutdown. This step is optional. These steps describe how to generate the control program with the ORACICS macro.

Step 9.1: Generate the PLTPI Program

The ORACICS macro generates a command level CICS program that is added to the CICS PLTPI table for automatic initialization during CICS startup. An example of the command to generate the PLTPI program is:

```
ORACICS TYPE=PLTPI,START=ORA0,TRANSAC=ORA2
```

In this example, ORA0 is the thread table definition and ORA2 is the Oracle Access Manager for CICS control transaction. This command is equivalent to issuing one of these commands from a terminal:

```
ORA2 START MOD(ORA0)
```

or

```
ORA2 START ORA0
```

The ORACICS TYPE=PLTPI statement is used as input to the assembly procedure and produces an assembler language CICS command-level program.

Use the following JCL to create the PLTPI program:

```
//ORAPLTPI JOB (ORA),'Oracle PLTPI PROGRAM'
//CIN      EXEC PGM=CINAMES,REGION=4000K
//*
//STEPLIB  DD DSN=oran.orav.CMDLOAD,DISP=SHR
//SYSIN    DD DUMMY
//SYSOUT  DD SYSOUT=*
//SYSERR  DD SYSOUT=*
//*  REPLACE INFILE WITH USER TNSNAMES SOURCE TO BE USED FOR CICS
//INFILE  DD DISP=SHR,DSN=tnsnames
//OUTFILE  DD DSN=&&TEMPPDS(CINAMES),
//            UNIT=SYSDA,DISP=(,PASS),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//            SPACE=(400,(100,100,5))
//ASM      EXEC PGM=IEV90,REGION=4000K
//SYSPRINT DD SYSOUT=*
//SYSLIB  DD DSN=&&TEMPPDS,DISP=(OLD,DELETE)
//        DD DSN=oran.orav.MACLIB,DISP=SHR
//        DD DSN=CICS.MACLIB,DISP=SHR
//        DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1  DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSPUNCH   DD ***dataset for assembler pltpi source program
//SYSIN      DD *
          ORACICS TYPE=PLTPI,START=ORA0,TRANSAC=ORA2
/*
```

Step 9.2: Generate the PLTSD Program

The ORACICS macro generates a command-level program that is added to the CICS PLTSD table for automatic shutdown during CICS termination. An example of the command to generate the PLTSD program is:

```
ORACICS TYPE=PLTSD,PROGRAM=ORACICS,STOP=ORA0
```

In this example, `ORACICS` is the name of the Oracle Access Manager for CICS control program and ORA0 is the name of the CICS adapter to be stopped. This command is equivalent to issuing the terminal command:

```
ORA2 STOP NAME(ORA0)
```

The ORACICS TYPE=PLTSD statement is used as input to the assembly procedure and produces an assembler language CICS command-level program.

Use the following JCL to create the PLTSD program:

```
//ORAPLTSD JOB (ORA),'Oracle PLTSD PROGRAM'
//CIN      EXEC PGM=CINAMES,REGION=4000K
//*
//STEPLIB  DD DSN=oran.orav.CMDLOAD,DISP=SHR
//SYSIN    DD DUMMY
//SYSOUT  DD SYSOUT=*
//SYSERR  DD SYSOUT=*
//*  REPLACE INFILE WITH USER TNSNAMES SOURCE TO BE USED FOR CICS
//INFILE  DD DISP=SHR,DSN=tnsnames
//OUTFILE  DD DSN=&&TEMPPDS(CINAMES),
//            UNIT=SYSDA,DISP=(,PASS),
//            DCB=(RECFM=FB,LRECL=80,BLKSIZE=400),
//            SPACE=(400,(100,100,5))
//ASM      EXEC PGM=IEV90,REGION=4000K
//SYSPRINT DD SYSOUT=*
//SYSLIB  DD DSN=&&TEMPPDS,DISP=(OLD,DELETE)
//        DD DSN=oran.orav.MACLIB,DISP=SHR
//        DD DSN=CICS.MACLIB,DISP=SHR
//        DD DSN=SYS1.MACLIB,DISP=SHR
//SYSUT1  DD UNIT=SYSDA,SPACE=(CYL,(5,5))
//SYSPUNCH   DD ***dataset for assembler pltsd source program
//SYSIN      DD *
          ORACICS TYPE=PLTSD,PROGRAM=ORACICS,STOP=ORA0
/*
```

Step 9.3: Input PLTSD and PLTPI to the CICS DFHEITAL Procedure

Use the generated assembler language CICS command-level programs (PLTSD and PLTPI) as input to the CICS DFHEITAL procedure.

Step 9.4: Add the User-Defined Name to the PLTPI Table

Add the user-defined name of the load module containing the generated program for PLTPI (output of the DFHEITAL procedure) to the PLTPI table. If your output from DFHEITAL is named ORAPLTI, then you can also name your load module ORAPLTI. For example:

```
CEDA DEFINE PROGRAM(ORAPLTI) GROUP(ORACLE) LANGUAGE(ASSEMBLER)
```

Define the load module to CICS with a CEDA DEFINE command. The generated program can be modified to include override parameters.

For CICS release 3.1.1, this program must be placed after PROGRAM=DFHDELIM in the CICS PLTPI table.

Step 9.5: Add the User-Defined Name to the PLTSD Table

Add the user-defined name of the load module containing the generated program for PLTSD (output of the DFHEITAL procedure) to the CICS PLTSD table. If your output from DFHEITAL is named ORAPLTS, then you can name your load module ORAPLTS. For example:

```
CEDA DEFINE PROGRAM(ORAPLTS) GROUP(ORACLE) LANGUAGE(ASSEMBLER)
```

Define the load module as a program to CICS with a CEDA DEFINE command.

Step 9.6: Make Generated Programs Available

Ensure the generated programs are available in a library included in the CICS DFHRPL.

While using the PLTPI and PLTSD programs, status messages are issued to the system console.

Post-Configuration Steps

The following steps are optional.

Step 1: Modify the Sample Compilation Procedures

The Oracle SRCLIB library contains these procedures for preparing high-level language programs. Modify these procedure names according to your site's naming conventions and move them to an appropriate procedure library if necessary.

CICSPCCC	for IBM SAA AD/Cycle COBOL/370 programs
CICSPCC2	for VS COBOL II programs
CICSPCCI	for IBM C/370 programs

CICSPCCS	for SAS/C programs

Step 2: Use the SRCLIB Member OSAMPLE

SRCLIB member OSAMPLE is a COBOL II program that displays an employee name from the EMP table. To use the sample program, you must:

1. Ensure the EMP table is on the database.
2. Ensure SCOTT/TIGER is a valid logon id on the database. Also ensure SCOTT /TIGER has access to the EMP table.
3. Ensure "Step 6: Generate the ORACSTUB Stub for CICS", is completed.
4. Tailor the JCL in SRCLIB member CICSPCC2 for your installation. Then execute CICSPCC2 to compile OSAMPLE. This places OSAMPLE into a library in the CICS DFHRPL concatenation.
5. Define the program OSAMPLE to CICS by using standard RDO procedures.
6. Define a transaction to CICS (for example, OSAM) using standard RDO procedures and associate it with the OSAMPLE program.
7. Once the START command is issued for Oracle Access Manager for CICS, you can invoke the transaction.

Multiple Versions in the Same CICS Region

Multiple versions of Oracle Access Manager for CICS can coexist in the same CICS region as long as the release levels are compatible. For example, release 9.0.1 could co-exist with another release 9.0.x version but not with a release 8.1.x version.

To configure a second version, perform the following steps:

1. (This step is required only if both versions will be executing concurrently. If not, go to Step 2.) If both versions will be executing concurrently, create an ORACSTUB stub specifying the adapter name that has been chosen for the 9.2.0 version. This will be linked with applications programs accessing the 9.2.0 version.
2. This version uses the ORACICS load module. Both versions have an ORACICN load module. If a prior version uses the ORACICN load module name as distributed, there is no conflict.
3. Define a second control transaction to CICS, associating this transaction with the ORACICS program.

Recovery Considerations

Resource recovery is determined by whether COMMIT (CICS) or COMMIT (Oracle) was designated to Oracle Access Manager for CICS as the recovery choice in the thread definition table parameter COMMIT.

Using COMMIT (CICS)

CICS maintains and recovers its own protected resources (that is, VSAM data sets, DL/I databases, and so forth) with the use of the SYNCPOINT and ROLLBACK commands. Use SYNCPOINT to indicate a logical unit of work is complete and does not need to be backed out in case of failure. Use ROLLBACK to indicate a logical unit of work is incomplete and changes need to be backed out in case of failure.

When a CICS transaction syncpoints explicitly by issuing an EXEC CICS SYNCPOINT or implicitly at transaction end, Oracle Access Manager for CICS ensures all Oracle server updates in the logical unit of work (LUW) are committed or backed out. Use the Oracle two-phase commit support to accomplish this.

An Oracle error occurs if you specify the CICS option and an Oracle EXEC SQL COMMIT or EXEC SQL ROLLBACK is issued.

Using COMMIT (Oracle)

Oracle has its own recovery mechanism, independent of CICS, and utilizes the COMMIT and ROLLBACK SQL commands to control it. To commit or rollback Oracle changes, an application must issue an EXEC SQL COMMIT or EXEC SQL ROLLBACK. If an application does not issue a COMMIT or ROLLBACK SQL command, then all outstanding Oracle database updates are rolled back at the end of the CICS transaction.

A CICS SYNCPOINT also does not perform a SQL COMMIT, nor does a SQL COMMIT perform a CICS SYNCPOINT. If an application is performing update operations on both CICS protected resources and Oracle tables, then the application must issue both a CICS SYNCPOINT and a SQL COMMIT to affect both sets of resources.

Two-Phase Commit Processing under CICS

When CICS syncpoint processing is selected as the resource commit or recovery choice, Oracle Access Manager for CICS can participate in two-phase commit processing under CICS.

CICS serves as the coordinator and Oracle Access Manager for CICS uses the two-phase commit protocol to interface with the Oracle server.

This process is transparent to the transaction. Oracle Access Manager for CICS implements the CICS syncpoint resource manager interface, enabling Oracle resources to participate in CICS distributed transactions and removing the requirement that Oracle COMMIT and ROLLBACK commands be issued in addition to the CICS SYNCPOINT command.

To use the CICS syncpoint resource manager interface, you must specify CICS as the recovery choice in the thread table or as a startup option. You can specify the option by:

- Using the COMMIT thread table parameter

  For more information, refer to the "Thread Definition Table Parameters" on page 11-6.

- Using the COMMIT parameter on the START command

  For more information, refer to the START command description on page 11-31.

First Phase

In the first phase of the two-phase commit protocol, the CICS syncpoint manager presents a PREPARE request, which requests a promise to commit or rollback the transaction to Oracle Access Manager for CICS. Oracle Access Manager for CICS communicates with the Oracle server Sand returns one of three responses to the CICS syncpoint manager:

ABORT	indicates the Oracle server could not complete the CICS PREPARE request.
PREPARED	indicates the Oracle server has all resources necessary to subsequently commit and rollback the transaction. When the prepare phase is completed, the transaction is said to be in-doubt.

READ-ONLY	indicates no data has been modified so no PREPARE is necessary.

Second Phase

The second phase of two-phase commit processing is the commit phase. The CICS syncpoint manager presents a COMMIT/ROLLBACK request to Oracle Access Manager for CICS. Oracle Access Manager for CICS then communicates with the Oracle server to complete the processing.

Failures can result during two-phase commit processing. These items are in-doubt resolutions:

- CICS warm or emergency restart
- Oracle server restart
- Manual recovery

CICS Warm or Emergency Restart

In this case, CICS has access to a log of in-doubt LUWs. When Oracle Access Manager for CICS starts, it resynchronizes with CICS and the Oracle server to obtain the transactions that are in-doubt, and communicates with the Oracle server to complete two-phase commit processing.

LUWs that include Oracle and other CICS recoverable resources can be lost in the case of a CICS cold start and might require manual resolution.

Oracle Server Restart

When Oracle Access Manager for CICS detects an Oracle server restart, it resynchronizes in-doubt Oracle server LUWs with CICS and communicates with the Oracle server to complete two-phase commit processing.

Manual Recovery

Oracle in-doubt CICS transactions should not be manually committed or rolled back using the FORCE command. However, some situations, such as a CICS cold start, might require manual intervention. The Oracle server maintains a pending transaction view, DBA_2PC_PENDING. The GLOBAL_TRAN_ID in DBA_2PC_PENDING for CICS transactions is the concatenation of these fields in the order shown:

OPS`$applid`	is the eight character CICS applid, or the value of AM4COID in the thread table.
`adapter name`	is the four-character name of an Oracle Access Manager for CICS adapter.
`logical unit of work`	is eight characters and represents the value of the logical unit of work obtained form CICS.

For more information about the OPS`$applid` and `adapter name` fields, refer to "Configuration Steps" in this chapter on page 11-5. For additional information about using DBA_2PC_PENDING for manual recovery, refer to *Oracle9i Database Administrator's Guide*.

Shutting Down Oracle Access Manager for CICS with FORCE

To shut down Oracle Access Manager for CICS forcibly, use the STOP command with the IMMEDIATE FORCE parameter. When the IMMEDIATE FORCE parameter is used with the STOP command, Oracle Access Manager for CICS terminates all threads and shuts down. Caution must be used with this parameter. The shutdown is not an orderly shutdown. For more information, refer to the STOP command on page 11-33.

If you start Oracle Access Manager for CICS after you start the Oracle server, then automatic restart takes effect when Oracle terminates. When the Oracle server is restarted, Oracle Access Manager for CICS automatically restarts and resynchronizes any in-doubt logical units of work. If you start Oracle Access Manager for CICS before the Oracle server, then you must perform the initial start of Oracle Access Manager for CICS manually.

CEDF Support

Use CICS EDF to debug Oracle Access Manager by transaction. SQL statements are shown on the EDF screen before and after each transaction is run. To begin an EDF session, start the adapter and include the keyword EDF. For example:

```
ORA2 START MOD(ORA0) EDF
You must be running Oracle Access Manager for CICS Version 4.1 or higher
to use the Execution Diagnostic Facility.
```

Oracle Access Manager for CICS Command Usage

Oracle Access Manager for CICS commands are entered with the transaction id followed by the Oracle Access Manager for CICS command. The general format is:

```
trans_id command parms
```

where:

`trans_id`	is the Oracle Access Manager for CICS transaction id (usually ORA2).
`command`	is the Oracle Access Manager for CICS command.
`parms`	is one or more parameters for the command.

Oracle Access Manager for CICS commands can be entered in two ways:

- You can enter the CICS transaction id (usually ORA2) followed by the command. For example:

  ```
  ORA2 DISPLAY NAME(ORA0)
  ```

  In this case, the control transaction sends the display to the terminal.
- You can enter only the transaction id for the control transaction. In this case, the control transaction prompts you for a command.

  Use the [PF3] key to exit from the control transaction.

DISPLAY

Options

DISPLAY NAME, DISPLAY TRAN NAME, DISPLAY STATUS NAME

Syntax

```
DISPLAY NAME(adapter)
DISPLAY TRAN NAME(adapter)
DISPLAY STATUS NAME(adapter)
```

where `adapter` is the name of an Oracle Access Manager for CICS adapter.

Purpose

This command allows you to monitor your Oracle Access Manager for CICS system. The screen displayed varies according to which DISPLAY command is issued.

DISPLAY NAME EXAMPLE

DISPLAY NAME displays a screen showing general information about an Oracle Access Manager for CICS system.

When the command:

```
ORA2 DISPLAY NAME(ORA0)
```

is issued (where ORA2 is the name of the transaction defined to control Oracle Access Manager for CICS and ORA0 is the name of the Oracle Access Manager for CICS adapter), the screen is displayed:

DISPLAY TRAN NAME EXAMPLE

DISPLAY TRAN NAME displays a screen showing all the transactions defined in a thread definition table.

When the command:

```
ORA2 DISPLAY TRAN NAME(ORA0)
```

is issued (where ORA2 is the name of the transaction defined to control Oracle Access Manager for CICS and ORA0 is the name of the Oracle Access Manager for CICS adapter), the screen is displayed

Pressing the [Enter] key refreshes the display, pressing the [PF7] key scrolls backward, and pressing the [PF8] key scrolls forward.

The status codes and the thread characteristics they indicate for the DISPLAY TRAN NAME display are:

Status Code	Thread Characteristic
20	reserved
40	use general pool thread (WAIT=POOL)
80	wait for available thread (WAIT=YES)

The status code can contain more than one meaningful value. For example, a value of 60 (40 + 20) indicates transactions use threads, are running on worker tasks, and if no threads are available, a pool thread is used.

You can display all the active threads used for a particular transaction by entering **S** to the left of a transaction displayed in the DISPLAY TRAN NAME display:

Pressing the [Enter] key refreshes the display, pressing the [PF7] key scrolls backward, and pressing the [PF8] key scrolls forward.

In this display, the value `1` in the EXEC SQL column indicates one SQL call has been performed. The columns in this display are:

THREAD	thread number
TASK	CICS task number
TERM	CICS terminal id
PROGRAM	program name
EXEC SQL	number of SQL calls performed
COMMIT	number of SQL commits performed if using ORACLE COMMIT
ROLLBACK	number of SQL rollback calls performed if using ORACLE ROLLBACK
ERROR	current error code value

DISPLAY STATUS NAME EXAMPLE

DISPLAY STATUS NAME displays a screen showing all active threads. An active thread is one currently in use by a CICS transaction.

When the command:

```
ORA2 DISPLAY STATUS NAME(ORA0)
```

is issued (where ORA2 is the name of the transaction defined to control Oracle Access Manager for CICS and ORA0 is the name of the Oracle Access Manager for CICS adapter), the screen is displayed

Pressing the [Enter] key refreshes the display, pressing the [PF7] key scrolls backward, and pressing the [PF8] key scrolls forward.

The columns in this display are:

THREAD	thread number
TASK	CICS task number
TRAN	CICS transaction id
TERM	CICS terminal id
PROGRAM	program name
LOGONID	thread user's logon id
STATUS	status code

The status codes and the thread characteristics they indicate for the DISPLAY STATUS NAME display are:

Status Code	Thread Characteristic
00	no values set
08	reserved
20	high priority thread (for subtasks only, PRIORITY set to HIGH)
40	pool thread
80	protected

Note: All the status codes except code 20 apply to the display of both local and remote Oracle data. Status code 20 applies only to local Oracle data.

The status code can contain more than one meaningful value. Some codes are not listed and can be ignored because they are for diagnostic purposes.

To purge a transaction or thread from within the DISPLAY STATUS NAME display, enter **P** to the left of the transaction or thread. You receive one of the following responses indicating the outcome of the purge operation:

*	indicates the purge was successful.
E	indicates the purge was not successful.
?	indicates the current transaction was completed before the purge was attempted.

Caution: The purge facility performs cleanup, and issues an ORAP application abend for the transaction.

START

Syntax

```
START MOD(modname) [MAX(threads) SSN(ssn) NAME(adapter) COMMIT(option)]
```

where:

`modname`	is the name of the load module for the Oracle Access Manager for CICS thread definition table. If the NAME parameter is not specified, then this is used as the name of the Oracle Access Manager for CICS adapter.
`threads`	is the maximum number of threads supported by this adapter. This overrides the value specified in the ORACICS macro.

`ssn`	is the name of the Oracle subsystem corresponding to this adapter, if the Oracle Access Manager for CICS transaction is accessing local Oracle data. If the transaction is accessing remote Oracle data, then the `ssn` is the four-character alias name used in the CICS TNSNAMES entry.
`adapter`	is the name of the CICS adapter. If this parameter is not specified, then the `modname` is used as the name for the CICS adapter.
`option`	is the recovery choice. This overrides the value specified in the ORACICS macro. The two valid values for `option` are:

CICS	to use CICS SYNCPOINT
ORACLE	to use Oracle COMMIT/ROLLBACK

Purpose

This command starts the Oracle Access Manager for CICS adapter.

The parameter values specified in the START command override any values specified in the load module named in the MOD parameter.

Because the load module specified in the MOD parameter contains thread definitions, starting the adapter with only the MOD parameter is normally sufficient.

START EXAMPLE

This example provides only the load module and adapter names:

```
ORA2 START MOD(ORA0) NAME(ORA0)
```

where:

ORA2	is the Oracle Access Manager for CICS control transaction
ORA0	is the name of the load module
ORA0	is the name of the thread definition table. Subsystem name and thread definitions are taken from the `ORA0` table.

In this example, an override is specified for the Oracle Access Manager for CICS adapter name.

STOP

Syntax

```
STOP NAME(adapter) [IMMEDIATE] [FORCE] [WAIT]
```

where:

`adapter`	is the name of an Oracle Access Manager for CICS adapter started with the START command.
IMMEDIATE	is an optional parameter that rejects all requests but does not shut down until the system quiesces.
FORCE	is an optional parameter that terminates all threads and shuts down. Any active transactions will receive an ORAP application abend on the next Oracle access. When there are no active Oracle transactions, the adapter will shut down.
WAIT	is an optional parameter that waits for the adapter to shut down before returning control to the terminal.

Purpose

This command stops an Oracle Access Manager for CICS adapter started with the START command. When the FORCE option is used with the STOP command, Oracle Access Manager for CICS abends all currently running transactions and forcibly shuts down.

STOP FORCE EXAMPLE

This example shows the STOP command with IMMEDIATE FORCE:

```
ORA2 STOP NAME(ORA0) IMMEDIATE FORCE
```

where ORA2 is the Oracle Access Manager for CICS control transaction and ORA0 is the Oracle Access Manager for CICS adapter name.

12 Oracle Access Manager for IMS TM

This chapter discusses how to configure and operate the Oracle Access Manager for IMS TM. It also describes how the product is integrated with IMS.

The following topics are included:

Oracle Access Manager for IMS TM Applications

You can run applications built with the following Oracle products for OS/390 under Oracle Access Manager for IMS TM:

- Pro*COBOL
- Pro*C
- Pro*PL/1
- Oracle Call Interface

Refer to *Oracle9i Enterprise Edition User's Guide for OS/390* for additional information about using Oracle Precompilers and Oracle Call Interface.

Integration with IMS

Oracle Access Manager for IMS TM is based on the IMS External Subsystem Attachment Facility (ESAF), a published IMS interface for incorporating external resources into IMS transaction management. ESAF defines program interfaces (exits) that IMS invokes for:

- Initialization and connection
- Signon and signoff
- Thread creation
- Synchronization
- Termination functions

Some aspects of Oracle Access Manager for IMS TM installation, configuration, and operation are specific to the ESAF architecture. This guide provides some information on ESAF; however, IBM IMS documentation is the definitive source for ESAF information.

Although ESAF is named and discussed as subsystems, Oracle Access Manager for IMS TM does not run as a subsystem in a separate address space. When Oracle Access Manager for IMS TM is configured, IMS region initialization loads the main body of Oracle Access Manager for IMS TM code into the control region and into each dependent region accessing an Oracle database. IMS calls the various ESAF exits under the application management task (dependent regions) or the ESI task (control region). No additional tasks are created by or for Oracle Access Manager for IMS TM. Processing in the control region is primarily for recovery activities. Processing in the dependent region is for Oracle requests issued by transactions running in the region. The control region must connect to Oracle successfully before any dependent region is allowed to connect.

Oracle Access Manager for IMS TM Design

Oracle Access Manager for IMS TM minimizes resource consumption for processor utilization, memory utilization, and activities that can be serialized. One connection is maintained between an IMS region and a given target Oracle database, regardless of the number of distinct transactions or security contexts (Oracle user ids) the region hosts over time. Oracle Access Manager for IMS TM logically maps the combination of IMS thread and Oracle user id to the generic Oracle server session mechanism. The sessions are created and managed dynamically. They are based on the stream of transactions created by users and configuration parameters set by the installation.

The Oracle Access Manager for IMS TM code runs in 31-bit addressing mode and resides above the 16M address line. All dynamically acquired virtual memory is also above the 16M line. Oracle Access Manager for IMS TM is completely reentrant. If the installation prefers, all IMS regions accessing Oracle databases can share a single copy of the code. Each instance of Oracle Access Manager for IMS TM (that is, each distinct Oracle database to be accessed) requires allocation of less than 4K of global (ECSA) virtual memory.

Configuration Overview

You must install Oracle for OS/390 including the Oracle Access Manager successfully before you configure the product by:

- Placing the main Oracle Access Manager for IMS TM code and supporting modules into the IMS RESLIB or another data set concatenated to RESLIB in the IMS region JCL
- Ensuring Oracle Access Manager for IMS TM code and supporting modules are available to STEPLIB and DFSESL DD statements
- Installing the linking stub module AMILS (used to link Oracle Access Manager for IMS TM client transaction programs) in a load library accessible to application developers
- Placing Oracle Access Manager for IMS TM macros (used to define the product configuration) in an appropriate Assembler language macro source code library

- Installing the sample programs and material related to installation verification (this material is not required to configure the product)

These steps are specific to Oracle Access Manager for IMS TM. The installation process can also involve other components of Oracle9*i* for OS/390 or Oracle8*i* packages.

You must configure an instance of Oracle Access Manager for IMS TM for each distinct Oracle database that is accessed by transactions in an IMS subsystem. IMS (ESAF) requires two identifying characteristics for each instance:

- Subsystem identifier (SSM)
- Language interface token (LIT)

Each is a one-character to four-character identifier you choose. The subsystem identifier is specific to Oracle Access Manager for IMS TM. It is not the subsystem identifier associated with the OSDI subsystem.

The LIT and SSM

The LIT is embedded in transaction programs using an instance of Oracle Access Manager for IMS TM. When a transaction program runs, IMS associates its LIT with a given subsystem identifier through an entry in the IMS subsystem parameter file, referred to as an SSM member of IMS PROCLIB. The SSM member is an IMS region parameter file defining each external subsystem that can be accessed from that region.

Different IMS regions can have different SSM members. The control region SSM must specify every external subsystem instance (Oracle Access Manager for IMS TM, DB2 for OS/390, or others) that the IMS subsystem can access. Dependent regions can access any external subsystem defined in the control region SSM by default. To limit the external subsystems available to a dependent region or vary the parameters for one or more subsystems from what is specified for the control region, create a separate SSM for the dependent region. To prevent access to all external subsystems from a region, specify a dummy (empty) SSM. You are responsible for ensuring that each dependent region SSM allows access to the external subsystems required by the transactions scheduled in that region.

Resource Translation Table

IMS designates the data that can be coded in an SSM entry. The few defined parameters are not sufficient for implementation of Oracle Access Manager for IMS TM. The SSM does allow you to specify an additional subsystem parameter module called a resource translation table (RTT) whose contents are not specified by IMS. IMS loads the module during region initialization and passes its address to Oracle Access Manager for IMS TM as part of the ESAF exit interface.

Oracle Access Manager for IMS TM uses the RTT to specify most of its parameters and options. At least one RTT is required for each Oracle Access Manager for IMS TM instance. You generate an RTT by coding S/370 Assembler language macros (examples provided with Oracle Access Manager for IMS TM) that create the needed parameter structures. These structures are queried by Oracle Access Manager for IMS TM at runtime.

Although the RTT is a generic ESAF entity, its contents are specific to the external subsystem being accessed. Therefore, the Oracle RTT bears no resemblance to the one used by DB2, for example, and serves a different purpose.

The installation codes macro parameters that specify:

- The Oracle Net address of the target database
- Characteristics of IMS transactions
- How target database sessions are managed
- Other details

The macro calls are assembled and linked to produce a load module. The load module is placed in (or concatenated to) the IMS RESLIB. The module's name must be specified as the RTT parameter for the Oracle Access Manager for IMS TM instance in the control region SSM. For additional details, refer to "Configuration Steps" on page 12-15.

Different RTT modules can be created for the same target Oracle database to vary Oracle Access Manager for IMS TM behavior from region to region. The RTT specified in the control region SSM entry for the database is considered the master RTT. It specifies parameters that cannot be varied among the dependent regions, such as the address of the target Oracle database. All of its parameters apply in dependent regions that do not have a separate RTT. The additional RTT modules are specified in the dependent region SSM.

Distributed Option Considerations

If the target Oracle database does not have the distributed option installed, then IMS transactions are not allowed to update any data in that database.

Security Considerations

Oracle and IMS/MVS security differ in some respects. Oracle database user ids can be up to 30 characters. OS/390 user ids are limited to seven or eight characters. You have a choice of security schemes when creating user ids for Oracle Access Manager for IMS TM. You can:

- Create new Oracle user ids specifically for Oracle Access Manager for IMS TM-based applications
- Replicate the OS/390 user ids on the Oracle side and adhere to a single user id view
- Use existing Oracle user ids or schema names that differ from what is being used in OS/390

Determining the Oracle User id

Oracle Access Manager for IMS TM accommodates these differences by providing a flexible way to determine the Oracle user id for a given transaction. Oracle user ids are controlled by parameters generated in the RTT. In the RTT, you can specify the type of Oracle user id an IMS application can have:

1. The IMS user id

   For terminal-oriented transactions, this is the signon id if the terminal is signed on. Otherwise it is the IMS LTERM name. For non-message driven batch message processing (BMP), the IMS user id is one of a hierarchy of choices defined by IMS. In many IMS installations, the IMS user id is authenticated by RACF or a similar security product.

2. The IMS program specification program (PSB) name

   The IMS PSB name identifies the IMS application.

3. The application program (load module) name
4. A specific (constant) Oracle user id up to 30 characters

   This choice makes it possible for Oracle Access Manager for IMS TM to accommodate use of any Oracle user id.

If you want to replicate the IMS/MVS user id structure in Oracle, then choose the IMS user id. You can also choose the IMS PSB name and the application program if they fit your installation. You can specify choices on a PSB-by-PSB basis in the RTT generation, and you can designate a default choice for PSBs not explicitly specified.

Session Authentication

An Oracle session created for an IMS transaction using Oracle Access Manager for IMS TM is subject to Oracle's normal authentication mechanisms. These mechanisms are generally controlled by the Oracle database administrator or by system security staff. Oracle supports two kinds of session authentication:

- Password

  Password authentication requires the client (Oracle Access Manager for IMS TM in this case) to supply a password when establishing an Oracle session for a user id.

- EXTERNAL

  EXTERNAL authentication requires the client interface software to verify the user is already authenticated by the client operating environment.

A given Oracle user id is subject to one or the other of the authentication mechanisms, as specified by the DBA or security staff in the CREATE USER or ALTER USER SQL statement. In addition to individual user id specifics, the DBA can specify whether the Oracle instance allows EXTERNAL authentication of sessions coming through Oracle Net. If remote clients (including Oracle Access Manager for IMS TM) are prohibited, then they can use only password-authenticated user ids.

Without this restriction, Oracle Access Manager for IMS TM supports both forms of authentication. It specifies actual Oracle passwords for user ids and supports the client mechanics of EXTERNAL authentication when required by the intended Oracle user id. Oracle Access Manager for IMS TM does not invoke the OS/390 SAF/RACF interface to authenticate the user id before using it with Oracle EXTERNAL authentication. The validity of the user id is an installation responsibility.

Oracle Access Manager for IMS TM does not automatically know an Oracle user id authentication mechanism or password. You specify these on a individual user id basis in the Oracle Access Manager for IMS TM RTT generation. You can provide a default choice for all unspecified user ids.

Oracle Access Manager for IMS TM does not store Oracle passwords in clear text. Oracle server release 7.1 and above transmits logon data (user id and possible password) to the target database in encrypted form.

Error Processing

IMS defines a characteristic called a region error option (REO), which specifies how an ESAF implementation handles non-application processing errors, such as a loss of communications with the external subsystem. The REO has one of three one character values:

R	specifies error code associated with the failure are returned to the application program
Q	specifies the application program is terminated with an abend code U3044. The input transaction is requeued to be processed when access to the subsystem is restored.

A	specifies the application program is terminated with an abend code U3983. The input transaction is discarded. Oracle Access Manager for IMS TM supports all three REO processing options. IMS allows you to specify the REO at the region level in the IMS PROCLIB member defining external subsystems. If you omit the REO, IMS assumes a default value of R.

Oracle Access Manager for IMS TM also provides a mechanism so that you can specify the REO at the application level on the basis of PSB name. An REO specified at the PSB level overrides whatever REO is given or defaulted for the region.

Recovery Considerations

ESAF allows automated recovery after an outage by providing a two-phase commit interface for synchronizing updates. It also provides a process for resolving partially-committed (in-doubt) transactions when connections are reestablished. The IMS control region resolves in-doubt transactions with participation of all ESAF-defined external resources.

To support this recovery, an Oracle Access Manager for IMS TM instance establishes a connection from the control region to the target Oracle database and executes functions to forcibly commit or rollback pending transactions as directed by IMS. The Oracle Access Manager for IMS TM RTT generation must specify a valid Oracle user id under which these recovery actions are performed. The user id does not have full DBA privileges, but does require several privileges that are part of the Oracle DBA role. You can create a new Oracle user id for this purpose or use an existing user id with the required privileges. The RTT generation must provide authentication specifications allowing the recovery user id to connect to Oracle.

Clarification of Cursor Close Behavior

The Oracle server's normal behavior leaves cursors open across COMMIT and ROLLBACK operations. You can use the precompiler MODE option to request application behavior closer to current ANSI standards, including closing of cursors on commit or rollback. The Oracle Access Manager for IMS TM supports this behavior and IMS-initiated commit (GU or SYNC) and rollback (ROLL or ROLB). If MODE is set to ANSI or one of its variations is not specified at precompile time, then Oracle cursors remain open across IMS-initiated commit or rollback. However, if the application is defined in the RTT AMITRANS macro as OID=IMSID and the beginning of a new transaction causes a change in the user id, then cursors are closed and database session state is lost before the new transaction begins. A change in Oracle user id forces the current Oracle session to be deleted and a new one created. It is the responsibility of the application to be aware of this condition and act accordingly.

This situation cannot arise if the AMITRANS macro specifies PSB, PGM, or a fixed user id string as the Oracle user id. It also does not occur if the AMITRANS macro has OID set to IMSID and a new transaction does not convey a change of user id, which might occur when the new input message comes from the same user, LTERM, or from a BMP.

Handling Oracle Unavailable Situations

Oracle unavailable refers to any situation in which the Oracle Access Manager for IMS TM cannot access the target Oracle server. This situation might result from shutdown or failure of a variety of system components including:

- The target Oracle instance
- Oracle Net for OS/390 or a specific protocol adapter
- Other network software (for example, TCP/IP, VTAM)
- Physical network components (routers, for example)
- Remote Oracle Net listener

Oracle Access Manager for IMS TM makes no operational distinction among these situations. It recognizes Oracle is unavailable, perhaps temporarily. There is a difference, however, in whether the condition is detected on the Oracle Access Manager for IMS TM initial connection attempt versus a loss of Oracle access after normal connections are established.

Initial Connection Failure

When IMS starts, if the RTT AMIRT macro CONNECT parameter is set to START, then the control region immediately attempts a connection to the target Oracle server. An Oracle-unavailable condition is detected immediately. On the other hand, if the CONNECT parameter is set to DEFER or defaulted, then the control region waits for the first dependent region to make an Oracle request before attempting its own connection. If the control region is unable to connect to Oracle, then Oracle Access Manager for IMS TM instance is placed in a logical stopped state.

When Oracle Access Manager for IMS TM is in a stopped state after failure of its initial connections, all applications issuing requests for that Oracle Access Manager for IMS TM instance receive a U3049 pseudo-abend and are requeued for later processing. This occurs regardless of the REO option. Even when REO is set to R, an application does not receive an Oracle error when the Oracle Access Manager for IMS TM initial connection attempt fails.

When an Oracle Access Manager for IMS TM instance is placed in the stopped state, it must be started with the IMS command /START SUBSYS after the target Oracle server becomes accessible. This state causes Oracle Access Manager for IMS TM to reinitialize and reattempt a connection to Oracle. If the connection attempt is successful, then queued transactions and new transactions process normally. IMS remembers a subsystem is in stopped state over a warm start. Even if IMS is shutdown and restarted, the /START SUBSYS command is required.

Failure After Initial Connection

If the Oracle Access Manager for IMS TM initial connection attempt from the control region is successful, then a subsequent loss of access is handled without placing the instance in the stopped state. The failure is likely to be detected during execution of an application making Oracle requests. Application behavior in this case is governed by the REO in effect for the transaction, if coded on the AMITRANS macro, or the region from the SSM entry.

If REO is set to R (or taken as a default), then the Oracle error code associated with the lost connection is returned to the application. Applications that use option R must be careful to check SQLCODE or the return code from an OCI call. If the application fails to detect the error and issues another Oracle request, then a loop between IMS and Oracle Access Manager for IMS TM can result. Refer to IBM IMS documentation for more information.

With REO set to Q, an application is abended with U3044, requeued, and the transaction is placed in PSTOP status. With option A, the application receives a U3983 abend and the input transaction is discarded. These abends might cause IMS to invoke Oracle Access Manager for IMS TM to resolve in-doubt processing in the control region. This process also fails if Oracle has become unavailable and IMS holds the associated recovery tokens to be processed later.

These circumstances do not place the Oracle Access Manager for IMS TM instance in the stopped state. Instead, Oracle Access Manager for IMS TM remains active and attempts to reestablish a connection to Oracle when another application makes an Oracle request. If the connection attempt is successful, then Oracle Access Manager for IMS TM resumes normal operation. If it fails, then the application is processed according to the REO in effect.

Each attempt to reestablish the connection to Oracle results in some message traffic to the system console or master terminal operator (MTO) console. A high frequency of failing attempts might result in an excessive message load. A RECONTM parameter on the AMIRT macro can specify the MAXIMUM frequency of reconnect attempts in elapsed second terms. The default for RECONTM is 60 seconds: Oracle Access Manager for IMS TM does not attempt to reconnect on behalf of an application if fewer than 60 seconds have passed since the last attempt. If RECONTM is 0, then every application making an Oracle request causes an attempt to reconnect.

Oracle Environment Variables

Environmental variables control aspects of Oracle product behavior. Environment variables are distinctly-named parameters, set by a mechanism external to the product. Oracle NLS support relies on environment variables to determine the user’s preferred message language, character set encoding, and other locale-sensitive attributes.

In a non-IMS batch job or TSO session, the Oracle9*i* for OS/390 products read a sequential file containing environment variable settings. However, IMS environments prefer different environment variable settings from one transaction to the next in the same region. Sequential file processing might negatively affect performance. Oracle Access Manager for IMS TM specifies environment variables in the RTT generation.

Environment variables are defined in groups in the RTT. Each group is associated with specific transactions (PSBs), specific Oracle user ids, or with the RTT as a whole by coding the group name on the transaction, session, or main RTT definition macros. When an Oracle software component requests the value for a particular variable, Oracle Access Manager for IMS TM checks environment variable groups for:

- Current transaction definition
- Current session definition
- RTT default

These groups are checked in this order to locate a value. The transaction-level specification of a variable has highest priority, followed by a session-level specification, followed by the RTT default.

It is not necessary to provide environment variable definitions at any level if the normal Oracle9*i* for OS/390 defaults are acceptable. The Oracle Access Manager environment does not use the CONNSTR variable, and all variables whose names begin with CRTL_. If these variables are specified, then they are ignored.

Configuring Oracle Access Manager for IMS TM

Oracle Access Manager for IMS TM must be installed successfully before you can configure it. Refer to the *Oracle9i Enterprise Edition Installation Guide for OS/390*, for more information.

Oracle Access Manager for IMS TM Configuration Checklist

- ❏ Step 1: Define an OS/390 Subsystem Identifier for the Instance
- ❏ Step 2: Choose a Value for the Instance and Generate the LIT
- ❏ Step 3: Create a User Id in the Target Oracle Database Used to Conduct Recovery
- ❏ Step 4: Determine the Oracle User Id, Authentication, and Environment Variable
- ❏ Step 5: Code and Generate the Control Region and Dependent Region RTT
- ❏ Step 6: Add a Control Region and Dependent Region SSM Entry for the Instance
- ❏ Step 7: If a New SSM Member is Created for Any Region, Specify the Member to IMS
- ❏ Step 8: Make the Oracle Access Manager Code and Modules Available to IMS Regions
- ❏ Step 9: Shutdown and Restart IMS

When IMS is restarted, the control region reports the status of Oracle Access Manager for IMS TM initialization. The control region automatically initializes Oracle Access Manager for IMS TM in the dependent regions used. Actual connection to the target Oracle database might occur, depending on options specified in the RTT (for example, one option is to defer connection until a transaction actually issues an Oracle request). Transaction programs prepared for use with Oracle Access Manager for IMS TM can execute.

Configuration Steps

Be sure you have read "Integration with IMS" and "Configuration Overview" earlier in this chapter, before beginning the steps in this section.

Step 1: Define an OS/390 Subsystem Identifier for the Instance

Oracle Access Manager for IMS TM requires an OS/390 subsystem id. The id is used as part of an internal communication mechanism. Any valid one to four character subsystem id known to OS/390 can be used as long as it is not used by another subsystem or another instance of Oracle Access Manager for IMS TM.

An IPL normally is required to add new subsystem identifiers to OS/390. However, if the subsystem name you assign is not formally defined, Oracle Access Manager for IMS TM dynamically creates its own entry on the OS/390 subsystem control table (SSCT) chain. It creates the entry by using RTT macro AMIRT DYNSUBS set to YES, which is the default. This entry allows the product to be configured and used without requiring an OS/390 IPL. The interface for adding an SSCT entry is not a published OS/390 interface. If you prefer, you can specify that Oracle Access Manager for IMS TM is not to dynamically create its own SSCT. This is done by specifying the RTT macro AMIRT DYNSUBS be set to NO. In this case, Oracle Access Manager for IMS TM is not able to run until the subsystem name is added and OS/390 is re-IPLed.

Step 2: Choose a Value for the Instance and Generate the LIT

The LIT is a four-character identifier. It is generated using an Assembler language macro (AMILI) and embedded at linkedit time in each IMS transaction program using the associated instance of Oracle Access Manager for IMS TM. The LIT you choose must be unique within the IMS subsystem. It cannot duplicate the LIT of another Oracle Access Manager for IMS TM instance or the LITs associated with other external resources.

The LIT can be identical to the subsystem id. However, an identical LIT can cause problems if you expect to vary the LIT subsystem configuration in the future. For example, if you change the SSM so an existing LIT is associated with a new subsystem, the new LIT might be confused with the old subsystem identifier.

LIT generation is performed by coding and assembling an AMILI macro instruction as follows:

```
[name]AMILI [LIT=lit]
```

where:

`name`	is the CSECT name to use for this LIT generation. If the name is omitted, then the name AMILI000 is used by default. If a name is specified, then ensure it does not conflict with external names occurring in transaction programs. Because only one Oracle Access Manager for IMS TM LIT can be linked into an application, it does not matter if the same CSECT name is used in LITs for different Oracle Access Manager for IMS TM instances.
`LIT=lit`	specifies the one to four character LIT. The value can be enclosed in apostrophes. It must conform to IMS ESAF requirements, such as alphanumeric characters. If this parameter is omitted, a LIT value of ORA1 is generated.

Assembly of the LIT requires access to the Oracle Access Manager for IMS TM macros. The resulting object code can be linkedited into a load module library or saved as an object deck. It must be included in the linkedit of IMS transaction programs using the associated instance of Oracle Access Manager for IMS TM. The LIT contains no executable code and does not have addressing mode (AMODE) or residency mode (RMODE) limitations.

A sample LIT generation job:

```
//AMILIT1 JOB (ORA),'ORACLE LIT GEN'
//  EXEC ASMCL,PARM.LKED='LIST,RMODE=ANY'
//ASM.SYSLIB  DD DSN=oran.orav.MACLIB,
//             DISP=SHR
//ASM.SYSIN    DD *
ORA3LIT  AMILI LIT=ORA3
         END
/*
//LKED.SYSLMOD DD DSN=IMS1.DEV.LIB(ORA3LIT),
//            DISP=SHR
//
```

Step 3: Create a User Id in the Target Oracle Database Used to Conduct Recovery

Oracle Access Manager for IMS TM requires a user id in the target Oracle database so recovery sessions can be established when needed by the control region. Although you can use an existing Oracle user id with the appropriate privileges, Oracle Corporation recommends creating a distinct user id dedicated to this purpose. The user id must have or be granted these privileges:

- CREATE SESSION (system privilege)
- FORCE ANY TRANSACTION (system privilege)
- SELECT ON SYS.PENDING_SESSIONS$ (object privilege)
- SELECT ON SYS.PENDING_TRANS$ (object privilege)

No other privileges are required. If you are creating a user id dedicated to Oracle Access Manager for IMS TM recovery purposes, then Oracle Corporation recommends only these privileges be granted to the user id.

The recovery user id does not create data in the target database. Therefore, RESOURCE privileges and user id tablespace defaults and quotas are unimportant. The user id profile is important because the recovery user id must not be subject to Oracle resource limits that could cause an interruption of recovery activity. Such activity includes the profile IDLE_TIME limit because the recovery session can stand idle for a long time.

The authentication method for the recovery user id can be password or EXTERNAL. When creating a new user id, choose an authentication method based on your Oracle and IMS security practices. The recovery user id is coded explicitly in the RTT; it is not derived from an RTT transaction specification. The control region RTT must contain a session authentication entry allowing the recovery id to connect successfully to Oracle. This entry can be the default session entry if the default authentication method applies to the recovery id. If it does not, an explicit session entry for the recovery user id must be included in the RTT.

If you plan to access a particular Oracle database from more than one IMS subsystem, all IMS subsystems can use the same recovery user, assuming the user id's profile allows at least one session per IMS. However, Oracle Corporation recommends you create a distinct recovery user id for each distinct IMS subsystem. This allows better activity tracking in the Oracle server and simplifies problem resolution.

SQL statements for creating a recovery user id are shown in the following example. They are suitable for use in a Server Manager or SQL*Plus session:

```
CONNECT SYS/CHANGE_ON_INSTALL
CREATE USER IMS1RECO IDENTIFIED BY RECO1PW PROFILE NO_LIMIT;
GRANT CREATE SESSION, FORCE ANY TRANSACTION TO IMS1RECO;
GRANT SELECT ON PENDING_SESSIONS$ TO IMS1RECO;
GRANT SELECT ON PENDING_TRANS$ TO IMS1RECO;
```

The example assumes a profile named NO_LIMIT is already defined to Oracle. The GRANT SELECT statements specify unqualified table names because the connection is with user id SYS, which owns the PENDING_TRANS$ and PENDING_SESSIONS$ tables.

Step 4: Determine the Oracle User Id, Authentication, and Environment Variable

Review the transaction programs you plan to run with this Oracle Access Manager for IMS TM instance to choose the method for determining the Oracle user id for each transaction. When you know what Oracle user ids to use, establish the authentication method for their Oracle sessions. If you are creating new Oracle user ids for Oracle Access Manager for IMS TM applications, you can choose whichever authentication method is appropriate for your installation's security practices.

Finally, consider the user and transaction management requirements to determine environment variable settings at the RTT, session, and transaction level.

Step 5: Code and Generate the Control Region and Dependent Region RTT

Four macros are used to code the RTT:

- AMIRT
- AMITRANS
- AMISESS
- AMIENV

For a summary of RTT macro parameters governing session cache, refer to "Clarification of Cursor Close Behavior" on page 12-10.

AMIRT

The AMIRT macro is invoked once or twice in an RTT definition. The first invocation specifies the connection string for the target Oracle database and other options with subsystem-wide or region-wide scope. The connection string for the target Oracle database is meaningful only in the RTT used by the control region. The address can be omitted in a dependent region RTT. If it is specified, then it must be identical to the one specified for the control region. Otherwise, Oracle Access Manager for IMS TM initialization for the dependent region fails.

The AMIRT macro is coded using this syntax:

```
[name] AMIRT [DBADDR='string']
[CONNECT={START|DEFER}]
[RECOID='string']
[DYNSUBS={YES|NO}]
[ENVTAB=envname]
[END={YES|NO}]
```

where:

`name`	can be any name allowed by the assembler. It is ignored.
`DBADDR='string'`	specifies the Oracle connection string of the target Oracle database. This string must be a complete address string (not a TNSNAMES alias identifier). Refer to the example in the note at the end of this `DBADDR='string'` description. The address is normally enclosed in apostrophes because it can contain special characters such as blanks and parentheses. This parameter can be omitted in a dependent region RTT and when coding AMIRT END set to YES to conclude an RTT definition.

If accessing a remote database, Oracle Net address strings can be lengthy. You need to code continuations of the line on which DBADDR is specified. Remember the assembler normally requires a nonblank continuation indicator in position 72 and the continuation begins in position 16 of the next record. Positions 1-15 must be blank.

The easiest way to determine what to specify for an Oracle Net address string is to look in an existing TNSNAMES configuration file. Refer to Chapter 10, "Oracle Net", for more information.

Note: The DESCRIPTION keyword is required. For example, for a local database:

```
AMIRT DBADDR='(DESCRIPTION=
(ADDRESS=(PROTOCOL=XM)(SID=QA74)))'
```

or, for a remote database:

```
AMIRT  DBADDR='(DESCRIPTION=(ADDRESS=
(PROTOCOL=TCP) (HOST=144.25.40.217)
(PORT=1521) (SSN=TNS)
(CONNECT_DATA=(SID=QA74))))'
```

`CONNECT={START|DEFER}`	specifies whether the region is to establish the connection to Oracle at region startup (START) or wait until the first Oracle access request is made by a transaction program (DEFER). The default is DEFER. This option applies to both control and dependent (MPP) regions. However, the control region is required to establish a connection before any dependent region can connect. CONNECT set to DEFER for the control region serves no purpose if any dependent region immediately uses CONNECT set to START. This parameter is ignored for BMP and IMS fast path (IFP) regions, which always operate with the equivalent of CONNECT set to DEFER.
`RECOID='string'`	specifies the Oracle user id used for IMS recovery activity. During Oracle Access Manager for IMS TM startup processing, the control region accesses Oracle using this id if there are pending uncommitted transactions. An AMISESS macro specifying this id or a default entry allowing this id to logon to Oracle, must be included in the control region RTT. This parameter is ignored in a dependent region RTT.
`DYNSUBS={YES|NO}`	specifies whether Oracle Access Manager initialization for the control region is permitted to create the Oracle Access Manager subsystem name entry if the name is not defined formally to OS/390. (Refer to the discussion of this topic in "Step 1: Define an OS/390 Subsystem Identifier for the Instance" on page 12-15.) The default for this parameter is YES. The parameter is ignored in a dependent region RTT.

`ENVTAB=envname`	specifies the name field of an AMIENV macro coded in the same RTT generation. It must conform to Assembler name syntax requirements. The environment variable specifications in the named AMIENV are used in all Oracle Access Manager processing unless overridden by an AMIENV set specified in an AMITRANS or AMISESS macro in this RTT generation.
`END={YES|NO}`	specifies the end of the RTT definition. When the RTT contains one or more uses of the AMITRANS or AMISESS macros, the AMIRT macro is invoked once at the beginning (where DBADDR is specified) and once at the end with only END set to YES specified. In an RTT definition containing no AMITRANS or AMISESS macros, AMIRT can be invoked a single time with parameters and END set to YES combined.

AMITRANS

The AMITRANS macro assigns characteristics to IMS applications by PSB names. It determines the Oracle user id that causes a transaction's Oracle access. The Oracle user id assignment can be a fixed value (for example, SCOTT) or it can be one of three dynamic values associated with the transaction instance. You can assign different PSB names to different user id determinations. A default determination can be specified for PSB names that do not have an individual entry.

An RTT definition does not need to include AMITRANS macros. The RTT definition assumes a single default user id determination method for all transactions that run in a region without macros. AMITRANS macros used in an RTT definition, must appear after the first AMIRT macro. AMIRT with END set to YES must also appear at the end of the RTT definition.

The characteristics of one or more individual transactions can be specified in a single use of AMITRANS.

The AMITRANS macro is coded as:

```
[name]   AMITRANS  PSB=(psb1,...)
   OID={IMSID|PSB|PGM|'string'}
 [REO=c]
[ENVTAB=envname]
```

where:

`name`	can be any name allowed by the assembler. It is ignored.
`PSB=(psb1...)`	specifies the IMS PSB names of the applications to which a common set of characteristics are being assigned. A name specification of * designates a default for transactions for which no specific entry is given in any AMITRANS macro. The * can be included on an AMITRANS call also listing specific PSB names. No more than one * entry can be created in a single RTT definition and no specific PSB name can be repeated in an RTT.
`OID={IMSID|PSB|PGM|string}`	specifies the method of determining the Oracle user id for the transactions listed. The choices are:
`IMSID`	is the OS/390 and IMS authorization id (or a substitute) used.
`PSB`	is the IMS PSB name used.
`PGM`	is the program name used.
`string`	is the fixed character string used. This value can be enclosed in apostrophes. It must be a valid user id in the target Oracle database and must conform to the content rules for Oracle user ids.

`REO=c`	specifies the IMS region error option used with the associated transactions. This option is coded as a single letter. Permissible values and their meanings are discussed in Error Processing.
`ENVTAB=envname`	specifies the name of an AMIENV macro coded in the same RTT generation. It must conform to Assembler name syntax requirements. The environment variable specifications in the named AMIENV are used in all Oracle Access Manager processing for the indicated transactions. They take precedence over specifications of the same variable in environment tables associated with the AMISESS and AMIRT macros.

When OID is set to IMSID, determination of the user id varies with the environment. For message-driven transactions:

- Use the RACF-validated signon id if the terminal originating the transaction is signed on.
- Use the LTERM id if the terminal is not signed on

In a non-message-driven region:

- If the address space is present, use the ASXBUSER field.
- If the address space is not present, use the IMS PSB name.

If an RTT contains no AMITRANS macros, the region operates with this default:

```
AMITRANS PSB=*,OID=IMSID
```

This statement specifies that all transactions use the MVS/IMS user id (or a substitute) as the Oracle logon id. If the RTT contains any AMITRANS macros, this default is not assumed. If the RTT contains one or more AMITRANS macros and you also want to specify default transaction characteristics, you must add an AMITRANS for PSB=* or * to the PSB list of an existing AMITRANS macro call in the RTT definition.

AMISESS

The AMISESS macro is used to indicate, by Oracle user id, the type of logon authentication to use and, if required, the Oracle logon password.

An RTT definition does not need to include any AMISESS macros. In this case, a default set of session characteristics is assumed for all sessions created by the region. If any AMISESS macros are used in an RTT definition, they must appear after the first AMIRT macro. A second use of AMIRT with END set to YES is required at the end of the RTT definition.

The session characteristics of one or more Oracle user ids can be specified in a single use of AMISESS.

The AMISESS macro is coded as:

```
[name]   AMISESS  OID=('string' [,'string'...])
    AUTH={EXTERNAL|'string'}
    [ENVTAB=envname]
```

where:

`name`	can be any name allowed by the assembler. It is ignored.
`OID=('string' [,'string...])`	specifies the Oracle user ids whose session characteristics are being described. A user id specification of * designates a default for user ids for which no specific entry is given. The * can be included on an AMISESS call also listing specific user ids. No more than one * entry can be created in a single RTT definition.

`AUTH={EXTERNAL|` `'string'}`	specifies how the user id is authenticated at Oracle logon. EXTERNAL indicates Oracle is to assume the user is already authenticated by IMS or OS/390. In this case, no password is sent to Oracle at logon time. Oracle verifies the user id is known and is created with the IDENTIFIED EXTERNALLY option. If the connection to Oracle is through Oracle Net, the Oracle instance must be configured to allow such connections through Oracle Net. **Note:** If EXTERNAL is specified, it is important to review the LOGON_AUTH setting in the Database Service definition. The alternative to EXTERNAL is to specify the Oracle logon password. The value specified can be enclosed in apostrophes and must match the Oracle user id password. Only one password can be specified per AMISESS macro, so an AMISESS macro with more than one OID value associates the same password with all of the user id. **Note:** RTT passwords are stored in an encrypted form that can be decrypted. It is the user's responsibility to secure the RTT module, for example, by RACF-protecting the RESLIB library in which the RTT is stored.
`ENVTAB=envname`	specifies the name field of an AMIENV macro coded in the same RTT generation. It must conform to Assembler name syntax requirements. The environment variable specifications in the named AMIENV are used in all Oracle Access Manager processing for the indicated sessions unless overridden at the AMITRANS level.

If an RTT contains no AMISESS macros, the region operates with this default:

```
AMISESS OID=*,AUTH=EXTERNAL
```

This default specifies that all transactions connect to an Oracle instance using the external authentication mechanism. Whatever Oracle user id that a transaction uses must be known to the target Oracle instance and have the IDENTIFIED EXTERNALLY attribute. If the connection uses Oracle Net, the target Oracle instance must be configured to permit externally authenticated logons through Oracle Net.

If the RTT contains any AMISESS macros, this default is not assumed. If the RTT contains one or more AMISESS macros and you also want to specify default session characteristics, you must add an AMISESS for OID=* or * to the OID list of an existing AMISESS macro call in the RTT definition.

AMIENV

The AMIENV macro defines values for various Oracle software environment variables, particularly NLS support. One AMIENV macro call defines a distinct set of variable name and value pairs. The AMIRT, AMITRANS, and AMISESS ENVTAB parameters allow a given set to be associated at the region, transaction, or session level. When a variable is specified at multiple levels, the order of precedence is AMITRANS, AMISESS, AMIRT.

AMIENV allows you to specify any environment variable names. The names and values are not validated during the RTT generation. Do not misspell the name of a variable because AMIENV treats it as though the variable is not specified. An erroneous value for an environment variable becomes apparent at runtime when the software attempts to use the value. How the error is reported depends on the variable and specified value.

In the RTT generation, all AMIENV macros must appear after the first AMIRT macro. They can be mixed with AMITRANS and AMISESS macros and, unlike AMITRANS and AMISESS, also can appear after the END set to YES call to AMIRT.

The AMIENV macro is coded as:

```
name AMIENV (n1,v1,...)
```

where:

`name`	is a name field conforming to Assembler rules. The name field is required and uniquely identifies the set of variables within the RTT. It is specified as the ENVTAB parameter of an AMIRT, AMITRANS, or AMISESS macro in the RTT.
`n1`	is an environment variable name, optionally enclosed in apostrophes. Environment variable names used by Oracle products are generally all uppercase.

`v1`	is the value assigned to the environment variable, optionally enclosed in apostrophes. The apostrophes are required if the value includes characters such as blanks or punctuation that are not part of the Assembler's name syntax. Up to 256 name and value pairs can be specified within the parentheses. A given variable name can appear no more than once in a single AMIENV macro.

Step 6: Add a Control Region and Dependent Region SSM Entry for the Instance

If you access external subsystems from IMS, then you already have an SSM parameter file. If not, you must create one. The parameter file is a member of the IMS PROCLIB data set. Its member name is in the form:

```
<imsid><SSM_suffix>
```

where:

`<imsid>`	is the IMS system id.
`<SSM_suffix>`	is a one to four character suffix you choose. The suffix is passed to IMS using the SSM startup region parameter.

IMS TM Version 4.1 introduces a new keyword syntax for SSM entries. This syntax is not supported by Oracle Access Manager for IMS TM. You must code the SSM entry using IMS/DC Version 3.1 positional syntax, which is accepted by both IMS versions.

An Oracle Access Manager for IMS TM entry in the parameter file such as IMSLORA0, is a single logical record:

```
ssn,lit,ORAESSD,rtt,reo,crc
```

where:

`ssn`	is the one to four character subsystem id for this Oracle Access Manager for IMS TM instance. This parameter is required and must begin in the first position of the record.
`lit`	is the one to four character LIT for this Oracle Access Manager for IMS TM instance. This parameter is required.

`ORAESSD`	is the name of Oracle Access Manager for IMS TM external subsystem module table (ESMT). This parameter is required and must be specified as shown.
`rtt`	is the load module name for the generated Oracle Access Manager for IMS TM RTT. This module is placed in the IMS RESLIB or in a data set concatenated to RESLIB. This parameter is required in the control region SSM and optional in a dependent region SSM. If omitted in a dependent region SSM, the region uses the RTT specified for the control region.
`reo`	is the region error option, a one character value specifying how Oracle Access Manager for IMS TM is to handle external subsystem failures during application request processing. It includes the situation where an application issues a request before a connection to the external subsystem can be made. This parameter is optional. If omitted, the IMS default is R. Oracle Access Manager for IMS TM also allows the REO to be specified on a PSB-by-PSB basis in the RTT. The REO specified at the PSB level overrides this REO.

The allowed values are:

- `R` returns an Oracle error code to the application
- `Q` abnormally terminates the application with a U3044 abend code. The input transaction is requeued to be processed when an external subsystem connection becomes possible.
- `A` abnormally terminates the application with a U3983 abend code. The input transaction is discarded.

`crc`	is an optional command recognition character. This parameter is not used and can be omitted.

Step 7: If a New SSM Member is Created for Any Region, Specify the Member to IMS

If in Step 6 you created a new SSM member, you must specify the member to IMS. Both the control and dependent regions use the SSM keyword parameter for this purpose. The value for the parameter is the one to four character `<SSM_suffix>` discussed in Step 6.

The SSM parameter can be specified in the execute procedures for the control region or dependent regions and can be specified in the JCL for batch message processing jobs. For IMS Version 4 or higher, it can be included in the operator command for /START SUBSYS. Refer to the IBM *IMS/ESA System Definition Reference* for more information on the coding of the SSM parameter.

Step 8: Make the Oracle Access Manager Code and Modules Available to IMS Regions

If this is the first Oracle Access Manager instance you have configured for this IMS subsystem, you must make the Oracle Access Manager load modules available to the IMS control and dependent regions. Place the Oracle Access Manager modules in the concatenation path of two DD statements in each region: STEPLIB and DFSESL. The DFSESL concatenation must consist of APF authorized libraries and must include the IMS RESLIB data set in addition to the data set containing Oracle Access Manager for IMS TM modules.

All RTTs must be in a data set that is part of STEPLIB and DFSESL.

You must copy the AUTHLOAD and MESG modules that are required by the Oracle Access Manager to another authorized library that is part of each region’s STEPLIB and DFSESL. Oracle's AUTHLOAD cannot be used because IMS does not support the PDSE format.

In addition to your generated RTTs, the following modules are required:

- ORAAMSD (main code module)
- ORAESSD (external subsystem module table)

You must also copy the following members from Oracle’s MESG library:

- AMIxx (Oracle Access Manager messages)
- ORAxx (Oracle server messages)
- SQLxx (Precompiler runtime messages)

- OCIxx (Oracle call interface messages)
- LXxxxxx (character set data objects, where xxxxx corresponds to different types of data objects. All must be copied.)
- TNSxx (TNS messages)

In addition, if you are accessing a remote Oracle instance through Oracle Net, you need the NLxx message modules.

The *xx* appearing in the message module names is a language indicator. For American English messages, *xx* is US. If you are using the NLS_LANG environment variable to choose other languages, you must include the appropriate message modules for those languages. Refer to Appendix D, "National Language Support" for a discussion of NLS considerations.

Note: Access Manager for IMS messages are only available in American English. If NLS_LANG is not American English, the message number for the Access Manager for IMS message is used without text.

Step 9: Shutdown and Restart IMS

When appropriate, follow your installation's normal procedure to shutdown and restart the IMS subsystem. You can use a warm start or cold start.

If your configuration is successful, you see message AMI-0108 displayed at the IMS MTO console once for each region that is permitted to access the new Oracle Access Manager instance. If the RTT for a region specifies CONNECT set to START (on the AMIRT macro), you also see message AMI-0113 indicating a connection is established to the target Oracle instance. Otherwise, IMS delays the connection attempt until a transaction issues a request.

If the connection attempt fails for any reason, an error message is displayed and IMS places the Oracle Access Manager instance in a stopped state. Once you have corrected the problem, you can restart the Oracle Access Manager instance by issuing the IMS /START SUBSYS <ssn> command.

For more information on operating the Oracle Access Manager, refer to "Starting and Stopping Oracle Access Manager for IMS TM" on page 12-33.

IMS External Subsystems

IMS refers to Oracle Access Manager for IMS TM as an external subsystem. IMS requires an OS/390 subsystem identifier for each instance of Oracle Access Manager for IMS TM you configure. However, the product does not run as a separate OS/390 address space. Instead, the Oracle Access Manager code runs inside the IMS control region and in each dependent region that accesses Oracle data. A separate address space is required for an Oracle9*i* for OS/390 database or for Oracle Net for OS/390. Oracle9*i* for OS/390 or Oracle Net for OS/390 is used when running Oracle Access Manager for IMS TM. Refer to Chapter 10, "Oracle Net" for Oracle Net for OS/390 operating considerations.

The IMS control region is responsible for monitoring the status of all external subsystems, including all configured instances of Oracle Access Manager for IMS TM. It views each instance as being stopped or started. You can query IMS to determine the status of a subsystem using the IMS operator command DISPLAY SUBSYSTEM. Refer to IBM IMS documentation for a detailed description of the DISPLAY SUBSYSTEM command.

For example, to request the status of a subsystem with the identifier AMI1, issue the IMS command:

```
/DIS SUBSYS AMI1
```

from the IMS master terminal operator (MTO) or OS/390 system console. IMS might respond with something like:

```
DFS000I SUBSYS  CRC   REGID   PROGRAM    LTERM     STATUS
DFS000I AMI1    #                                  NOT CONN
```

In this example, the Oracle Access Manager for IMS TM instance AMI1 is not currently connected to the target database. If the subsystem is started and active in several regions, then the display might look like:

```
DFS000I SUBSYS  CRC    REGID   PROGRAM  LTERM     STATUS
DFS000I AMI1    #          1   PRGL08   HN1LN006  CONN
DFS000I AMI1    #          3   PRGL15Q  HN1LN083  CONN
DFS000I AMI1    #          4   PSVB     HN0LN19A  CONN
DFS000I AMI1    #         12   PRGNQ    HN1LN072  CONN
DFS000I AMI1    #         14   PBNC2UP            CONN
```

Although Oracle Access Manager for IMS TM does not have a display command, it does display status messages under circumstances such as:

- Startup
- Termination
- Recovery activities
- Errors

These messages always begin with the prefix AMI- followed by a unique message number. The complete set of Oracle Access Manager for IMS TM messages are documented in the *Oracle9i Enterprise Edition Messages Guide for OS/390*.

All Oracle Access Manager for IMS TM messages are written to the operator console and the MTO.

Starting and Stopping Oracle Access Manager for IMS TM

When the IMS control region is started, IMS attempts to initialize all external subsystems configured for that region. This means initializing all external subsystems the IMS subsystem can access. Dependent regions can initialize fewer or even no subsystems, depending on how they are configured.

Initialization of Oracle Access Manager for IMS TM in a given region validates configuration data and allocates virtual memory for required data structures. It does not necessarily attempt to make a connection to the target Oracle server. An Oracle Access Manager for IMS TM configuration option can specify connection to the server be deferred until an IMS application makes a request for Oracle data. This option can be set on a region-by-region basis.

There are connection dependencies. IMS requires the control region establish a connection before any dependent region connection is allowed. If the deferred connection option is used for the control region but a dependent region starting immediately does not defer connection, then the control region is forced to connect immediately.

Regardless of whether connection is made immediately or deferred until the first Oracle request, there are a variety of reasons why the Oracle Access Manager for IMS TM connection process might fail. The target Oracle server might be down. If the target server is remote, then any of a number of required components, such as Oracle Net for OS/390, networking software or hardware, or Oracle Net on the target platform, might be down.

If the connection process fails, then Oracle Access Manager for IMS TM displays error information and informs IMS. IMS places the subsystem in the logical stopped state. If the connection is deferred and is associated with an application request for data, then IMS terminates and requeues the requesting transaction.

When Oracle Access Manager for IMS TM connection fails and the subsystem is stopped, it remains so until the operator issues IMS command START SUBSYSTEM.

Oracle Access Manager for IMS TM does not automatically notify IMS to start when the cause of connection failure is resolved. When the START SUBSYSTEM command is issued, Oracle Access Manager for IMS TM reinitializes in each region. Connection to the target Oracle server is attempted or deferred as indicated in configuration data, just as in the original region startup.

Failures and Recovery

Normal IMS and Oracle Access Manager processing can be interrupted by a failure of any of the hardware and software components involved. Because Oracle Access Manager for IMS TM supports IMS two-phase commit synchronization, active transactions at the time of a failure are one of the following:

- Completely committed
- Completely rolled back
- Left in a recoverable state called in-doubt

When problems are rectified and normal IMS and Oracle Access Manager for IMS TM processing resumes, the IMS control region performs a resolve in-doubt process. This process determines the status of each in-doubt transaction and commits it or rolls it back accordingly. Oracle Access Manager for IMS TM displays messages during this process, indicating the action taken with each in-doubt transaction.

IMS also performs resolve in-doubt processing any time an application abnormally terminates. Therefore, the same messages can be expected after an abend in a transaction using Oracle Access Manager for IMS TM.

From the Oracle server side, transaction activity from Oracle Access Manager for IMS TM falls into the same general class as Oracle distributed transactions. The same internal Oracle tables and views record the status of all such transactions. Transactions that originated in Oracle Access Manager for IMS TM can be identified by their parent database id. Oracle Access Manager for IMS TM constructs a parent database id using a combination of the IMS id and the Oracle Access Manager for IMS TM subsystem id. Detailed information on these tables and views can be found in *Oracle9i Database Reference.*

13

Oracle Enterprise Manager Intelligent Agent and Data Gatherer

This chapter introduces you to Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390, and provides configuration information.

The following topics are included:

Overview

The Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390 is a component of the Oracle9*i* for OS/390 product set. It runs jobs and events sent by the Oracle Enterprise Manager and can start up and shut down a database. Additionally, it can function regardless of the status of the network connection, and it can run even if the database is down.

The Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390 is implemented as a command under OS/390 Unix System Services (USS).

Although an Oracle server is not required to run Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390, the agent's primary function is to monitor an Oracle server or servers. Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390 can monitor only servers that are running Oracle8 release 8.0.4 or later. Chapter 18, "Migration and Upgrade Considerations", covers issues related to the compatibility of Oracle9*i* for OS/390 with previous releases.

Running the Customization Script

If you chose to install Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390, you will need to run an interactive script named customize.sh. The script can be run at any time after the installation by performing the following steps:

1. Set your $ORACLE_HOME environment variable to the value specified at installation. Refer to the *Oracle9i Enterprise Edition Installation Guide for OS/390* for more information.

Note: $ORACLE_HOME must have the correct value before customize.sh is run. Note also that several product directories might contain scripts named customize.sh. The version located in $ORACLE_HOME is the correct one.

2. Issue the following commands:

```
> cd $ORACLE_HOME
> ./customize.sh
```

3. You will be prompted to set up the OEM backup utility. If you answer Yes (Y), then:
   a. You will be prompted to provide the data set high-level qualifier of your database.
   b. You will be prompted to provide a dsname that will be used for creating temporary RMAN cmdfile data sets.
   c. You will be prompted to provide a dsname that will be used to store the TSO REXX data set.
4. The customize.sh script generates several export statements that can be used to set all environment variables necessary for running the Intelligent Agent and Data Gatherer. You will be prompted by the customize.sh script to choose whether these exports are to be added to your .profile, or written to a separate file for you. You must ensure that the specified environment variables are set before you run the Intelligent Agent and Data Gatherer.

Updating the Parameter Files

After you have run the script, you must configure Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390 to access the databases you wish to manage. You do this by updating several parameter files that are placed on your system by the install process. A working copy of these files is placed into the $ORACLE_HOME/network/admin directory, and a backup copy is placed in the $ORACLE_HOME/network/admin/sample directory. Leave the backup copy unchanged, and edit only the working copy in the $ORACLE_HOME/network/admin directory. The contents of all of these files are documented in the *Oracle9i Net Services Book Set*.

To simplify the configuration process, the installed parameter files use several easily-located character strings to indicate places where you must provide installation-specific information. To configure the Agent, edit each of the parameter files in turn, issue global change commands with your editor to update each of the strings, and save the updated file. The agent reads the parameter files as it starts up and in this way manages the specified databases. Note that the parameter files as installed are designed to manage only a single Oracle instance; refer to the *Oracle9i Net Services Book Set* to learn how to update the parameter files to manage multiple instances. If you make any changes to the contents of these files, the Agent will need to be stopped and restarted before the changes will take effect.

The strings used in the parameter files are as follows:

%DB%	is the Oracle SID of the Oracle for OS/390 instance that you wish to access.
%DBH%	is the host name of the managed Oracle for OS/390 instance. This value can be different from %HOST%.
%DBP%	is the port number of the Oracle for OS/390 instance you are managing.
%DBT%	is the Net service name for the TCP connect descriptor.
%DBX%	is the Net service name for the cross memory connect descriptor.
%HOST%	is the host name where the listener is running.
%PORT%	is the port number of the listener.
%SERV%	is the service name for the Oracle for OS/390 instances you wish to access.
%SS%	is the subsystem ID of the Oracle for OS/390 instances that you wish to access. It must match the value that is specified for SUBNAME on the SETSSI command for your OSDI subsystem.
%TNS_ADMIN%	is the value of $ORACLE_HOME with /network/admin subdirectories appended to it. Be sure to supply the value of $ORACLE_HOME rather than the literal string.

After the configuration files in $ORACLE_HOME/network/admin have been updated, test the Agent. First, run the export statements created by the customize.sh script to set the environment variables as required. Then, test connectivity with the tnsping command; only when tnsping is successful should you attempt to use the Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OS/390.

Controlling Operations of the OS/390 Agent

The Oracle service (OS/390 started task) must be started prior to starting the agent. The Oracle Enterprise Manager Intelligent Agent and Data Gatherer requires an additional tnsnames.ora entry for cross memory database access. That entry is included in the sample tnsnames.ora file and must be updated to include the correct subsystem ID and service name.

Note: The value of the HOST parameter in both listener.ora and tnsnames.ora must be a host name, not an IP address. While other utilities, such as tnsping, will accept either a host name or an IP address, the agent parses entries in listener.ora, tnsnames.ora, and oratab to create services.ora, which contains the list of all valid entries. If your HOST parameter values are all IP addresses, your services.ora will be empty, and the agent will fail trying to allocate 0 bytes of memory.

Caution: Do not modify the TCL scripts (job and event scripts written in Tool Command Language) that come with the agent. If you want to submit a job different than the ones that are predefined with the agent, then use the TCL job where you are allowed to pass in arbitrary scripts and have the agent run them.

14

Oracle9*i* Real Application Clusters

This chapter provides guidelines for configuring Oracle9*i* Real Application Clusters after installing Oracle9*i* for OS/390. The following topics are included:

Overview

With the Oracle9i Real Application Clusters option, separate Oracle9*i* for OS/390 instances run simultaneously on one or more nodes (OS/390 images) and share a single physical database.

It is important to note that the OS/390 implementation follows the Oracle9i Real Application Clusters design guidelines. The only difference is the use of OS/390 filetypes. For more information, refer to the *Oracle9i Real Application Clusters Book Set*.

Oracle9i Real Application Clusters has the following characteristics:

- One or more instances of Oracle9*i* Real Application Clusters can be started on a given node in the parallel sysplex.
- Each instance has a separate SGA and set of background processes.
- All instances share the same data files and control files of a given database.
- All instances can execute transactions concurrently against the same database and each instance can have multiple users executing transactions.
- Row level locking is preserved across the instances.

Applications accessing the database can run on the same nodes as multiple instances of Oracle9*i* Real Application Clusters, or on separate nodes of a parallel sysplex, or on distributed systems using the client/server architecture. Oracle9i Real Application Clusters can be part of a distributed database system. Distributed transactions access the data in a remote database in the same way, regardless of whether the data files are owned by an Oracle9i for OS/390 server (in exclusive mode) or an Oracle9i Real Application Clusters instance (in exclusive or parallel mode).

Other non-Oracle workload can run on each node of the sysplex or you can dedicate the entire sysplex or part of the sysplex to the Oracle server. For example, an Oracle9*i* Real Application Clusters instance and its applications might occupy three nodes of a five-node sysplex, while the other two nodes are used for non-Oracle applications.

Refer to the *Oracle9i Real Application Clusters Book Set* for more information about the Oracle9*i* Real Application Clusters environment. Refer to the concepts in the *Oracle9i Real Application Clusters Book Set*, especially the discussions on compatibility issues, restrictions, and database design guidelines.

Setting Up Real Application Clusters

The following steps describe important tasks you must perform when setting up an Oracle9*i* Real Application Clusters system, but they do not include all the tasks you might need to do for your site. Before performing these steps, review the concepts manual in the *Oracle9i Real Application Clusters Book Set* for more information.

Checklist for Setting Up Oracle9*i* Real Application Clusters:

1. Review and set options in the CREATE DATABASE statement
2. Set up additional threads of redo log files and rollback segments
3. Share all non-VSAM data sets
4. Set up the common OSDI parameters
5. Set up the common Oracle9*i* Real Application Clusters initialization parameters
6. Set up the instance-specific Oracle9*i* Real Application Clusters initialization parameters
7. Create the startup JCL

Step 1: Review and Set Options in the CREATE DATABASE Statement

Review and set the following options to allow multiple instances of Oracle9*i* Real Application Clusters to function properly. For a complete description of the CREATE DATABASE and ALTER DATABASE statements and options, refer to the *Oracle9i Database Administrator's Guide* and the *Oracle9i SQL Reference*.

MAXDATAFILES The MAXDATAFILES option of CREATE DATABASE determines the number of data files a database can have. With Oracle9*i* Real Application Clusters, databases tend to have more data files and log files than an exclusive mounted database.

MAXINSTANCES The MAXINSTANCES option of CREATE DATABASE limits the number of instances that can access a database concurrently. The default value for this option under OS/390 is 15. Set MAXINSTANCES to a value greater than the maximum number of instances you expect to run concurrently.

MAXLOGFILE and MAXLOGMEMBERS The MAXLOGFILES option of CREATE DATABASE specifies the maximum number of redo log groups that can be created for the database. The MAXLOGMEMBERS option specifies the maximum number of members or number of copies per group. Set MAXLOGFILES to the maximum number of instances you plan to run concurrently multiplied by the maximum anticipated number of groups per thread.

MAXLOGHISTORY The MAXLOGHISTORY option of CREATE DATABASE specifies the maximum number of redo log files that can be recorded in the log history of the control file. The log history is used for automatic media recovery of Oracle9*i* Real Application Clusters.

For Oracle9*i* Real Application Clusters, set MAXLOGHISTORY to a large value, such as 100. The control file can then store information about this number of redo log files. When the log history exceeds this limit, the Oracle server overwrites the oldest entries in the log history. The default for MAXLOGHISTORY is 0 (zero), which disables log history.

Step 2: Set Up Additional Threads of Redo Log Files and Rollback Segments

Each Oracle9*i* Real Application Clusters instance requires its own redo log threads and rollback segments. Refer to the concepts manual of the *Oracle9i Real Application Clusters Book Set* for how to plan, create, and assign threads of redo and rollback segments to each instance.

Step 3: Share All Non-VSAM Data Sets

Because non-VSAM data sets will be used by multiple instances, ensure that the data sets used in the startup JCL of Oracle9*i* Real Application Clusters instances are specified with a DISP=SHR JCL parameter.

Step 4: Set Up the Database Region Parameters

Review and modify the OSDI database region parameters that apply to individual Oracle9*i* instances in one of the members of PARMLIB (for example, *sid*PARM, where *sid* is the database service identifier). OSDI database region parameters are described in "Database Region Parameters" on page 3-10. These parameters are required to start individual Oracle9*i* instances. There are no Oracle9*i* Real Application Clusters-specific database region parameters.

Step 5: Set Up the Common Oracle9*i* Real Application Clusters Initialization Parameters

Review and modify the Oracle9*i* Real Application Clusters initialization parameters that apply to all instances in the *sid*INIT member of PARMLIB. The *sid*INIT member might also contain other parameters which are not directly applicable to Oracle9*i* Real Application Clusters but apply to all instances. For a complete list of parameters common or identical across all instances, refer to the *Oracle9i Real Application Clusters Book Set*.

Step 6: Set Up the Instance-specific Oracle9*i* Real Application Clusters Initialization Parameters

Create the Oracle9*i* Real Application Clusters initialization parameters that must be unique for each instance in a new member of PARMLIB. You may create as many of these members for as many instances as you plan to run mounting the same database. Each of these members also should specify the common initialization parameter file using the IFILE parameter.

Alternately all parameters can be specified in one member of PARMLIB. For syntax and notation required for using one initialization parameter file, please refer to the *Oracle9i Real Application Clusters Book Set*.

At a minimum, the INSTANCE_NUMBER, INSTANCE_NAME, and THREAD parameters are recommended for each instance.

Step 7: Operating an OSDI Database Service

Each instance of Oracle9*i* Real Application Clusters executes in a manner similar to a single OSDI-based database service. For more information, refer to Chapter 3, "Configuring a Database Service and Creating a New Database" and Chapter 5, "Operating a Database Service".

Before starting or stopping Oracle9*i* Real Application Clusters instances or doing backup and recovery operations, refer to the information about these topics in the *Oracle9i Real Application Clusters Book Set*.

Cross System Communication Facility (XCF)

Oracle9*i* Real Application Clusters requires the Cross System Communication Facility, commonly known as XCF. For more information about configuring XCF, please refer to the IBM document *Setting Up a Sysplex*, GC28-1779.

OS/390 Resource Name Usage

Oracle9*i* Real Application Clusters instances will, by default, make use of the following OS/390 resource names. If these names conflict with other software on your system and this cannot be changed, contact Oracle Worldwide Support for assistance.

XCF Group Names

Each instance will create and join XCF groups with names of the format Onnnxxxx, where `nnn` is an internally generated 3 character alphanumeric string to uniquely identify a cluster, and `xxxx` is a 4 character alphanumeric string to identify the type of XCF group. The following is an example using XCF names for the first Oracle9*i* Real Application Clusters instance:

```
XCF group for IPC Messages: O001ORAM
XCF group for Node Monitor: O001ORAN
Oracle Groups: O001Gmmm
```

In the above example, the *mmm* are Oracle group numbers and are assigned serially.

OS/390 ENQ Names

Each instance will also take out global (sysplex-wide) enqueues on resources with names in the format:

```
Qname : OnnnORAG
```

where *Qname* `is the` queue name to be used for ENQs used by this cluster.

```
Rname : OnnnGxxx#yyy….yyy
```

where *Rname* is the resource name used by ENQs consisting of *xxx* (a three character alphanumeric string) and *yyy*…*yyy* (a variable length alphanumeric string, up to 18 characters).

For example, for the first Oracle9*i* Real Application Clusters instance with the database name ORACLE9I, the following name will be used:

```
O0001ORAG.O001G001#DBORACLE9I
```

15 Oracle Programmer

Oracle Programmer under OS/390 includes OS/390 task and program requirements that must be considered for Oracle applications running in special environments.

Most Oracle applications run as conventional CICS transactions, TSO CALLs, TSO commands, or batch jobs. The OS/390 task and program requirements are usually only an issue for applications that use:

- Multitasking
- Cross memory services
- Authorized OS/390 functions
- Execution in supervisor state
- Execution in a PSW key other than 8

Some examples of system software facilities that use special environments are subsystem-based products, exits from OS/390 system functions, and transaction processing monitors.

The following topics are included:

Oracle Programmer INCLUDE Files

Oracle Corporation provides several INCLUDE files in the H library for use in the Oracle Precompiler and Oracle Call Interface programs. You do not have to specify the member name that includes the language suffix in your source program. However, Oracle Corporation recommends you specify the data structure name, such as SQLCA, in your source program. The Oracle Precompilers includes the appropriate member name based on the Oracle Precompiler language.

Table 15–1 Data Structures for Oracle Precompiler and Call Interface programs

Language	Member Name	Data Structure
C	SQLCAC	SQLCA
C	ORACAC	ORACA
C	SQLDAC	SQLDA
COBOL	SQLCACOB	SQLCA
COBOL	ORACACOB	ORACA
FORTRAN	ORACAFOR	ORACA
FORTRAN	SQLCAFOR	SQLCA
PL/I	ORACAPLI	ORACA
PL/I	SQLCAPLI	SQLCA

For more information about data structures, refer to the product-specific Oracle Programmer documentation.

ASID and Task Considerations

A connection to the OSDI-managed Oracle database service has affinity to the address space and OS/390 task (TCB) under which it originated. An OS/390 task can establish and maintain multiple connections to Oracle, but no connection can migrate to another OS/390 task or address space. If Oracle activity is to move from one task to another, then these steps must take place:

1. The first task must terminate all Oracle connections. This can be accomplished with an EXEC SQL COMMIT or EXEC SQL ROLLBACK using the RELEASE option.
2. The second task must logon to Oracle. This is considered a new session for Oracle and SMF accounting purposes.

3. The second task must reparse all the SQL statements and reestablish the connection environment of the first task.

When an Oracle connection is initiated from TSO, the interface routine attaches a subtask to process break signals when the user presses the [Attn] key. This is necessary because TSO does not allow a task to field a break signal while it is running in Oracle in cross memory mode. The subtask is responsible for setting and maintaining a functional STAX (attention exit) environment for the connection.

Program Status Word Protect Key

Connections to Oracle can be established in any supported OS/390 protect key, in addition to the normal problem state key of 8. Oracle notes the protect key from the program status word (PSW) at the time of the initial connection. All subsequent calls to Oracle by that connection must be performed using the same protect key.

The application-side virtual storage into which Oracle moves the results of database operations must be modifiable under that protect key. Normally this application-side virtual storage is acquired by the ORADRV interface routines. When these routines are invoked in problem state, they acquire storage in the default subpool zero. When these routines are invoked in supervisor state, they request storage in subpool 250, which assigns storage from subpool zero with the task protect key (TCBPFK field of the TCB).

For system integrity reasons, Oracle Corporation strongly discourages the use of PSW key 0 for Oracle connections.

PSW State

An Oracle connection can be established in either problem state or supervisor state. However, the same state needs to be maintained for all subsequent Oracle calls. Oracle interface routines always return control in the same state and key in which they were invoked.

Multiple Executions under a TCB

If an Oracle application or tool is invoked multiple times under the same TCB, then unpredictable results might occur.

Cross Memory Mode

Calls made to Oracle while in cross memory mode are not supported. Cross memory mode is defined as:

- Primary ASID does not equal home ASID
- Secondary ASID does not equal home ASID
- Secondary-addressing bit (PSW S-bit) is on

APF Authorization

A task connecting to Oracle can be APF-authorized. The connection is the same as non-authorized connections except for differences in the behavior of certain OS/390 facilities, such as security products.

16

Oracle9*i* Performance

The performance of your Oracle server depends on a number of factors. This chapter describes some of those factors and provides recommendations for improving and tuning your system to promote better Oracle performance.

The following topics are included:

CPU Usage

Under the OSDI environment, an Oracle9*i* for OS/390 database server can be configured to use multiple address spaces. This greatly expands the amount of virtual memory available to support higher workload levels, especially in the form of additional users. The first address space that is created is known as the AS1 or Control Address Space. The additional address spaces are Auxiliary Address Spaces. After the Oracle instance startup is completed, the user sessions are evenly distributed among the configured address spaces, or regions, based on available virtual memory.

Oracle Server Regions

The Oracle regions will consume CPU resources as users communicate with the database service to establish or terminate a session and to perform other special activities belonging to the Oracle regions, such as such as opening and closing files, database file I/O, and network connections. Additionally, the Control Address Space owns the Oracle background tasks, which consume CPU resources when they are active.

Background tasks in the Control Address Space use small amounts of CPU resources to perform normal background processing, such as:

- writing out to disk database and log file blocks from the buffer areas
- cleaning up after abnormal termination of user connections to the instance
- performing recovery operations
- synchronizing two-phase commit operations
- keeping critical data for instance recovery in the control files
- updating database file headers during checkpoints
- archiving closed redo log files

Typically, the Oracle regions consume very moderate CPU resources compared with those of the client regions. In a special case, however, the Oracle regions can use significant amounts of CPU resources. This situation occurs when a client uses the Oracle parallel execution feature. When this feature is used, the portion of database work that is parallelized (executed in parallel) will be performed by parallel execution slaves in the form of special subtasks within the Oracle regions, whereas the portion of work that is not parallelized (not executed in parallel) will be performed by the client task, as normally occurs. For details on how and where these parallel slaves are created, please refer to "Oracle Parallel Execution" on page 16-31.

Client Regions

When executing database operations in normal mode (in other words, without parallel execution), most resources are charged to the client address space, whether it is a TSO, batch, CICS, IMS, or Oracle Net address space. The first four examples represent clients executing locally in the OS/390 image. The last example is the case of a remote user accessing the Oracle9*i* for OS/390 server from another node in a network by using Oracle Net services.

Under OSDI, the Oracle Net address space performs both the protocol related networking function and the actual, non-parallel-executed database work on behalf of the remote user.

Oracle Net Client Processing

Network clients are dispatched within the Net region as preemptable SRBs. An SRB is a lightweight OS/390 facility to dispatch processes. When used in conjunction with a new mechanism for prioritizing work, this scheme provides a very efficient and manageable means of execution.

Further, a preemptable SRB can be associated with an enclave, an artifice that assigns priority to a unit of work independent of any specific OS/390 address space or process. Multiple processes (both tasks and preemptable SRBs) can be executed on behalf of an enclave and are granted resources according to installation-defined goals for the associated work.

An OS/390 system component called Workload Manager (WLM) monitors work that is executed under this scheme, and dynamically adjusts task and preemptable SRB dispatching priorities and other factors to ensure that defined goals are met. Refer to "Dispatching Priority" on page 16-22 for a detailed description of how to handle Oracle dispatching priorities for both local and remote clients.

A Net service under OSDI creates enclave SRBs for dispatching all network client requests. The CPU time associated with their database work is therefore attributed to the Net address space. Using WLM in goal mode, the Net region can run at a high dispatching priority to ensure good performance for the networking service, whereas the dispatching priority of the remote client database requests can be managed separately according to their own needs.

Memory Requirements

Above and Below the 16M Line

The Oracle server allocates nearly all storage areas above the 16M line. Only one small data area and a few load modules are allocated below the line. All other data areas, including the SGA, PGA, context areas, sort work areas, working storage, and most Oracle load modules are located above the 16M line. The sizes of these data areas depend on the settings of various INITORA and OSDI parameters and on the user workload.

IEFUSI Exit

An OS/390 exit called IEFUSI might be installed on your system. The IEFUSI exit prevents started tasks or batch jobs from getting the maximum region size when REGION=0M is specified. If an IEFUSI exit is implemented, it is specified in the SMFPRMxx member of SYS1.PARMLIB that is used during OS/390 initialization. To effectively run Oracle with an IEFUSI exit installed, ensure that the exit is coded to allow batch jobs or started tasks with the names of your Oracle regions to allocate a large amount of virtual memory above the 16M line.

Because Oracle allocates only the amount of memory it needs, you can safely allow Oracle to allocate any amount of memory up to the two gigabyte limit per address space that is imposed by 31-bit addressing conventions.

Reducing Real Storage Requirements

An excellent method for reducing the real storage requirements for Oracle is to place reentrant modules in OS/390 link pack areas. Most Oracle modules are link-edited with AMODE set to 31 and RMODE set to ANY. Place such modules in the extended pageable link pack area (EPLPA) above the 16M line. Some modules are linked with RMODE set to 24. Place them in the pageable link pack area (PLPA) below the 16M line. Current versions of OS/390 automatically load modules from the link pack area libraries into the appropriate link pack areas.

Keep in mind that under the current OS/390 implementation, placing modules into LPA reduces the private area of every address space in the system. In other words, there is a trade-off between code sharing and virtual memory availability.

Note: If you choose to take advantage of this OS/390 feature, be aware that any STEPLIB/JOBLIB definition in the JCL has precedence over the placed modules of any link pack areas. Make sure that such modules do not exist in the STEPLIB/JOBLIB libraries so that the link pack area copy will be used instead.

LPA Considerations for Database and Net Regions

Only the subsystem code module (ORASSI) is automatically shared, and it is shared by all Oracle subsystems and services.

The Oracle database and Net regions run different programs from the Oracle AUTHLOAD library. Each Oracle database address space has its own copy of ORARASC and a few other modules, and the Net address space has ORANET8 and several others modules as well. There is no sharing of this code, even between address spaces of the same database service. Due to an operating system restriction, you cannot put ORARASC into LPA -- doing so makes it impossible to run any other copy of ORARASC (for example, an ORARASC at a different maintenance level), whether from LPA or not. ORARASC is also quite small in size, so it is not necessary for it to be shared. The Net modules, such as ORANET8, are also quite small in size. In addition, because a single Net service can be used to access multiple database services, typically only one Net service will be deployed. As a result, sharing of Net code is also typically unnecessary.

If you are running multiple Oracle database regions (from either the same or different instances), an excellent candidate for LPA usage is the Oracle kernel (in other words, the ORACLE module), because it is quite large in size. As described above, prior consideration should be given to the impact on any non Oracle workloads that may be constrained by virtual storage.

LPA Considerations for Local Oracle Users

If your installation will run multiple concurrent local users, you can place the following modules from the Oracle CMDLOAD into the link pack:

- LIBCLNTS

  LIBCLNTS contains the interface routines for all Oracle client accesses. This includes tools, utilities, precompilers, precompiler applications and Access Managers. LIBCLNTS is an excellent candidate because of it's relatively large size. Place it in the EPLPA above the 16M line.

- SQLPLUS

  SQLPLUS is the primary Oracle batch and interactive SQL processor. It is used for user database queries, updates, table creates and drops, and so forth. In many systems, this module is used heavily enough to warrant link pack area placement. Place it in the ELPA.

Do not place other load modules from the Oracle libraries into the link pack area, because they are referenced infrequently, link edited RMODE 24, or not reentrant.

Oracle Server Storage Requirements

The Oracle server makes static-fixed, static-variable, and dynamic virtual memory allocations as the Oracle regions are started up and begin providing database services to users. Static-fixed memory allocations are storage areas that are always allocated in the regions including space for the Oracle load modules, working storage, and OS/390 data areas. Static-variable memory -- mainly the SGA -- differ from one warm start of the server to the next, depending on initialization parameter values. Dynamic memory allocations occur as users connect to the instance and access information that is stored in the server. The primary factors determining the number of concurrent Oracle users that can be supported under OS/390 are the user memory allocation requirements (depending on the application design), the INITORA and OSDI parameter values, the amount of virtual memory that Oracle regions are allowed to allocate, and the amount of central storage that is available for use by the Oracle regions.

Database Server Address Space Configuration

Exhausting virtual memory in an address space will lead to any of a number of types of failures, because it is impossible to predict which system activity requests for memory are going to be denied.

This scenario is best avoided by configuring a database instance with enough address spaces to contain the largest expected workload in terms of memory required. Doing this requires an understanding of the workload as well as of the database address space topography on OS/390.

In addition to carefully configuring server address spaces, you can use certain database region parameters that provide controls designed to reduce the likelihood of exhausting address space memory. These are discussed in "Limiting Sessions in a Server Address Space" on page 16-11, and "Limiting Memory Allocations in a Server Address Space" on page 16-11.

Determining the Number of Oracle Address Spaces

Each database address space starts out with a given amount of private virtual memory: 2048 megabytes less the memory that is used or reserved by OS/390 for shared access by all address spaces: SQA, CSA, LPA, and the OS/390 nucleus and related data. The sizes of these spaces, and thus the amount of private memory remaining in each address space, varies from one OS/390 system to another. You may need to consult with your systems staff to determine the available private area size on your system.

Once you know the private area size of your system, you must subtract from it the amount of memory that will be allocated in each address space for purposes other than Oracle sessions. This memory allocation includes the Oracle SGA, the Oracle kernel -- usually named ORACLE -- and other minor load modules (ORARASC, ORARSSRB, ORADIE), and the load modules and data structures of the OS/390 Oracle infrastructure, including the IBM Language Environment (LE) interface. The size of the Oracle SGA is determined primarily by parameters that you specify in the INITORA file and is displayed during Oracle startup. (For additional discussion of the SGA, refer to section "The Oracle SGA on OS/390" on page 16-9.) The size of the Oracle kernel and other modules can be determined using ISPF browse on the load library that contains it.

After the foregoing are subtracted, the remaining private memory in each server address space is available for Oracle sessions. The maximum amount of memory that is required by a given session depends mainly on the behavior of the application: the number of cursors opened, the specific SQL statements used, PL/SQL and/or Java requirements (if any), and so forth. In addition, there are several INITORA parameters (SORT_AREA_SIZE, HASH_AREA_SIZE, and so forth) and a database region parameter (INIT_STACK_SIZE) that affect the memory resources that are allocated during each session execution. This can be quite difficult to estimate in advance of running the application. The most reliable way to determine memory requirements is to review the Oracle session SMF records (which contain a session memory high-water mark) and analyze them to determine the average peak session memory. For more information on Oracle SMF records refer to "Interpreting an Oracle SMF Record" on page 9-4.

Once you know the average session memory requirement you can calculate the number of sessions that will fit in one address space as:

N = P / S

where:

N	is the desired result
P	is the available private memory per address space
S	is the average peak session memory

If we let **T** be the total number of concurrent Oracle sessions to be supported then you need (**T / N**) server address spaces. This number should be rounded to the next higher whole number, and to allow for reasonable variability in workload level, it may be advisable to add one more address space. In doing so, keep in mind that an SQA cost is associated with starting additional address spaces, which is discussed in the section "The Oracle SGA on OS/390" on page 16-9.

Oracle Corporation recommends that you specify the number of address spaces calculated here as the INIT_ADR_SPACES parameter (so the address spaces all start when the service is started). A somewhat higher number can be specified as MAXAS (maximum address spaces) on the DEFINE SERVICE command. This makes it possible to start additional address spaces dynamically if the initial estimate proves to be low. There is no cost for having additional address spaces in the MAXAS parameter until those address spaces are actually started. Note that MAXAS **must** be equal to or greater than INIT_ADR_SPACES. Care should be taken to specify a high enough value to accommodate unpredictable workload growth or spikes.

Note that using more than one address space results in the Oracle server becoming a "cross-memory address space" in OS/390 terms. These address spaces are not available for reuse when the Oracle server terminates. The OS/390 PARMLIB parameter RSVNONR specifies the number of address space numbers to reserve for use as they become unavailable. If you use multiple address spaces for the Oracle server, then you should increase the value specified for this parameter. Specifying too small a value, or letting RSVNONR take the default value could result in an unscheduled IPL if the number of address space IDs becomes exhausted. For example, you could stop the Oracle server for maintenance and then become unable to restart without an IPL. For more information, refer to the *OS/390 Initialization and Tuning Reference*, *SG28-1752*.

The Oracle SGA on OS/390

An Oracle server instance has a single System Global Area (SGA) regardless of the number of address spaces or regions configured. The Oracle SGA is shared across all of regions of a server using an OS/390 service called IARVSERV that allows one address space to "view" a range of private virtual memory that belongs to another address space. The Oracle SGA belongs to the first server address space (primary region) and is viewed (shared) by any other regions that are configured for that server. The virtual address range of the Oracle SGA must be reserved in each of the auxiliary regions to support the viewing mechanism. This is why the Oracle SGA size is subtracted from the private area size in every server address space of a given instance (not just the primary region) in the memory calculations of the previous section, "Determining the Number of Oracle Address Spaces".

Sizing the Oracle SGA and, most significantly, sizing the database buffer cache and the shared pool, are important instance tuning activities. The numerous INITORA parameters that do this and the general considerations for specifying their appropriate values are covered in the *Oracle9i Database Reference* and the *Oracle9i Database Performance Book Set*. A few guidelines for configuring some critical INITORA performance parameters are also presented in section "INITORA Parameters" on page 16-13. The following paragraphs describe some OS/390-specific issues to be aware of when tuning the Oracle SGA.

Because the SGA is not permanently pagefixed on OS/390 as it is on some other systems, there is little benefit in reserving SGA expansion space with the SGA_MAX_SIZE parameter. When you specify SGA_MAX_SIZE, the indicated maximum size is reserved (in virtual memory) in all server address spaces even if it is not all used.

Keep in mind that the Oracle SGA is mapped in all of the server address spaces of a given instance as discussed above. This means that increasing a server’s SGA size reduces the virtual memory available for Oracle sessions in every server region for that instance. If you do this, you may need to increase the number of regions in order to support your peak workload. The relationship is **not** linear. A 25% increase in SGA size may require more than a 25% increase in the number of server regions. When you make a significant change in the Oracle SGA size, repeat the calculations described in the previous section to determine the number of server address spaces that you need.

Another factor in Oracle SGA sizing is the overhead of the IARVSERV memory sharing mechanism. Currently, OS/390 must reserve 32 bytes of ESQA (Extended System Queue Area) for each "view" of each 4K page of memory shared. SQA is an expensive resource because it is page-fixed (always backed by real memory) and because it is globally addressable, using up an address range that would otherwise be part of the private area of each address space. Exhausting OS/390 SQA is a situation best avoided, so you should calculate the SQA overhead for your Oracle SGA and discuss this with your OS/390 systems staff before attempting to start the server.

Note: The total amount of SQA to reserve for all uses is an OS/390 system initialization parameter and cannot be changed without an IPL.

As an example, an Oracle server configured to run in 10 address spaces with a 512 megabyte SGA requires

32 x 10 x ((512 x 1024 x 1024) / 4096) bytes

or 40 Megabytes, of SQA, a significant amount.

Note: The IARVSERV SQA overhead occurs only when running Oracle servers in two or more address spaces. When a server is configured to run in a single address space only, IARVSERV is not used, and no SQA requirement is imposed. The current IARVSERV implementation provides page-level (4K unit) sharing granularity with a rather high cost in real memory overhead (on the order of 3% of all aggregated virtual views) for mapping tables.

The User Stack Area in OS/390

Each Oracle server user requires some extent of private memory to be used as a save area during normal execution. This area is known as the user stack.

When a user session is initiated, the connection is routed to a particular Oracle region. This region will then acquire a stack area based on the INIT_STACK_SIZE parameter. If the user requires more stack, additional extents are dynamically allocated and freed when they are no longer required. The actual stack requirement is dependent on the type of database call being used (SQL, PL/SQL, Java, OCI) and its complexity.

To fine tune this parameter, you might want to use the Oracle session SMF records analysis method described in the "Database Server Address Space Configuration" section (that was presented earlier in this chapter), or you can run your workload by varying the INIT_STACK_SIZE settings and then comparing the CPU usage of the various tests. The lower the INIT_STACK_SIZE value, the higher the potential CPU overhead that is caused by dynamic stack expansion.

As a starting point, you can use the Oracle Corporation recommended minimum value of 128K.

Limiting Sessions in a Server Address Space

The MAX_SESSIONS parameter also plays a role in managing virtual memory use. This is a hard limit on the number of sessions that can be active in one server address space, and it defaults to 1024 sessions. If a new bind (client connection) is routed to an address space that is at the MAX_SESSIONS limit, the server waits until some existing session unbinds (disconnects) before accepting the new session.

The idea behind MAX_SESSIONS is to keep the address space from accepting so many sessions that virtual memory is exhausted and unpredictable failures occur. The assumption is that it is better not to let an application connect and get started than to let it connect and incur transaction failure partway through its processing. A good value to use for MAX_SESSIONS is the value N (maximum sessions per address space) that was calculated in the section "Database Server Address Space Configuration" on page 16-6.

Both INITORA and database region parameter values must be set high enough to allow the required number of users to connect to the server. Please refer to the discussion about the INITORA PROCESSES parameter in section "PROCESSES" on page 16-16.

Limiting Memory Allocations in a Server Address Space

Two more database region parameters provide additional control over memory consumption in a server address space. The MAX_SESSION_MEM parameter allows you to impose a limit on the total virtual memory allocated to any single session in the Oracle server. This applies to all session-private memory requests made by the server, including the C stack and "heap" areas.

The limit is imposed on all sessions, including background processes and even parallel query slaves. If a session requests memory that would take it over the limit, the session receives an error (usually an ORA-04030) and the current transaction is rolled back.

Care should be taken not to choose too small a session limit. STARTUP processing in the current Oracle release requires about 10 megabytes of session memory. Note that session memory usage is reported in the Oracle SMF record, which can be used to help determine an appropriate limit amount.

Another parameter, REGION_MEM_RESERVE, allows you to limit the total memory allocated to all sessions and the Oracle SGA in an address space before exhausting address space private area. The "reserve" amount you specify remains available for internal implementation and OS/390 system function use.

Specifying an adequate reserve amount prevents the situation of exhausting address space memory and significantly reduces the impact of memory consumption problems. Requests for memory that would exceed the aggregate limit are rejected, resulting in an error and transaction rollback in the affected session.

Real Storage: Working Set

The amount of real storage ("working set size") that is required by the Oracle regions is very workload dependent, varying significantly with transaction complexity and rate, user concurrency, and locality of program and data reference. A lightly-loaded instance might require only 50K of working set for every megabyte of virtual memory that is allocated, while an instance that is supporting a much higher workload might require 750K of working set for every megabyte of virtual memory that is allocated. In general, Oracle load modules and the SGA remain in central storage while the context areas, sort work areas, and other individual session-related areas are more likely to be paged out to expanded or auxiliary storage when they are not heavily used.

Virtual Memory Allocation

The amount of virtual memory that is consumed by the server regions may vary significantly at run time because memory usage levels are very dynamic and fluctuate according to user workload. This is especially true when users connect and/or disconnect frequently or when users execute applications that open and close a large number of cursors, and when sorts are performed. Region size limitations can therefore become important, even with a small number of users connected to the instance. A region size limit that is too small prevents users from connecting to the server or from accessing information. Oracle Corporation recommends that you allow the server to allocate as much virtual memory as required and that you avoid imposing any region size limitations on the software. Other OS/390 facilities can be used to control the amount of central and expanded storage that is used by the server. The easiest way to allow Oracle9*i* to use the maximum amount of virtual storage is to specify the REGION=0M keyword parameter on the EXEC statement in the region startup JCL. Refer to "IEFUSI Exit" on page 16-4.

Oracle Tuning

This section deals with methods of optimizing your Oracle performance under OS/390. Before tuning your Oracle server specifically for running on OS/390, you should tune it for optimal performance independent of the operating system. For information on tuning methods that are independent of the operating system, refer to the *Oracle9i Database Performance Book Set*.

INITORA Parameters

Certain INITORA parameters can be used to optimize Oracle performance. Some of these parameters are discussed here, including:

- CURSOR_SHARING
- DB_CACHE_SIZE, DB_nK_CACHE_SIZE
- FAST_START_IO_TARGET
- FAST_START_MTTR_TARGET
- JAVA_POOL_SIZE
- LOG_CHECKPOINT_INTERVAL
- PROCESSES
- SESSION_CACHED_CURSORS
- SESSIONS
- SHARED_POOL_SIZE
- SORT_AREA_RETAINED_SIZE
- SORT_AREA_SIZE
- TRANSACTIONS

CURSOR_SHARING

Oracle compares SQL statements and PL/SQL blocks that are issued directly by users and applications, as well as recursive SQL statements that are issued internally by Oracle. If two identical statements are issued, then the SQL or

PL/SQL area that is used to process the first occurrence of the statement is shared. This means that it is used for the subsequent executions of all identical statements (no matter which users they belong to), thus reducing memory usage and improving performance.

To promote or maximize statement reusability and shareability, different execution arguments can be passed at run time by using bind variables. But even with bind variables, two SQL statements that are doing exactly the same job may not be considered identical. In addition to being identical, character-by-character in the SQL text, the bind variables in the SQL statements must match in name and data type. For example, the following two statements cannot use the same shared SQL area:

```
SELECT * FROM emp WHERE deptno = :department_no;
SELECT * FROM emp WHERE deptno = :d_no;
```

Generally, therefore, only truly identical statements can be shared. However, non-identical statements that are considered "similar" can also share SQL areas when the CURSOR_SHARING parameter is set to FORCE, frequently solving shared pool memory allocation problems and minimizing statement parsing overhead. Two separate occurrences of a SQL statement or a PL/SQL block are considered "similar" and can use a shared SQL area if they differ only in the literals. For example, the following two statements are considered similar:

```
INSERT INTO T VALUES(1, 'foo', 4)
INSERT INTO T VALUES(2, 'bar', 7)
```

The CURSOR_SHARING parameter may solve some performance problems. It has the following values: FORCE, SIMILAR, and EXACT (default). Setting CURSOR_SHARING to FORCE or SIMILAR will force similar statements to share SQL by replacing literals with system-generated bind variables. Replacing literals with bind variables improves cursor sharing with reduced memory usage, faster parses, and reduced latch contention.

Note: Shared SQL may be less appropriate for data warehousing applications. Also, setting CURSOR_SHARING to FORCE or SIMILAR may affect the execution plans of the statements. Refer to the *Oracle9i Database Performance Book Set* for further details.

DB_CACHE_SIZE, DB_nK_CACHE_SIZE

The size of the buffer cache is one of the most significant tuning adjustments controlling Oracle performance. Although these tuning parameters are very dependant on the application, the following two guidelines are for adjusting buffer cache size:

- Increase the cache size until 85 to 95 percent of your applications' requests for Oracle data are satisfied from the buffer cache. Note that for some applications that have large, randomly retrieved table data, you may not be able to get a high buffer hit ratio. Also, some smaller tables may be assigned a separate buffer pool that guarantees a 100% hit ratio. Refer to the use of multiple buffer pools in the *Oracle9i Database Performance Book Set*.
- Do not allow the instance to use so much memory that you impact the performance of everything running on the system.

The first guideline is easy to follow. The *Oracle9i Database Performance Book Set* describes a method for estimating the optimal number of buffers for an instance. Alternatively, you can simply increase the DB_CACHE_SIZE parameter value until you get the preferred hit ratio while your user applications are running.

The second guideline establishes limits for the size of the buffer cache. As Oracle memory usage increases, paging and swapping rates might increase as well, until OS/390 is so busy paging that it cannot complete any useful work. In general, maximum throughput first increases, then decreases, as the buffer cache size is increased. Throughput increases as the cache hit ratio improves, but decreases as Oracle uses too much memory and as the system paging rate goes up. These two trends balance at a point of optimal performance that is different for every Oracle instance and for every OS/390 installation.

Because the buffer cache is an architectural feature for avoiding disk I/O operations, you can, to some extent, trade off cache size for I/O rate. An instance with a small cache and a high performance I/O environment might perform as well as an instance with a large cache and a poorly performing I/O environment. Disk environments supported by storage processors with ample cache and high DASD cache hit rates can provide superior I/O performance that can alleviate some of the performance penalty of low buffer cache hit rates. On OS/390 systems with a lot of I/O capacity but with little central and expanded storage available, concentrate on obtaining optimal Oracle I/O performance by adjusting the layout of the database files to minimize device and path contention, and to maximize DASD cache hit rates. On systems with adequate memory, you can increase the size of the buffer cache to achieve an 85 to 95 percent hit ratio.

FAST_START_IO_TARGET FAST_START_MTTR_TARGET LOG_CHECKPOINT_ INTERVAL

Be aware that Oracle9*i* performs both full and incremental checkpoints, and therefore, care should be exercised to properly size ALL log files. The size of the smallest redo log is one of the key factors in determining the incremental checkpoint frequency. Ensure, therefore, that all log files are evenly sized so that checkpoint overhead can be effectively controlled. Also, fewer (but larger) log files are preferred to many (but smaller) log files.

To minimize checkpoint activity, set LOG_CHECKPOINT_INTERVAL to a large value, such as 5,000,000. This value should not be lower than the number of 4K blocks in your largest log file. Also set both FAST_START_IO_TARGET and FAST_START_MTTR_TARGET to zero to reduce incremental checkpoints to a minimum. This way, checkpoint overhead is practically limited to those times when LGWR switches from one log file to the next. This might, however, increase database recovery time by requiring more data to be processed upon recovery. For more information about minimizing checkpoint activity, refer to the *Oracle9i Database Performance Book Set*.

Note: In Oracle9*i*, the LOG_CHECKPOINT_INTERVAL and FAST_START_IO_TARGET parameters have been deprecated in favor of the FAST_START_MTTR_TARGET parameter.

JAVA_POOL_SIZE

This parameter defaults to a value of 25,165,824 bytes, or 24M. If you do not use Java at the Oracle kernel level, then you can reduce your SGA size considerably by specifying the minimum value, 0.

PROCESSES

The PROCESSES parameter needs to be set as accurately as possible to get the best performance from your instance. Many internal operational values that control Oracle performance are derived from the PROCESSES value. You can monitor the actual use of this and other Oracle resources to fine-adjust the SGA size and to optimize its performance by querying the V$RESOURCE_LIMIT view.

In general, if [the database region parameter MAX_SESSIONS] multiplied by [the number of Oracle address spaces started] is greater than [the PROCESSES parameter value], then you will never hit MAX_SESSIONS, because the instance specified PROCESSES limit will be reached first. When this occurs, any attempt to create a new session will immediately receive an Oracle kernel error message indicating that there is no more room for sessions. It will then be the user's responsibility to reschedule those requests for later. On the other hand, if PROCESSES is greater than the OSDI controlled limit (the number of address spaces times MAX_SESSIONS), an Oracle region might hit MAX_SESSIONS. If this happens, any additional session request will be put on hold without getting an error. In this case, the server waits until some existing session unbinds (disconnects) and then automatically accepts the new session.

From the perspective of virtual memory allocation, all of the user infrastructure at the OSDI level, such as LE context, is created for each session when it is first used, and the session ID translation table is allocated dynamically as needed. However, the INITORA PROCESSES, SESSIONS, and TRANSACTIONS parameters do cause resources to be statically allocated in the SGA, and the PMON background process scans the whole list many times during each housekeeping interval. The best strategy is therefore to keep INITORA PROCESSES small and to set MAX_SESSIONS appropriately, depending on the desired effect.

SESSION_CACHED_CURSORS

SQL and PL/SQL are extremely complex and powerful languages for database processing, requiring a statement to prepare for execution. This is known as "cursor parsing". To help an application run optimally, it is necessary to analyze how parsing works.

Two kinds of parse calls exist, hard and soft. A "hard parse" occurs when the SQL or PL/SQL statement is not found in the shared SQL area (shared pool), so a complete parsing is required (for example, retrieving object descriptions from the data dictionary, checking the user's privileges on those objects, calling the optimizer to generate the execution plan, and so forth). This is the most expensive kind of parsing, and should be minimized for repeated execution. This is automatically done, provided there is enough space in the shared pool to keep the parsed representation between executions. As a result of a hard parse, the private portion of a SQL statement (in the session's heap memory) as well as the shared portion (in the shared pool) are created. The private portion is linked to the shared portion. This is called "a statement cursor" or simply "a cursor". To execute any SQL statement in the shared pool, a cursor is always required.

The other type of parse call is known as a "soft parse", and is performed when the statement is already in the shared pool, but the session has lost the "link" to the shared portion, so that the private portion must be rebuilt and linked to its shared portion again. This happens when the cursor was closed for whatever reason. Of course, this soft parse is not as expensive as a full or hard parse, but still requires some processing. The user must be authenticated again to run the SQL statement, and all name translations must be done once more. All system statistics, including hard and soft parse call activity, are maintained by Oracle and can be displayed from the V$SYSSTAT view.

To eliminate soft parsing in COBOL, C, or other 3GL applications, the precompiler option HOLD_CURSOR=YES should be used. Other options, such as RELEASE_CURSOR and MAXOPENCURSORS, can be used in conjunction with this to achieve optimal results. Refer to the appropriate Precompiler Programmer's Guide for details.

For non-3GL programs (when you do not have the same degree of control over cursors) such as Oracle Forms, the cursors will automatically be closed when a new form is called. So if you subsequently return to the caller, at least a soft parse will be performed for each cursor. Other third-party tools may produce the same effect.

In this case, you should enable a special cursor caching mechanism that will keep a copy of the user's cursors even though they are closed. You can specify SESSION_CACHED_CURSORS=50 in the INITORA as a starting point and check the statistic '**session cursor cache hits**' from V$SYSSTAT to see your gains, and you can make further adjustment if necessary. For every hit that you get, you save one soft parse call. But be aware that this is done at the expense of increased memory allocation for every session in the SGA.

SESSIONS

This parameter does not generally need to be set, because a good default value is computed from the PROCESSES parameter value. Oracle Corporation recommends that you set PROCESSES accurately and let this parameter assume its default value unless auditing is enabled. Refer to the *Oracle9i Database Performance Book Set* for more information.

SHARED_POOL_SIZE

Shared pool requirements are dependent on a number of factors, primarily the number and complexity of SQL (and PL/SQL) statements executed by users who are logged on. But there is also a per-user per-cursor memory cost even for shared cursors (statements), so the requirements depend on the number of concurrent users as well. You will need approximately 250 bytes of memory in the shared pool, per concurrent user, for each open cursor that the user has, whether the cursor is shared or not. This is in addition to the actual SQL (and PL/SQL) context areas.

Having a shared pool that is configured with insufficient space might lead to poor or erratic response times and to increased CPU costs during application execution.

On the other hand, the shared pool should not normally be over-allocated. Maintaining an unnecessarily large code inventory will incur CPU overhead and will increase the SGA size. Shared pool tuning is a continuous and iterative process. Your requirements will vary as the SQL and PL/SQL workload evolves.

All the SGA-related structures, including the shared pool, can be displayed by querying the V$SGASTAT view. In particular, this view shows the amount of free memory in the shared pool, so it can be used to monitor the shared pool usage at regular intervals for tuning purposes. For a detailed discussion on the effect of the SHARED_POOL_SIZE parameter on your application performance, and how to size it properly, refer to the *Oracle9i Database Performance Book Set*.

SORT_AREA_SIZE SORT_AREA_RETAINED_SIZE

Sort operations are extremely common in a vast majority of applications and database activities, so a careful tuning of the SORT_AREA_SIZE parameter is very desirable. This value determines the maximum amount of memory that can be allocated by a single sort operation within an instance. If additional memory (over and above SORT_AREA_SIZE) is required, then the sorted rows are written to disk, in other words, temporary segments are created. Even though the sort areas are allocated out of the Oracle region private memory, they are assigned to the allocating session heap memory, and they are not shared among users. When a sort is completed, the memory retained by that process for another sort is reduced to the value specified by the SORT_AREA_RETAINED_SIZE parameter, both for in-memory and for on-disk sorts. The released (if any) portion is made available for other purposes within the same session, such as regular statement execution. Be aware that some extent of a sort area can be retained after sort completion to hold the sorted data as long as the cursor is kept open for subsequent result fetch. Finally, sort areas are not completely released and returned to the operating system for allocation by other users until the owner’s session is closed.

On a regular basis, for monitoring purposes as well as every time SORT_AREA_SIZE is adjusted, you should check the ′**sorts (disk)**′ and ′**sorts (memory)**′ statistics in the V$SYSSTAT view. The goal is to minimize the temporary table usage without incurring excessive memory contention. Please refer to "Index

Creation" on page 16-26 and "Sort Area Size" on page 16-33 for additional recommendations that apply to general users.

Note: For optimal results, it is recommended that you let the system automatically adjust the size of the session's working memory (including SORT_AREA_SIZE). Refer to the *Oracle9i Database Performance Book Set* for details on enabling this feature.

TRANSACTIONS

This parameter does not generally need to be set, because a good default value is computed from the PROCESSES parameter value. Oracle Corporation recommends that you set PROCESSES accurately and let this parameter assume its default value.

Minimizing I/O Bottlenecks

Access Methods

I/O operations are performed using an OS/390 facility called the Media Manager. This multiblock I/O facility utilizes the extended count key data channel command set, and is generally the fastest access method for performing disk I/O available in OS/390. Another advantage of the Media Manager is that it requires only a small amount of SQA memory.

Log Files

The Oracle logs are moderately active files, sustaining a variable I/O rate depending on the DML work being performed by the users of the database. In general, high levels of DML activity cause frequent I/O operations to the log file. These I/O operations are sequential writes. Other I/O activity on the same device can elongate the device response times by adding head seek time, which otherwise would not occur. Do not place your log files on devices with other actively accessed data sets unless your database is primarily accessed by retrieval operations (which do not cause I/O to the log files).

Relieving Log File I/O Bottlenecks

Log files are more likely to restrict Oracle performance if a single, poorly designed, DML intensive batch program is running in your system.

A simple application design change relieves the single-user log file I/O type of bottleneck. A batch program that is designed to commit numerous one-row DML operations at once will generate a lower I/O rate to the redo log file and can allow the database to sustain more DML operations per second than a batch program that commits one such operation at a time.

Archiving

When log archiving is enabled, configure Oracle to use a minimum of four log files. These log files need to be allocated on two different DASD volumes in flip-flop fashion, with files one and three allocated on one volume, and files two and four allocated on the other volume.

This arrangement avoids I/O contention between the Log Writer (LGWR) and the log archive batch job. With all log files on one volume, LGWR and the archive job both access the volume at the same time, potentially causing an I/O bottleneck. By allocating the files across two volumes, LGWR writes to one volume while the archive job reads from the other volume.

System Tablespace

The system tablespace can become an I/O bottleneck when all rollback segments and temporary tables are allocated in the system tablespace. This occurs if you place a significant update-intensive or disk-sort-intensive workload on your Oracle instance without a properly laid out database.

You can avoid this bottleneck by creating additional database files using new tablespaces to contain these heavily accessed objects. You also need to configure Oracle properly by recreating the rollback segments in the desired tablespaces, and you need to ALTER your users to change their temporary tablespace assignments from the system tablespace to your newly created temporary tablespaces. Configure these tablespaces to allocate space in large extents (using the DEFAULT STORAGE clause) to minimize space management overhead. For more information, refer to the sections on rollback segments, temporary tablespaces, and tuning sorts in the *Oracle9i Database Performance Book Set*.

OS/390 Tuning

The Workload Manager (WLM) is an OS/390 facility that allows installations to effectively manage their Oracle as well as non-Oracle workloads based on business priorities. Goals can be defined to reflect business priorities. The system manages the amount of resources, such as CPU and storage that are necessary for a given workload, to achieve its goal.

Like other WLM managed workloads, Oracle workloads should be assigned to appropriate service classes based on attributes such as subsystem name, service name, user name, and transaction name. Service class structure and importance are determined by the business needs of an installation. Workloads should also be classified into report classes to facilitate monitoring and validation of an installation's workload management policies.

Oracle Regions

Service classes allow you to control the priority of your Oracle instances relative to other workloads. Service classes should be defined for your Oracle instances based on the performance requirements of the instances. You usually do not need to define a service class for each Oracle instance, because multiple Oracle instances with similar performance requirements can typically be mapped to a given service class (production instances versus test and development instances, for example). Report classes provide more granular reporting capability for different Oracle instances within a given service class and should be used where necessary to monitor CPU, memory, and I/O resources that are consumed by individual Oracle instances in the reports that are generated by SMF/RMF and other measurement subsystems.

Dispatching Priority

The service class of the Oracle regions determines the relative dispatching priority of the background processes and other special tasks within the Oracle instances. The Oracle regions typically consume very moderate amounts of CPU resources. Normally, the bulk of the CPU resources that are consumed to process database requests are incurred by the client address spaces for local requests (and Oracle Net SRB enclaves in the case of remote requests) and should be managed accordingly (refer to "Local Clients" on page 16-24 and "Remote Clients" on page 16-26). Oracle, therefore, typically does not need to run at a high priority, but you may want to consider the special conditions that are associated with the parallel execution feature that is discussed below in the last paragraph of this section.

In general, the Oracle regions can be configured for lower dispatching priority (or lower importance) than high priority CICS and TSO workloads, and at about the same priority as high importance (or non-discretionary) batch workload. For example, CICS users for a given Oracle instance should be configured for higher priority (or importance) than the corresponding regions of the Oracle instance.

Similarly, in the case of TSO, higher priority (or importance) should be assigned to first or second period TSO workloads than to the Oracle regions.

If the Oracle dispatching priority is set too low, and if the system suffers from significant CPU contention (indicated by high processor delay in the Oracle regions), then some important Oracle internal requests might not get immediately processed, or the background tasks might not get dispatched often enough to perform the required work. For example, the buffer pool might become filled with modified buffers, and users might need to wait for Oracle to get dispatched and write out some database blocks to allow user processing to continue. The following scenario illustrates this situation:

You see the `'free buffer waits'` event (from a UTLBSTAT/UTLESTAT, STATSPACK, or similar report) showing a significant total value (the unit is hundredths of second) relative to the report interval during a DML-intensive period (update, delete, or insert operations). First, consider enlarging the buffer pool, or pools, to trade memory for I/O requests (you can have multiple buffer pools in Oracle9*i*). Assuming that no significant I/O bottlenecks are affecting the database files (fix them first if any occur), and if the AS1 or Control Address Space CPU delay is low, then it might be necessary to increase DB_WRITER_PROCESSES to schedule more parallel I/O. If the AS1 CPU delay is significant, however, you probably need to set the dispatching priority to a higher value first. Be aware that checkpoint activity also forces modified buffers to disk, adding to the I/O stress. You should also compare the `'physical writes'` and `'physical writes non checkpoint'` statistics from your report to make sure that the write activity is not being unnecessarily inflated by a poorly tuned checkpoint mechanism. Refer to the LOG_CHECKPOINT_INTERVAL discussion in section "INITORA Parameters" on page 16-13 and to the *Oracle9i Database Performance Book Set* for information on minimizing checkpoint overhead and for information on the UTLBSTAT/UTLESTAT and STATSPACK script utilities.

On the other hand, whenever users take advantage of the parallel execution feature that runs under special subtasks in the Oracle regions, the dispatching priority of the Oracle regions becomes an increasingly important tuning issue. In this case, Oracle dispatching priority determines how quickly these special requests are serviced and how much those Oracle users impact the overall throughput of the OS/390 system.

Memory Control

Because the Oracle regions run non-swappable, all workload controls that impact swapping or multiprogramming levels will have no effect on these regions.

Local Clients

TSO

The following considerations apply to resource intensive Oracle workloads within a TSO environment:

1. Increase the relative importance levels of TSO first and second periods. This supports transactions requiring greater resources and may result in a larger percentage of all transactions being completed in the first and second periods.
2. Consider adding a fourth or fifth performance period to account for extremely resource intensive TSO transactions.
3. Establish separate service classes for Oracle users. Use the service classes to reflect goals and relative importance of different TSO workloads that are classified by user attributes such as userid or accounting information.

CICS and IMS

CICS and IMS workloads can be managed using service classes and can be classified using attributes such as userid, transaction name, luname, and subsystem instance name.

Batch

Because batch workloads are typically discretionary in nature, Oracle batch jobs do not need to be separately classified. However, Oracle batch jobs can be distinguished from other batch jobs by establishing separate service classes, as described for the TSO environment.

When Oracle batch jobs are run under a certain service class, consider their priority relative to other Oracle and non-Oracle workloads. In a normal to heavily loaded system, if Oracle batch jobs run at a lower priority than others, the Oracle jobs might be swapped out for lengthy periods. If an Oracle job is swapped out while holding a critical latch, it may adversely impact the performance of other Oracle users.

Special Needs Functions

Special services classes should be considered for privileged users or special jobs such as those described below.

DBA Accounts

The database administrator (DBA) frequently needs priority access to the database in order to perform functions on behalf of all Oracle users. Granting higher relative importance to these types of work shortens the elapsed time for these functions to the benefit of all users. Except for database import and export, DBA functions generally do not require large amounts of Oracle and system resources compared to those of the user community.

Database Imports and Exports

Import and export functions are good candidates for higher relative importance when they involve the entire database. If the performance parameters of your system force swapping among long running batch jobs, you might want to consider non-swappable status for import and export.

Import and export performance can be optimized by maximizing the size of the buffer that is used to transfer rows to and from the export file. The buffer needs to be large enough to hold approximately 1000 table rows to get the best performance from these utilities. In addition, you can improve performance by increasing the number of buffers that are available for reading and writing the export file. Use the DCB BUFNO JCL parameter to increase the number of buffers. The I/O operations that are issued by QSAM and BSAM will not generate a channel program using more than 30 buffers or more than approximately 240 KB. For I/O bound processes such as Export and Import, you should specify a BUFNO that allocates approximately 480 KB of buffers. This value will give you the maximum amount of overlap between two maximum I/O channel programs. Refer to *Oracle9i Database Utilities* for more information.

Data Loading

The direct path in SQL*Loader is much more efficient than the conventional path. When using the DIRECT option, SQL*Loader is generally I/O bound on the input data file. To reduce the elapsed time that is required for a load operation, you need to increase the number of buffers that are available for reading the input file by adding a DCB BUFNO parameter to the input file allocation. Performance improvements occur as the number of buffers is increased to 200, although 48 buffers yield a significant improvement in the data load rate.

If you cannot use the DIRECT option, then specify the largest bind array size (using the ROWS parameter) that you can. An array size of approximately 1000 rows improves performance significantly over the default size of 64 rows.

Index Creation

The creation of indexes on large tables can consume significant resources. Consider higher relative importance and non-swappable status for these functions.

In addition to OS/390 tuning parameters, you need to consider special session settings to support index creation in large tables. Increasing the SORT_AREA_SIZE parameter value can substantially reduce the elapsed times of index creation jobs. This can be done selectively at the session level by using the ALTER SESSION SQL command so that other non-critical jobs will still use the INITORA specified value.

Sorting Data by Key Before Loading

You can use SYNCSORT, DFSORT, or another OS/390 sort utility to sort the data by key before loading it into Oracle. Once the data is sorted, load the data into Oracle and create the index with the NOSORT option. For large data loads, this technique can save significant amounts of time when loading data and creating the index.

Remote Clients

Remote clients that access an Oracle server through the Oracle Net service are dispatched on a lightweight unit of work called an enclave SRB within the Net address space. The performance characteristics of such work can be effectively managed when used with WLM in goal mode. Enclave transactions are managed separately from one another as well as from the Oracle Net address space they run in.

The Net startup option ENCLAVE (CALL|SESS) controls how the database request from the client is handled (described under "PARM" on page 10-4). With ENCLAVE(SESS) specified in the PARM value used at Net startup, classification of the work is done once when a new remote connection is made. Oracle Net presents WLM with attributes for workload classification. Some of the network specific attributes that can be used for classification include protocol, host name, or IP address. The list of WLM attributes available for classification is shown in Table 16–1, "Workload Manager Attributes and Values". The enclave will be deleted at session termination (logoff) time. Because the classification happens only once per session, only velocity goals are appropriate for the enclave’s service class.

If ENCLAVE(CALL) is specified in the PARM value used at Net startup, then the enclave is deleted when the request from the client is finished (when Net needs more data from the client). Deleting the enclave reports the transaction completion to WLM, providing response time and transaction counts to any workload monitors such as RMF. The next request arriving from the client will be classified into a new enclave. The values available for classification are the same as with ENCLAVE(SESS) above, and are shown in Table 16–1, "Workload Manager Attributes and Values". Because the classification is done for each network request, response time goals should be used for the enclave's service class.

If there is no WLM policy active, then all of the Oracle Net work will be dispatched in SRBs that will be executed using the dispatching priority of the Oracle Net address space. If you are running in WLM compatibility mode but have a WLM policy active, it is possible to manage the enclaves in a performance group (PGN) distinct from Oracle Net by adding the SRVCLASS= parameter to the IEAICSxx member of PARMLIB. However, since you must still build a WLM policy and activate it in order to get this capability, it is recommended that you run in goal mode.

Table 16–1 Workload Manager Attributes and Values

ATTRIBUTE	VALUE
Subsystem Type	C'OSDI'
SI	OSDI subsystem name, for example, WFM1
UI	User ID from connect
NET	first 8 characters of dotted IP address (example: 100.024.)
LU	last 7 characters of dotted IP address (example: 020.003)
CT	Protocol from connect 'TCP'
SPM	position 1-8: Oracle database service name
SPM	position 9-89: TCP/IP hostname (left justified)

Note: Leading zero characters must be used in the nodes of the dotted IP address.

"Subsystem Type" is not strictly an attribute. WLM has several predefined subsystem types (JES for example). You must define a new subsystem type of "OSDI" to WLM if you desire WLM monitoring of OSDI work. Please refer to the IBM manual, *OS/390 MVS Planning: Workload Management*, C28-1761, for information on how to do this, and for information on how to utilize the attributes listed above to manage work. Generally, WLM is configured using ISPF and the IWMARIN0 REXX exec.

If you choose to run in goal mode without a policy active, or run without a section in the policy for the OSDI subsystem, then the client work will be assigned the service class SYSOTHER, which has a discretionary goal. Performance is likely to be unsatisfactory. Running in compatibility mode will result in the enclave running in a preemptable SRB at the dispatching priority of Oracle Net. Note that this means the CPU time for the user client work will be charged to the Oracle Net address space as SRB time.

The following is an example of a WLM classification rules ISPF panel:

```
Subsystem-Type  Xref  Notes  Options  Help
--------------------------------------------------------------------------
              Modify Rules for the Subsystem Type          Row 1 to 4 of 4
Command ===> ____________________________________________  SCROLL ===> PAGE
Subsystem Type .  : OSDI       Fold qualifier names?   Y  (Y or N)
Description    .  .  . OSDI SubSystem Type

Action codes:  A=After    C=Copy        M=Move       I=Insert rule
                  B=Before   D=Delete row    R=Repeat   IS=Insert Sub-rule
                                                                More ===>
              -------Qualifier-------------            -------Class--------
Action    Type       Name     Start                    Service     Report
                                             DEFAULTS: ORACLES     ________
 ____  1  SI           ORAC    ___                     ________    ________
 ____  2  NET        010.100   ___                     ________    ________
 ____  3  LU         001.080   ___                     ORACLEM     ________
 ____  3  LU         001.081   ___                     ORACLEH     ________

**********************************BOTTOM OF DATA ******************************
```

This rule assigns the service class ORACLEM to all work arriving from a client at IP address 10.100.1.80, and assigns ORACLEH to all work from the client at IP address 10.100.1.81. Note that the service class ORACLES is assigned as the default service class to Net workloads that cannot be classified by the above rules. It is very important to specify a default service class. Without a default service class, an error in the classification rules could result in no rules matching. In this case, the request will be assigned service class SYSOTHER, which has a discretionary goal. This will result in undesirable performance characteristics.

If ENCLAVE(CALL) is specified in the PARM value at Net startup, you should specify response goals, or percentile response goals for the service classes used by Oracle enclaves.

The following shows a sample screen defining a service class with three periods. The first period has a response time goal of 15 ms. at importance 1. This gives short requests high priority access to the CPU. If the request takes more than 50 CPU service units, the enclave is migrated to a second period at importance 3. If the request is still running after 500 service units, it is then migrated to a third period at importance 5. This design of service class goals is only feasible if the ENCLAVE(CALL) parameter is used. It has the advantage of providing fast, high priority response to short requests, while treating longer requests at low, batch-like priorities.

```
 Service-Class  Xref  Notes  Options  Help
 ----------------------------------------------------------------------
Modify a Service Class             Row 1 to 2 of 2
  Command ===> _________________________________________________________

  Service Class Name . . . . . : ORACLEH
  Description  . . . . . . . . . Oracle Mid Tier #1
  Workload Name  . . . . . . . . ORACLE    (name or ?)
  Base Resource Group  . . . . . ________  (name or ?)

  Specify BASE GOAL information.  Action Codes: I=Insert new period,
  E=Edit period, D=Delete period.

        ---Period---  ---------------------Goal---------------------
 Action  #  Duration   Imp.  Description
   __
   __    1  50         1    Average response time of 00:00:00.015
   __    2  500        3    Average response time of 00:00:00.500
   __    3             5    Execution velocity of 10
 ****************************** Bottom of data *********************
```

PL/SQL and Java

Oracle supports two major programming languages in the database: PL/SQL and Java. A large portion of Oracle customers use both PL/SQL and Java to build database applications. By adding Java to the server, Oracle Corporation has opened up the range of things that can be done in the server. Note that computational operations were typically being done outside the server on clients using C/C++. Now, much of that logic can be moved into the server with the benefits of reduced network latency and round trips, improved performance, and portability.

Choosing JAVA and PL/SQL

How do you decide whether to use PL/SQL or Java? Because PL/SQL and Java are two fundamentally different languages, the relative performance of the two is difficult to compare. The results of such a comparison will depend on the specific conditions of customer applications. Two important ideas, from a performance point of view, are: PL/SQL and JAVA can coexist, and both languages have been built around different design points and are therefore better suited for different tasks.

PL/SQL Optimized for SQL Processing and Distributed Data

For SQL intensive applications, PL/SQL is generally faster than Java. This is especially true when applications execute tight loops around SQL, when a large amount of data needs to be converted from SQL types to Java types, or when PL/SQL can use bulk operations. Similarly, for applications (such as triggers) that are characterized more by the speed of entry into the PL/SQL or Java engines than by the execution speed of the application, PL/SQL is generally faster. PL/SQL was also optimized in Oracle8*i* to perform some string operations, such as concatenation. PL/SQL is significantly faster both transparently (from internal optimizations) and via new features (such as bulk SQL). Applications (or parts thereof) that are highly SQL intensive should be written in PL/SQL.

Java Optimized for Computation and Open Distributed Computing

Java is a general purpose object-oriented programming language with a rich type system, a component model, and other facilities that supports multi-tier distributed computing standards (CORBA and EJB). In contrast with PL/SQL, which shares the same type system as SQL, Java has a much more general purpose and richer type system that was designed to represent arbitrarily complex data structures such as multi-dimensional arrays, graphs, and so forth. Because it is an object-oriented language, Java provides elegant facilities to inherit from existing Java types and to build complex class hierarchies and arbitrarily deep nested hierarchies quite easily. Further, the Java VM is much better tuned to execute programs which have little or no SQL, and it can use native numeric machine data types rather than the more precise (but often slower) Oracle SQL data types that are used by PL/SQL.

For applications that involve complex object-oriented or highly CPU-intensive "number-crunching" operations, Java can be faster than PL/SQL. For example, Java has native support for floating point operations whereas PL/SQL uses SQL numbers. Furthermore, when natively compiled using the Java compiler (NCOMP), Java's performance can be improved significantly.

Summary

For computation intensive programs, pure Java execution (method calls, data manipulation, algorithmic operations, and so forth) can be faster than PL/SQL. On the other hand, PL/SQL is generally more optimized for SQL access. As a net result, the overall performance of the application will depend to some extent on the relative balance between computational operations and SQL access. Finally, if the run-time performance of your application is dominated by long running SQL statements, then the choice between PL/SQL and Java will not make a significant difference to the overall run time. Nevertheless, if you are primarily doing SQL access, PL/SQL is a simpler and more efficient choice.

Oracle Parallel Execution

The Oracle parallel execution feature enables multiple server tasks to process a single piece of work concurrently. Except for the Parallel SQL*Loader special case, these parallel execution slaves run as subtasks within the Oracle regions. The parallel execution feature enables all of the following:

- multiple jobs to load concurrently into the same table using SQL*Loader with DIRECT_PATH set to TRUE
- multiple tasks to perform create index (scan and sort by separate tasks)
- multiple tasks to perform a create table as select from another table
- multiple tasks to perform query (separate tasks can coordinate, scan, or sort)
- other parallelized operations such as parallel DML

CPU Utilization

The parallel execution feature can dramatically reduce elapsed time for a given workload, as long as the necessary CPU capacity is available.

Although the percent increase in CPU time on a SQL statement basis is typically small due to some overhead that is required to parallelize, the elapsed time during which the CPU is consumed is much shorter than for non-parallel operation. This means that the percent of the machine that is used is significantly greater for a shorter time interval. For example, a non-parallel query might use 10 percent of the CPU for a period of 60 seconds, while a parallel query for the same query might use 99 percent of the CPU for a period of approximately 6 seconds. Monitoring and managing overall CPU utilization is therefore an important part of the tuning effort.

Parallel Execution Slaves

Any Oracle region is capable of starting parallel slaves as needed, within the limits specified by certain INITORA parameters and depending on user requests. The number of slaves (if any) that are specified in PARALLEL_MIN_SERVERS is automatically created in the AS1 during normal instance startup. These AS1 slaves can do parallel work for any database user, regardless of the particular region to which the session in question is assigned. Additional slaves, up to the PARALLEL_MAX_SERVERS value, will be dynamically added as required. Even though these on-demand started slaves can be created anywhere, depending on the requesting region (not necessarily AS1), they all are part of the parallel slave pool and can provide service to users in any instance region upon completion of the work for which they were originally created.

The OSDI DISPLAY SESSION command can be used to display the parallel execution slaves that are started in an Oracle instance.

Example:

```
F MYORA8,DISPLAY SESSION JOBNAME(P*)
```

or

```
F MYORA8,D SESS JOB(P*)
```

The parallel execution slaves are identified by a `JOBNAME` in the `'P`*`nnn`*`'` form where *nnn* is a number in the 000 to 999 range.

File Layout Considerations

The parallel execution feature works best on files that are allocated across many disk drives. This allows the parallel execution slaves to maximize concurrent access to the data files and to minimize device contention. The temporary tablespace that is used for sorting (queries and index creation) should also be allocated across several disks. This tablespace should be defined so that its space is allocated in large extents to minimize space management overhead. Note that the above technique is sometimes referred to as "Striping" and should not be confused with the OS/390 feature known as Extended Facility Striped Data Sets.

Often, the best method for determining the optimal degree of parallelism is to:

1. select a degree of parallelism
2. determine the CPU time and the elapsed time

3. compare those times with the times for several higher and lower degrees of parallelism

For the initial estimate of degree of parallelism for a single user, use one or two times the number of system CPUs, depending on the available system capacity. This is a good estimate for sort-intensive operations because they tend to be CPU bound. If the operation is I/O intensive (that is, more scanning of the data than sorting), the number of disk drives involved in the scans is a good starting point.

Sort Area Size

If memory is abundant, setting SORT_AREA_SIZE to a large value can be beneficial.

For example:

```
SORT_AREA_SIZE=1024000
```

Using a sort area much larger than 1M, however, may not provide a significant benefit unless it is large enough to completely eliminate the need of a temporary table on disk. Also be aware that a parallel operation using a sort will acquire as many sort areas as the parallel degree that is specified.

If memory is a concern on the system (insufficient memory or high paging), you might want to decrease the SORT_AREA_SIZE.

Memory use increases with sorting (for example, with certain selects and create index). This increase can be substantial and is dependent upon the value of SORT_AREA_SIZE and the degree of parallelism. However, this increase is over a shorter period of time, typically resulting in lower main storage occupancy (working set X elapsed time) and lower average overall storage use by Oracle. Paging rate variability can increase and must be monitored.

Verifying Parallelism

The V$PQ_SESSTAT, V$PQ_SLAVE, and V$PQ_SYSSTAT views can be used to view statistics on parallel workloads. Set RELEASE_CURSOR=YES in precompiled programs for parallel queries. This prevents parallel server tasks from being retained by a cursor when the query is complete.

The resource trade-off with the parallel query option is CPU and memory versus elapsed time and throughput. The goal is maximum gain in elapsed time or throughput with minimum cost of CPU or memory.

Parallel Execution Recommendations

Ideal candidates for parallel execution are:

- multiple CPUs
- low average system CPU utilization
- sufficient memory to devote to sorting space
- well-striped data
- table scans of tables too large to be cached
- overnight batch-window work (for example, parallel SQL*Load or index build)
- any workload with a limited transaction volume but a high performance requirement

Candidates that might be unsuitable for parallel execution are:

- unstriped data with multiple concurrent access
- data on volumes with poor response time
- systems with high memory and CPU utilization

Applications Performance Diagnosis

Identifying and Tuning High Load SQL

Whether you are writing new SQL statements or tuning problematic statements in an existing application, your methodology for tuning database operations essentially concerns CPU and disk I/O resources.

- Step 1: Find the Statements that Consume the Most Resources
- Step 2: Tune These Statements to Use Fewer Resources

Step 1: Find the Statements that Consume the Most Resources

Focus your tuning efforts on statements where the benefit of tuning demonstrably exceeds the cost of tuning. Use tools such as TKPROF, the SQL trace facility, SQL Analyze, Oracle Trace, and the Enterprise Manager Tuning Pack to find the problem statements and stored procedures. In addition, you can query the V$SORT_USAGE view to see the session and SQL statement associated with a temporary segment.

The statements with the most potential to improve performance, if tuned, include:

- Those consuming greatest resource overall.
- Those consuming greatest resource per row.
- Those executed most frequently.

In the V$SQLAREA view, you can find those statements still in the cache that have done a great deal of disk I/O and buffer gets. (Buffer gets show approximately the amount of CPU resource used.)

Step 2: Tune These Statements to Use Fewer Resources

Remember that application design is fundamental to performance. No amount of SQL statement tuning can make up for inefficient application design. If you encounter SQL statement tuning problems, then perhaps you need to change the application design.

You can use two strategies to reduce the resources consumed by a particular statement:

- Get the statement to use fewer resources.
- Use the statement less frequently.

Statements may use more resources because they do the most work, or because they perform their work inefficiently—or they may do both. However, the lower the resource used per unit of work (per row processed), the more likely it is that you can significantly reduce resources used only by changing the application itself. That is, rather than changing the SQL, it may be more effective to have the application process fewer rows, or process the same rows less frequently.

These two approaches are not mutually exclusive. The former is clearly less expensive, because you should be able to accomplish it either without program change (by changing index structures) or by changing only the SQL statement itself rather than the surrounding logic.

For more information on identifying and tuning problematic SQL statements, refer to the *Oracle9i Database Performance Book Set*.

SQL TRACE Facility

The SQL trace facility provides performance information about individual SQL statement execution. It can provide event-based statistics (parses, executions, fetches, physical and logical reads, row counts, and so forth) as well as time-based statistics (CPU and elapsed times). When a trace data set is processed by the TKPROF utility to convert it to a readable format, the SQL statement execution plan is also reported. In addition to enabling the SQL trace facility, you should activate the timed statistics gathering to make the trace data more meaningful for tuning purposes.

One of the methods to enable the SQL trace facility (including time-based data) is to place the following statements in the INITORA file:

```
SQL_TRACE=TRUE
TIMED_STATISTICS=TRUE
```

When the SQL trace facility is enabled this way, it takes effect at the instance level so that a trace data set is generated each time that a user logs on to the instance or each time that a program is executed. Trace data sets can be generated in the form of spool files belonging to a particular Oracle region or can be written directly into regular disk data sets. After a session is finished, its corresponding trace data set is closed and made available for processing.

If you do not want to log trace data for every user, you can issue the following SQL commands in the session for which you want to gather data,

```
ALTER SESSION SET SQL_TRACE=TRUE;
ALTER SYSTEM SET TIMED_STATISTICS=TRUE;
```

and omit these parameters from the INITORA definition. Other methods are available to enable a SQL trace for a given session from another user's session. For more information about the SQL trace facility, refer to the *Oracle9i Database Performance Book Set*.

Note: You must have ALTER SYSTEM system privilege to enable the timed statistics dynamically. The ALTER SYSTEM command can be executed from any session, because these statistics are enabled or disabled at the instance level only. The setting stays in effect as long as the database is mounted, or until it is reset by a new command.

Preparing a File for TKPROF Processing

If the Oracle trace files destination is SYSOUT (see the TRACE_DSNAME database region parameter), the following considerations apply. Otherwise, the trace data sets can be processed directly by TKPROF.

A trace file in the spool queue needs to be prepared for the subsequent TKPROF processing by copying the trace data (using SDSF) into a regular sequential file without the ASA control characters present in spool files. This is required because TKPROF cannot read an output file directly from the spool, nor can it recognize ASA control characters.

The ASA control characters can be eliminated by "printing" the trace data to a pre-allocated DD name instead of to a dsname. In this case, SDSF copies the data as is and does not keep control characters.

Example:

From ISPF option 6 (or the TSO READY prompt), issue:

```
ALLOC F(trc) DA(trace.dataset) NEW CAT LRECL(136) RECFM(V,B) -
SPACE(2 2) TRACKS BLKSIZE(4096)
```

Then enter ISPF/SDSF and browse the trace file you want to copy. From the command line, issue:

```
 PT F trc
 PT
 PT CLOSE
 TSO FREE F(trc)
```

TKPROF

The TKPROF program translates the Oracle trace data set into readable form. To invoke the TKPROF program from ISPF option 6, use:

```
CALL 'oran.orav.CMDLOAD(TKPROF)' '/DSN/trace.dataset
/DSN/output.dataset [options]'
```

where:

trace.dataset	is the name of the trace data set that you are translating.
output.dataset	is the name of the file where the translated trace data is written.
options	are any optional TKPROF arguments that you can specify.

The *trace.dataset* and *output.dataset* are not subject to FNA processing. The full data set names must be specified using /DSN/ notation. For more information about the TKPROF utility, refer to the *Oracle9i Database Performance Book Set*.

Alternatively, you can invoke the TKPROF utility in batch mode, as follows:

```
//MYSTEP   EXEC PGM=TKPROF,
// PARM='/DD/IN /DD/OUT [options]'
//STEPLIB  DD DISP=SHR,DSN=oran.orav.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=oran.orav.MESG
//SYSUDUMP DD SYSOUT=*
//SYSOUT   DD SYSOUT=*,DCB=(LRECL=350,BLKSIZE=3500,RECFM=VB)
//SYSERR   DD SYSOUT=*,DCB=(LRECL=350,BLKSIZE=3500,RECFM=VB)
//ORA@sid  DD DUMMY
//ORAPRINT DD SYSOUT=*
//IN       DD DISP=SHR,DSN=trace.dataset
//OUT      DD SYSOUT=*,DCB=(LRECL=350,BLKSIZE=3500,RECFM=VB)
//SYSIN    DD DUMMY
```

Oracle Access Manager for CICS

This section discusses how performance of Oracle Access Manager for CICS is affected by the following:

- Thread definition parameters
- Storage requirements for base code, thread table, and connected thread
- Tool stack requirements

Thread Definition Parameters

Several Oracle Access Manager for CICS thread definition parameters can significantly impact the performance of certain transactions connecting to an Oracle database instance from CICS.

Impacted transactions include:

- User transactions

  User transactions occur whenever a CICS user presses a program function [PF] key or the [Enter] key on their terminal.

- CICS transactions

  CICS transactions occur whenever a CICS user initiates a transaction to CICS by typing a transaction identifier in the upper left corner of the terminal screen and pressing a [PF] key or [Enter]. For pseudo-conversational CICS transactions, each user transaction is a CICS transaction. A CICS transaction can have Oracle database transactions imbedded within it.

- Oracle database transactions
  - When ORACLE COMMIT processing is selected as the recovery choice, Oracle database transactions are sequences of one or more SQL statements committed to or rolled back from the database together.

    Oracle database transactions explicitly terminate when the user issues a COMMIT or ROLLBACK SQL statement, and implicitly terminate when the user disconnects from the instance. Oracle database transactions can be imbedded within a CICS transaction.

    Only use Oracle COMMIT when coordinated recovery between non-Oracle and Oracle database resources is not a requirement.
  - When CICS is selected as the recovery choice, Oracle database transactions are included in the logical unit of work for the CICS transaction.

Thread Sharing

Threads are assigned to CICS transactions for the duration of the transaction. Some types of Oracle database transactions can continue for a long time, preventing multiple users from using the same thread.

Multiple users of pseudo-conversational CICS transactions (normally programmed in Pro*COBOL or Pro*C) can effectively share threads because each user transaction corresponds to one CICS transaction and one Oracle database transaction. When the user transaction completes, the Oracle database transaction completes and the thread that was in use is released. Each pseudo-conversational transaction user spends more time outside transactions than actually running transactions (that is, think time is higher than response time). Therefore, each thread generally supports multiple users of this type of application program.

When a user enters a conversational CICS transaction, the CICS transaction is not terminated until the user leaves the environment. A CICS transaction of this type generally consists of multiple user inputs for multiple Oracle database transactions. Therefore, each conversational transaction obtains a thread for a longer duration. This must be taken into account when defining the number of threads.

Subtask Sharing

A CICS transaction runs under the same CICS subtask for the duration of the transaction. For pseudo-conversational Pro*COBOL or Pro*C CICS transactions, threads using CICS subtasking are efficient. Each CICS transaction gets assigned to a thread, executes under a CICS subtask, and then frees the thread when the transaction ends. However, conversational CICS transactions are assigned to the same thread until the user ends the transaction by exiting it. This implies that each conversational transaction monopolizes a CICS subtask until the transaction ends. In other words, each thread supports only 1 to 1.5 conversational transactions depending on how often the users exit the transaction.

Authorization

Some CPU resources are required to connect Oracle Access Manager for CICS threads to the Oracle database instance. For the best performance, minimize the number of connections by setting up your thread definitions to use the autologon facility. This facility is implemented with the AUTH thread definition parameter. Pre-authorized threads improve transaction performance.

Note: The autologon facility is not supported when connecting Oracle Access Manager for CICS threads to a remote Oracle database instance.

The most efficient way to define thread authorization is by using the AUTH option `authorization string`, protecting the threads from being torn down by setting PROTECT to YES and specifying only transaction codes in the TRANSAC parameter that can use the authorization string. In this situation, the threads are connected to the instance when the Oracle Access Manager for CICS adapter is started, and do not have to reconnect until the next startup.

The second most efficient thread authorization scheme is to use the `program` or `transid` AUTH options, protect the threads from tear-down, and define the threads only for the program or transaction code specified. This scheme causes the thread to connect to the instance when the first transaction is assigned to the thread. All subsequent transactions use the same connection because they have the same transaction code or program name.

The least efficient way to define threads is without an AUTH parameter so the transactions explicitly connect to the instance. Oracle Access Manager for CICS thread logon processing is more efficient than explicit SQL CONNECT statement processing, especially when the same connection can be used by multiple CICS transactions.

PROTECT

Threads defined with PROTECT set to YES remain connected as long as the Oracle Access Manager for CICS adapter is started. CPU resources are required to connect threads, so thread protection can reduce overall CPU requirements.

Thread protection can cause the Oracle Access Manager for CICS adapter to use more memory in the CICS region. Protected threads remain connected even when they are not in use by a CICS transaction, and connected threads require slightly more memory than disconnected threads.

In general, protect threads that are used frequently and that are automatically connected using the autologon facility. There is no performance advantage to protecting threads that are not pre-authorized because thread protection simply maintains the connection to the Oracle database instance.

Base Code Storage Requirements

Table 9-3 shows the base code storage requirements for Oracle Access Manager for CICS.

Table 16–2 Base Code Storage Requirements for Configuration

Usage	Below 16M	Above 16M
ORACICS	0 bytes	28K
CICADPX	24 bytes	0K
LIBCLNTS	0 bytes	24M*
* Note: LIBCLNTS is loaded in above-the-line storage, not managed by CICS.		

Adapter Storage Requirements

Table 9-4 shows the adapter storage requirements for Oracle Access Manager for CICS, which are the same for both the local and remote configurations. The total variable storage requirement per active Oracle server/CICS transaction is 30,748 bytes.

Table 16–3 Adapter Storage Requirements for Local and Remote Configurations

Type of Storage	Usage	Area	Below 16M	Above 16M
Fixed	Per active Access Manager for CICS adapter	Global Exit Area	1344 bytes	0 bytes
Variable	Per active Oracle database/CICS transaction	Local Exit Area	432 bytes	0 bytes

Thread Table Storage Requirements

For the thread table, a fixed amount of storage is used. This calculation is used to determine a thread table's fixed storage requirements (located above the 16M line):

```
8176 bytes + (1128 bytes x MAXTHRDS)
```

Use the same calculation for both the local and remote configurations.

Connected Thread Storage Requirements

Table 16–4 shows the connected thread storage requirements for Oracle Access Manager for CICS.

All buffer storage for connected threads under CICS release 4.1 or higher is obtained above the 16M line.

Table 16–4 Connected Thread Storage Requirements

Type of Storage	Usage	Area	Below 16M	Above 16M
Variable	Per active Oracle database/CICS transaction	Oracle client storage	0 bytes	100K bytes*

*Note: This is above-the-line storage, not managed by CICS.

Oracle Access Manager for IMS

This section describes the storage requirements for Oracle Access Manager for IMS TM.

Access Manager for IMS TM Base Code Storage Requirements

Table 16–5 Access Manager for IMS TM Base Code Storage Requirements

CSA	Storage
Base requirement	500 Bytes
Control region RTT and Dependent region RTT	Varies according to customer definition
Total	500 Bytes + RTT size

All CSA allocations are ECSA (above 16M).

Access Manager for IMS TM Adapter Storage Requirements

Table 16–6 Adapter Storage Requirements, Control Region

Control Region	**Storage**
Base requirement	7524K

All control region allocations are extended private (above 16M).

Table 16–7 Adapter Storage Requirements, Dependent Region

Dependent Region	**Storage**
Base requirement	7524K
Total	7524K + application storage

All dependent region allocations are extended private (above 16M).

17 Error Diagnosis and Reporting

This chapter discusses the diagnosis of suspected Oracle database errors and the requirements for documenting these errors to Oracle Support Services. Specific topics in this chapter include documentation requirements, categorization of errors, system dump requirements, and methods of reporting to Oracle Support Services.

For information on OS/390-specific error messages, refer to the *Oracle9i Enterprise Edition Messages Guide for OS/390.*

The following topics are included:

Oracle Support Services

Oracle Support Services acts as the interface to the Oracle database user community. Refer to the applicable Oracle Support Services publications for a discussion of policies and procedures for using their services.

Providing Error Documentation

During the error resolution cycle, Oracle Support Services might request you provide them with machine readable data. Send machine readable data, not formatted or printed data. Magnetic tape is the most convenient form for sending large amounts of data.

If you are requested to send data to Oracle Support Services, then follow the documentation requirements provided in "General Documentation Requirements" later in this chapter. Failure to follow these requirements might result in the inability to process your tape. This could delay the resolution of any errors you are reporting.

Tape Format

Data must be sent on tape cartridges for the IBM 3480 or 3490 tape subsystem. Multiple files can be sent on a single volume. Each tape can be in standard label or no label format. In either case, include this written documentation with each tape:

- Volume serial, if it is a standard label tape
- Data Set name and label number for each file on each tape
- DCB characteristics of each data set

Tape Return

Oracle Support Services does not return tapes unless specifically requested to do so. The sending party must provide a return address on the tape and request that the tape be returned.

General Documentation Requirements

When you report a suspected error, you might be asked to describe the Oracle database subsystem and OS/390 operating system environments in detail. Provide the full version number of each component that has an error. The full version number includes important PUT levels for your OS/390 system.

Before you contact Oracle Support Services, ensure this information is available:

- Oracle database library naming conventions
- Method of accessing the Oracle database utilities (batch or TSO)
- Oracle database subsystem name
- Full version of the Oracle database kernel
- Full version of the Oracle database utility
- OS/390 release level
- PUT level
- RMID of any relevant OS module

In addition to describing the Oracle database operational environment, detailed documentation specific to the error might be required. This might include:

- Console logs
- Utility SYSOUT
- Utility input files
- System diagnostic messages
- Oracle database error messages
- System dumps
- Database engine trace data sets

Keep in mind that often more than one error is associated with a single failure. Describe all errors for the failure being reported. If your application uses Pro*C, Pro*COBOL, or another Oracle database Precompiler, then ensure your application displays or prints out all errors it encounters. Without a complete and consistent set of information, diagnosing the problem can be impossible.

Error Diagnosis

When investigating a potential Oracle server error, start by determining which component is failing, where it is failing, and the error category.

Components

When reporting a problem to Oracle Support Services, identify the component suspected of failure, along with its full version and correct release level. A list of components and their version numbers is documented in the *Oracle9i Enterprise Edition Installation Guide for OS/390* and any maintenance tape release bulletin.

Error Categories

Use these error categories to describe the error:

- Documentation
- Incorrect output
- Oracle database external error
- Abend
- Program loop
- Performance
- Missing functionality
- Wait state

Documentation

When reporting documentation errors, you are asked to provide this information:

- Document name
- Document part number
- Date of publication
- Page number

Describe the error in detail.

Documentation errors can include both erroneous documentation and omission of required information.

Incorrect Output

In general, an incorrect output error exists whenever an Oracle database utility produces a result that differs from written Oracle documentation. When describing errors of incorrect output, you need to describe, in detail, the operation of the function in error. Be prepared to describe your understanding of the proper function, the specific Oracle documentation that describes the proper operation of the function, and a detailed description of the incorrect operation.

If you think you have found a software bug, then be prepared to answer these questions:

- Does the problem occur in more than one Oracle tool? (Examples of Oracle tools are SQL*Plus and Oracle Developer/2000).
- What are the exact SQL statements used to reproduce the problem?
- What are the full version numbers of the Oracle database and related Oracle software?
- What is the problem and how is it reproduced?

Oracle Database External Error

Oracle database error messages are produced whenever an Oracle database utility or the Oracle database kernel detects an error condition. Depending on the circumstances, error messages might be fatal or nonfatal to the utility or kernel.

Be prepared to identify the exact error message and message number received and the complete circumstances surrounding the error.

Abend

Any program check in an Oracle database utility or the Oracle database kernel address is considered an error. A system dump is required as documentation in the event of a program check.

Ensure the system dump contains all of the private area of the Oracle database address space. Without it, diagnosis is sometimes impossible.

System abends might or might not indicate a failure of the Oracle database subsystem depending upon circumstances.

The following abends are not considered Oracle database failures:

- 013 - open failure
- 122 - cancelled by operator
- 222 - cancelled by operator
- 322 - CPU time exceeded
- 722 - SYSOUT lines exceeded

Program Loop

A program loop is evident when the Oracle server task consumes CPU time, but no actual work is performed. This situation is substantially different from an Oracle server task that performs most of its operations in cache (also known as a CPU bound job). CPU bound operations might include large batch sorts, sort merge joins, nested loop joins where the driving table is small enough to fit into the SGA, and so forth.

Any program loop that occurs within an Oracle server or utility address space is considered an Oracle database failure. Loop conditions are rarely experienced and are considered serious errors. The initial diagnostic approach with a loop consists of a system dump. If a task is in a program loop, then ensure the system dump includes all of the private area of the Oracle database address space.

Further diagnosis might be required using OS/390 SLIP commands. Oracle Support Services furnishes specific instructions on the use of SLIP depending upon circumstances.

Performance

Oracle database system performance is determined by many factors, most of which are not within the control of Oracle Corporation. Considerations such as system load, I/O topology, and database design make the documentation of performance errors difficult.

Provide detailed information about the state of your environment when reporting an error.

Specific documentation might include:

- CPU type and memory configuration
- Database topology
- I/O topology
- System workload by type
- Oracle database workload characterization
- Query execution plans

Missing Functionality

Enhancement requests can be opened with Oracle Support Services to request the inclusion of functions and features that Oracle products do not currently have. When opening an enhancement request, describe the specific feature or function to be added to the product and provide a business case to justify the enhancement.

Wait State

A wait state occurs whenever a required system resource (for example, an enqueue) is unavailable. Because the task requiring the resource is in a blocked or wait state, little or no CPU time is consumed. However, wait states are not limited to a lack of operating system resources. A wait state can occur within the Oracle server due to an incompatible lock request, high contention on an internal latch, an archive task that halts, and so forth.

A wait state in an Oracle database utility might or might not be considered an Oracle database error. A wait state that occurs due to operating system resource conflicts (for example, multiple requests for a tape mount) is not considered an Oracle database error. If a wait state occurs within an Oracle database utility, then a system dump is required of both the utility and kernel address spaces. If a wait state occurs in the Oracle server address space, then a system dump is required of that address space.

Diagnosing Wait State Problems

Before you contact Oracle Support Services, try these recommended approaches:

- Check the Oracle database alert log and system console logs for any error messages.
- Query the V$LOCKS dynamic view to determine whether there are any lock conflicts.
- For a high volume online transaction processing (OLTP) system, query the V$WAITSTAT dynamic view to determine whether there is contention for a class of data blocks. To perform this query, invoke SQL*Plus and enter these commands:

  ```
  SQL> CONNECT / AS SYSDBA
  SQL> SELECT * FROM V$WAITSTAT;
  ```

  For more information about the V$WAITSTAT view, refer to the *Oracle9i Database Reference*.
- If your database is in ARCHIVELOG mode, then ensure your archive data set destination is not full. If the destination is full, then the archiver task cannot copy any more full redo log data sets, thus preventing any further changes to the database. In this case, all tasks attempting to make changes are eventually placed in a wait state.

If you cannot diagnose the wait state problem using the previous suggestions, then first obtain these dumps and contact Oracle Support Services:

- An SVC system dump that includes all of the private area of the Oracle database address space.
- A SYSTEMSTATE dump from within Oracle database. When you contact Oracle Support Services, tell them you performed this task.

  To obtain a SYSTEMSTATE dump, invoke SQL*Plus and perform these commands:

  ```
  SQL> CONNECT  / AS SYSDBA
  SQL> ALTER SESSION SET EVENTS
  'immediate trace name systemstate level 10';
  ```

  Be sure to replace the `immediate trace name` with a value suitable for your task.

  A SYSTEMSTATE dump can be quite large, depending on the number of concurrent connections to the Oracle server, the number of resources each connection holds or requests in any mode, and the size of the SGA. The current cursor and object state for each concurrent connection is also included in the SYSTEMSTATE dump. The size of these dumps can easily be in the multiple of megabytes. A SYSTEMSTATE dump is synonymous with a SYSTEMSTATE trace data set.

System Dumps

When providing documentation on suspected Oracle database failures, it might be necessary for you to provide a system dump of the Oracle server or utility address spaces. Dumps are initiated through the OS/390 operator interface using the DUMP and SLIP commands, or automatically by Oracle database if it detects a problem.

Dumps sent to Oracle Support Services as documentation for suspected errors must not be formatted. Formatted dumps will not be accepted.

When specifying dump parameters in response to an OS/390 DUMP COMM=(' ') command, you must include this specification:

```
PSA,TRT,RGN
```

Additional parameters may be required.

System Dump Data Sets

Once a SYS1.DUMP`xx` data set is created, the system operator is notified whenever a dump to that data set occurs. Because all Oracle database abends are dumped to SYS1.DUMP data sets and are not dynamically allocated, you must ensure a SYS1.DUMP data set is always available.

You must also ensure the SYS1.DUMP data set is large enough to accommodate a dump of two address spaces (the Oracle database address space and the client address space). This allows for a complete dump if an Oracle database utility abends while in cross memory mode. Refer to "Oracle Server Storage Requirements" on page 16-6 in Chapter 16, "Oracle9i Performance", for more information about estimating the size of an Oracle database address space.

If a SYS1.DUMP data set is not available, then a dump might be lost.

Operator Initiated Dumps

Operator initiated dumps are accomplished with the OS/390 DUMP command:

```
DUMP COMM=(text)
```

where `text` is the title you want the dump to have.

After the DUMP command has been issued, you must respond to the system WTOR with the following:

```
R xx,[JOBNAME=(jobn)|ASID=(nnn),]SDATA=(PSA,TRT,RGN)
```

where:

`xx`	is the reply identification number.
`jobn`	is the name of the started task or batch job.
`nnn`	is the hexadecimal address space identifier of the address space you want to dump.

New operands can be requested.

SLIP

Proper documentation of an Oracle database error might require the setting of a SLIP trap. If this is the case, then contact Oracle Support Services for specific instructions.

TSO System Dumps

TSO symptom dumps provide a preliminary view of an abend condition in an Oracle database utility. A symptom dump is available by issuing the TSO command:

```
PROFILE WTPMSG
```

The effect of this command is permanent until you issue the command:

```
PROFILE NOWTPMSG
```

GTF

You might need to use GTF as a diagnostic tool under certain circumstances. Oracle Support Services provides specific instructions if this is the case.

18

Migration and Upgrade Considerations

This chapter addresses OS/390-specific considerations in converting existing Oracle for OS/390 database instances, TNS or Oracle Net services, and Oracle Access Managers to Oracle9*i*, Release 2 for OS/390. It supplements the generic product manual *Oracle9i Database Migration: Release 2 (9.2)*. Migration and upgrade considerations for OS/390-based Oracle application programs, tools, and utilities are covered in the *Oracle9i Enterprise Edition User's Guide for OS/390*.

Your existing Oracle for OS/390 database may be MPM-based, meaning that it uses the MPM subsystem for the database server and the TNS subsystem for SQL*Net or Oracle Net services. Oracle7 for OS/390 and Oracle8, Release 8.0 for OS/390 were provided only as MPM/TNS implementations. Oracle8*i* for OS/390 was delivered in both MPM/TNS and OSDI implementations. Oracle9*i*, Release 1 for OS/390 was delivered only as an OSDI implementation. It is important to note that moving from MPM/TNS to OSDI involves additional considerations covered only in this manual and in the *Oracle9i Enterprise Edition User's Guide for OS/390*.

The following topics are included:

Overview

Oracle uses the terms "migration" and "upgrade" to distinguish two categories of database conversion to a new software release. In an upgrade, you shut down the database, start it up with the new software release, and then execute provided SQL and PL/SQL scripts to update Oracle's internal dictionary tables and other objects. On successful completion of the scripts, your conversion is complete. In most upgrade situations Oracle also supplies "downgrade" scripts which can be run to return the dictionary to its original structure. Thus, an upgrade is relatively simple and, if there are problems with the new release, there is an easy fallback path.

A major release change may require changes to the structure of the Oracle control file. This conversion involves additional steps to extract data from your current control file and use it to build new control files when starting up with the new release. This scenario is called migration and is slightly more complex than an upgrade. Like an upgrade, migration also requires running scripts to update Oracle dictionary objects. Unlike an upgrade, however, migration usually does not provide a direct downgrade procedure for fallback: you must back up your database prior to the migration and restore the backup in a fallback situation.

If you are converting an Oracle7 database to Oracle9*i*, R2, your conversion is a migration: you will be replacing your Oracle7 control files with new Oracle9*i*, R2 control files. This migration requires that your Oracle7 database be at release 7.3.3.6.3 or higher; if it is an older release, you must upgrade to release 7.3.3.6.3 before beginning your migration to Oracle9*i*, R2. Since Oracle7 was provided only in an MPM implementation, you will also be moving from the MPM/TNS implementation to OSDI.

Note: *Oracle9i Database Migration: Release 2 (9.2)* states that an Oracle7 database must be at release 7.3.4 to be migrated to Oracle9*i*. This is not so on OS/390. The migration preparation fixes from release 7.3.4 were backported to release 7.3.3.6 on OS/390.

If your existing database is Oracle8, Release 8.0.6, Oracle8*i*, Release 8.1.7, or Oracle9*i*,R1, your conversion is an upgrade. A simple downgrade procedure is available for returning to Oracle8*i*, Release 8.1.7, or Oracle9*i*, R1; this is not available for 8.0.6. If the existing database is Oracle8, Release 8.0 or the MPM-based Oracle8*i*, Release 8.1.7, you are also moving from MPM/TNS to OSDI. Otherwise, you are already using OSDI and do not have to be concerned with MPM-OSDI transition issues.

Differences Between MPM/TNS and OSDI

OSDI provides a different OS/390 architecture and "internals" for executing the Oracle database server and Oracle Net components. This changes a number of OS/390-specific aspects of how these components are configured, started, and managed. Highlights of these differences are as follows.

One Subsystem, Many Services

With MPM, each Oracle instance and each TNS service is a separate OS/390 subsystem with a single associated address space. The OSDI subsystem has no dedicated address space of its own and it manages multiple Oracle database instances and Oracle Net services--as many as you choose to configure and run. While you can have multiple OSDI subsystems on one OS/390 image, there usually is no need to.

You must create at least one OSDI subsystem, and within it you must define and configure the database and/or network services you want to run. These definitions use the OSDI DEFINE SERVICE command and each specifies the particulars of the service, including a service name and the JCL procedure used to start a service address space.

Services Operate Differently

With MPM, Oracle database instances and TNS services could be started with an OS/390 START command (so they ran as started tasks or STCs) or they could be submitted to run as batch jobs. With OSDI, database and network services must be started with an OSDI START command and they execute as OS/390 "system address spaces." This mechanism relies on JCL procedures in a system procedure library. These procedures differ from the JCL you used for MPM and TNS.

All of the MPM- and TNS-specific operator commands and SYSLOG messages are gone with OSDI. There is a small set of OSDI subsystem operator commands for defining, altering, displaying, starting, and stopping database and network services. Individual database and network services also respond to OS/390 STOP (P) and MODIFY (F) operator commands. OSDI operator commands are issued through standard OS/390 command interfaces such as a system console, Netview, TSO SDSF, or MGCR (SVC 34). OSDI has no equivalent to the MPMCMD program. Relatively few SYSLOG messages are issued by OSDI database and network services. Those that are use the message id prefixes MIR and MIN, respectively. Messages from the OSDI subsystem component use prefix MIS. All of these "MI" messages are provided in US English only.

Database services under OSDI can execute as multiple OS/390 address spaces to increase available server virtual memory beyond the 2 gigabyte architectural limit. Additional address spaces beyond the first, called auxiliary address spaces, appear as distinct system address spaces similar to the first address space. Depending on how you configure the service, auxiliary address spaces are started automatically when the service is first started or they can be started manually (with an OSDI START command) any time after the first address space has initialized. Auxiliary address spaces do not respond to operator commands and cannot be terminated individually; they terminate only when the entire service is stopped.

Oracle database startup and shutdown are independent of database service start and stop with OSDI. With MPM, database startup could be invoked automatically within the MPM address space and database shutdown always caused termination of the MPM address space. Neither of these is so with OSDI: database startup is done independently of starting the database service address space(s) and database shutdown does not cause the address spaces to terminate. Stopping and restarting of database service address spaces is not required except when performing software maintenance, modifying the service JCL, and in certain failure situations.

Network Architecture Changed

With MPM and TNS, a separate network "listen" was initiated for each Oracle database instance that a remote client could access. This required executing an additional task called MPMTNS in each MPM/Oracle address space. Since each listen was on a distinct network endpoint (TCP/IP hostname and port), remote clients had to specify the correct endpoint address for the OS/390 database instance they wanted to access.

With OSDI, the network service listens on a single network endpoint for all database instances. Remote clients specify this endpoint address regardless of which OS/390 database instance they want to access. The client's target database instance is identified by a logical name called a SID; each OSDI database instance on a given OS/390 image has a distinct SID. Database instances do not need special configuration or a special task like MPMTNS in order to be accessed by a remote client. Unless prevented with the SAF mechanism discussed below, all OSDI database instances are accessible to remote clients who can supply a valid Oracle logon.

With MPM, network client database requests were executed using MPM worker tasks that were subtasks of the MPM jobstep program. With OSDI, network client requests are executed using preemptable enclave SRBs created by the network service address space. OSDI network client connections are subject to OS/390

Workload Manager (WLM) classification and can be independently prioritized relative to each other and to other work in the system.

Security Changes

OSDI provides built-in System Authorization Facility (SAF) checks at various points. Enabling these checks involves defining resources to your security subsystem (e.g. RACF) and granting access to those resources to the appropriate OS/390 userids. Checks are provided for:

- **OSDI Operator Commands** This controls which OS/390 consoles or users can issue various OSDI subsystem commands. (The OS/390 STOP and MODIFY commands used with OSDI services are controlled by existing OS/390 mechanisms.)
- **OSDI Bind Process** This controls which OS/390 address spaces can establish cross-memory connections to OSDI services. This check can be used to disable Net access to a given Oracle database instance.
- **SYSOPER and SYSDBA** This controls who can log on to an Oracle database service using AS SYSDBA and AS SYSOPER, which enable Oracle operating privileges such as STARTUP.
- **Oracle Logon** The most common form of SAF-based Oracle logon authentication, which was accomplished using an exit routine with MPM, is built in to OSDI. Exit routine capability is still provided for installations that have specific processing needs not provided by the built in feature.

Certain details of SAF- or exit-based Oracle logon authentication have changed with OSDI relative to MPM. Support for the role processing exit (in conjunction with init.ora OS_ROLES=TRUE) is discontinued with OSDI.

Parameter Changes

For Oracle database instances, all MPM parameters are gone. A small set of parameters control the database service address space. Also, a separate parameter file, called ORA$FPS, provides parameters related to creation and processing of OS/390 data sets in the Oracle server.

The interpretation of certain Oracle RDBMS parameters (usually referred to as INITORA or init.ora parameters) is changed with OSDI. Particularly important are the parameters related to log archival processing.

Database File Processing Changes

The syntax for specifying Oracle control, log, tablespace, and other files used by the server is changed with OSDI. (Backward compatibility is provided for syntax commonly used with MPM.) As mentioned in the prior section, file processing parameters are supplied in a separate parameter file rather than being appended to the file name supplied in a SQL statement. All database files created by an OSDI server--including archive logs--are VSAM Linear Data Set (LDS) clusters. Existing VSAM ESDS clusters with 4K Control Interval size continue to be accepted for compatibility, but may need to be converted to LDS in certain situations discussed later. Archive logs created as sequential (DSORG=PS) data sets under MPM must be converted to LDS to be used in recovery with an OSDI server.

Backup and Recovery Changes

Backup and recovery based on Oracle Recovery Manager (RMAN) is significantly different with OSDI. Existing procedures based on the MPM implementation of RMAN will require changes for OSDI.

SMF Record Changes

The formats of the Oracle session statistics SMF record and the Oracle audit trail SMF record are changed with OSDI.

Unsupported Features in OSDI

IXCF Protocol

Support for Oracle Net PROTOCOL=IXCF has been discontinued. IBM's TCP/IP can be configured to use XCF connections between nodes in a sysplex, so PROTOCOL=TCP provides a substitute mechanism.

Computer Associates or Interlink SNS/TCPaccess Support

SNS/TCPaccess is no longer supported as a separate Oracle Net driver, support is provided via the HPNS interface of the IBM TCP/IP driver.

OSDI Changes in Oracle9*i*, Release 2

Oracle9*i*, R2 introduces several OS/390 architectural changes described here. Read this section if you are moving to Oracle9*i*, R2 from any earlier Oracle release, including Oracle9*i*, R1.

Network Client Operations

Starting with Oracle9*i*, R2, OS/390 Oracle client programs that use Oracle Net open network protocol connections directly rather than using the Oracle Net service. This affects customer-written Oracle applications running in CICS/TM, TSO, and batch and Oracle tools or utilities running in TSO and batch. It also affects situations where an OS/390 database instance operates as a "network client", including opening network database links and connections for the UTL_HTTP PL/SQL package. It does not affect Oracle applications, tools, or utilities running in the OS/390 USS environment: these programs are already using network protocol services directly.

Previously, the network protocol interaction for non-USS applications was executed by the network service address space on the client's behalf. Now it is executed directly from the client's OS/390 address space and TCB. This change is expected to improve performance and eliminate client operational dependence on the network service. It has the following external considerations:

- **OMVS dubbing**. IBM's TCP/IP protocol makes use of OS/390 Unix System Services (USS). An OS/390 address space that uses IBM TCP/IP must therefore be a USS process or be capable of being "dubbed". Dubbing occurs automatically when needed but it requires the OS/390 userid associated with the address space to have a default USS segment defined in the security subsystem (e.g. RACF). If dubbing fails, the network connection will not be opened and the application probably will fail. You must ensure that all OS/390 clients that connect to remote Oracle servers can be dubbed. If in doubt, discuss this with your OS/390 security administrator.
- **No client dependence on network service**. With this change, stopping and starting the Oracle network service has no effect on OS/390 client network connections. OS/390 clients no longer require a TNS@xxx DD statement or other specification o identify a network service to use, and they no longer require OSDI bind authority to the network service. The network service is now involved only with inbound network operations (clients on other systems who are connecting to an OS/390 database instance).

Clients Use Language Environment

Beginning with Oracle9*i*, R2, all Oracle client software on OS/390, including Oracle tools, utilities, and the Oracle program interface code used by customer applications, uses IBM Language Environment (LE) for C program runtime services. LE replaces the MVS Oracle C runtime library in use since 1986. This change is primarily internal in nature but it does affect a few externals, including the syntax used to specify files and certain PARM and command line features. It also requires that LE runtime (e.g. SYS1.SCEERUN) be available in the system linklist or JOBLIB/STEPLIB when running Oracle tools, utilities, or customer-written applications.

Detailed information on this change is provided in the Oracle for OS/390 User's Guide. Since Oracle database administration involves running Oracle tools and utilities, refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* as necessary when performing administration tasks discussed in this manual.

Moving from MPM/TNS to OSDI

This section discusses OS/390-specific issues in moving existing MPM/TNS Oracle databases, network services, and Access Managers to OSDI. It is intended to serve more as a guide or checklist than as complete documentation: the actions discussed here are taken in conjunction with the generic migration or upgrade steps covered in the next major section and in *Oracle9i Database Migration: Release 2 (9.2)*.

Configure OSDI Command Security

If you are going to use SAF-based OSDI command security, enable it by defining appropriate resources in the security subsystem (e.g. RACF) and then granting appropriate rights to OS/390 userids. Do this before initializing the subsystem if you want command security to be in effect as soon as the subsystem is active. You may need to coordinate this effort with your system security administrator.

For detailed information on OSDI command security, refer to Chapter 8, "Security Considerations".

Configure the Subsystem

You will need to initialize at least one OSDI subsystem in order to configure and run Oracle database instances and network services with OSDI. This can be done any time prior to the point where an Oracle9*i*, R2 database or network service must be started. If your installation permits it, the subsystem can be initialized without an OS/390 IPL by using the SETSSI ADD operator command. Otherwise, you will need to add a subsystem specification to SYS1.PARMLIB and IPL the system to initialize the subsystem. If appropriate, work with your OS/390 system administrator to coordinate this.

Choose a unique subsystem name for OSDI. Even if you are installing OSDI on a different OS/390 system than the one where existing MPM and TNS services run, avoid using the same subsystem name you used for an MPM or TNS instance. Unlike MPM and TNS, OS/390 clients never need to know or specify the OSDI subsystem name. OSDI clients use a SID, a unique name assigned to each service, to identify an Oracle database instance to which they want to connect. Starting with Oracle9*i*, R2, outbound OS/390 network clients (ones running on OS/390 and connecting to a remote server) do not use the OSDI network service at all, so there is no requirement to specify or identify the network service in client applications.

Part of OSDI subsystem initialization includes reading and processing OSDI commands from a sequential file or PDS member. Normally this is where you will place DEFINE SERVICEGROUP, DEFINE SERVICE, and possibly other commands to configure the database and network services you will run. Therefore, read the following section and its related material before trying to initialize your subsystem. It is much easier to supply the OSDI definition commands in a file than to enter them "live" at a system console.

For details on OSDI subsystem definition and initialization, refer to Chapter 2, "Configuring and Initializing the Subsystem".

Configure Database Services

You must configure an OSDI database service for each MPM Oracle server you are converting. Normally all such services are defined in the same OSDI subsystem, but this is not required. Use the DEFINE SERVICE command of OSDI to create the OSDI structures used to manage the service. Normally this command would be placed in the data set read by OSDI subsystem initialization, but it can be issued at a system console or comparable facility. In the latter case, processing of the command is subject to SAF security checks if you have defined the appropriate resources to the security subsystem. The DEFINE SERVICE command must be issued before you can attempt to start the service.

Details on configuring a database service are found in Chapter 3, "Configuring a Database Service and Creating a New Database". What follows are some specific topics relevant to moving from MPM to OSDI.

SID

When you define an OSDI service, a SID is specified or assigned by default. In the case of a database service, the SID is the name that both local and remote clients use to identify the Oracle instance to which they want to connect. Thus the SID serves a purpose similar to the MPM subsystem name. In most cases you will want the SID of a database service to be the same as the subsystem name of the MPM instance you are converting. This is especially so if you have batch jobs or TSO scripts that use an ORA@ssn DD statement to specify a target Oracle server. With OSDI, the "ssn" part of this DD name is interpreted as the SID of the target database service, so using a SID that matches the MPM subsystem name means those batch jobs or scripts need not be changed.

If you do not specify a SID on your DEFINE SERVICE command, the SID defaults to the service name. For complex reasons having to do with OS/390 subsystem facilities, you do *not* want an OSDI service name to match the subsystem name of any MPM, TNS, or other subsystem you might have. Therefore, to have a SID that matches the MPM subsystem name, SID should be specified explicitly on your DEFINE SERVICE command, and will differ from the service name. For example:

```
DEFINE SERVICE ORADB1 TYPE(ORA) SID(ORA1) PROC(SORA1C) -
 DESC('Oracle9i R2 Test Instance')  MAXAS(3) -
 PARM('ORACLE.ORADB1.PARMLIB(SRVPARM1)')
```

Database Region Parameters

Your DEFINE SERVICE command for a database service will specify as a PARM string the name of a sequential data set or PDS member containing OSDI database region parameters. Although logically equivalent to MPM parameters, these are completely different and relatively few in number. The data set or PDS member need not exist when the DEFINE SERVICE command is issued, but it must exist when the database service is started with an OSDI START command. Details on these parameters are covered in Database Region Parameters on page 3-10.

Database JCL Procedure

Your DEFINE SERVICE command must specify the name of a JCL procedure used to start address spaces of the service. This procedure must be in a system procedure library such as SYS1.PROCLIB or one that is concatenated to it in the JES subsystem. The procedure need not be in place when DEFINE SERVICE is issued, but it must be when you start the service with an OSDI START command.

The database service JCL procedure is completely different from that used with MPM. It executes a different jobstep program and has different required and optional DD statements. Details on this JCL procedure are found in "Database Region JCL" on page 3-6.

Database File Processing

In the latter stage of your migration or upgrade, the Oracle9*i*, R2 server will mount and open your database, causing the OS/390 data sets comprising the database to be opened and accessed. Although the internal file access methods used by OSDI differ from those used by MPM, the change is almost entirely internal and requires no special action on your part. There are, however, some external considerations in OSDI file processing.

Chief among these is the use of separate file processing parameters. With MPM, certain parameters related to data set creation could be including in the file specification string in a SQL statement such as CREATE DATABASE or ALTER TABLESPACE ADD DATAFILE. This is no longer supported with OSDI: file processing parameters, organized by Oracle file type, are supplied in a parameter file, identified by an ORA$FPS DD statement, that is read by the OSDI database region program. Depending on how a particular Oracle instance is configured and used, you may need to establish parameters for one or more of the various Oracle file types. Refer to Chapter 4, "Defining OS/390 Data Sets for the Oracle Database" for details on Oracle database files and the ORA$FPS parameter file.

Another difference between MPM and OSDI is that with OSDI, all data sets created by the server that are part of the database are VSAM LDS clusters. With MPM, certain data sets, such as archive logs, were created as sequential non-VSAM (DSORG=PS) data sets. Any non-Oracle processing of Oracle archive logs, such as copying or migration procedures, should be examined to make sure that they will operate as expected with VSAM LDS clusters. Archive logs created by an MPM server must be converted to VSAM LDS before being used in recovery operations with an OSDI server. Refer to the example under "CFUTIL (Convert File Utility) Reference" on page 18-35 for further information on converting to VSAM LDS files.

Another topic in the area of file processing concerns the syntax for specifying database files to Oracle in SQL and in parameter files such as init.ora. OSDI uses IBM Language Environment syntax for specifying files and this syntax differs from that used by the MPM implementation. The old MPM syntax continues to be accepted: file specifications containing "/DSN/" and "/DD/" prefixes are accepted in both SQL (e.g. CREATE TABLESPACE) and in init.ora parameters (e.g. CONTROL_FILES). JCL-like keyword parameters coded after the data set name in such specifications are tolerated but ignored. Oracle recommends that you begin using the new syntax described in Chapter 4, "Defining OS/390 Data Sets for the Oracle Database". Compatibility for the MPM syntax will be dropped in a future Oracle release.

Except for the name(s) of the control files themselves, the names of all files comprising a database are stored in the control file. In the case of an MPM database, this means they are stored in MPM syntax such as "/DSN/ORA1.ORADB1.DBF". The compatibility feature mentioned above means that these file names will be processed correctly when you start up Oracle9*i*, R2. However, leaving the file names in the control file in this form can create problems later: it can cause Oracle to fail to detect if you are mistakenly adding a file to the database that is already part of the database. Because of this exposure, a special OS/390 utility program called CFUTIL is supplied that will update the control file so that all file names are in the new (OSDI or LE) syntax. During a migration or upgrade that involves moving from MPM to OSDI, you will run CFUTIL against your control file(s) to perform this update. In a fallback situation, where you are returning to MPM after running OSDI, you must run CFUTIL again to restore the old syntax: an MPM Oracle instance does not accept the new OSDI syntax.

The CFUTIL program also performs other file-related functions that are relevant to an MPM to OSDI move. Details on this utility are found in "CFUTIL (Convert File Utility) Reference" on page 18-35.

RDBMS Parameter Considerations

Since you are migrating or upgrading to a new release of Oracle, you should review the database initialization parameters (commonly referred to as "init.ora" parameters) in the *Oracle9i Database Reference*. Depending on the release you are coming from there will be some number of new or changed parameters, so your current set of parameters may need changes. In addition, the interpretation of a couple of init.ora parameters is changed by OSDI:

CONTROL_FILES. If you are migrating from Oracle7, you will be creating new control files, so the names specified here will change. Use data set names rather than DD names to specify the control files. This parameter can be changed to use the new LE-style syntax for file names, although the old syntax is still accepted.

LOG_ARCHIVE_. The interpretation of LOG_ARCHIVE_FORMAT, LOG_ARCHIVE_DEST, and LOG_ARCHIVE_DUPLEX_DEST is changed with OSDI. This is further complicated by the fact that generic Oracle9*i* has adopted new parameters for log archiving: the old parameters are accepted for compatibility but the new LOG_ARCHIVE_DEST_n and several related parameters provide additional functionality. Whether you switch to the new parameters or not, moving to OSDI dictates changing the values specified if log archiving is to work correctly. Refer to the discussion in Chapter 3, "Configuring a Database Service and Creating a New Database" and to the generic Oracle documentation for the old and new parameters.

Database Backup and Recovery Procedures

Non-RMAN-based backup and recovery considerations are mostly unchanged by OSDI. You should check to make sure that the use of VSAM LDS clusters (as opposed to ESDS) does not affect whatever software you are using for backups. If you have non-Oracle processing of Oracle archive logs, you will need to make sure that processing accommodates the VSAM LDS clusters written by OSDI instead of the sequential non-VSAM data sets written by MPM.

If you are using RMAN for backup and recovery of an MPM Oracle database, you will find significant changes in OSDI. Backup information stored by MPM RMAN, whether in the control file or an RMAN backup catalog, is not usable with OSDI. Most RMAN scripts developed for MPM will require changes for OSDI.

With OSDI, RMAN backups to the DISK device type are created as VSAM LDS clusters. Most installations that create DISK backups will need to add parameters to the server ORA$FPS for the relevant file types. For non-DISK backups, OSDI employs an External Data Mover (EDM) which runs in a separate OS/390 address space from the server. Using EDM requires a JCL procedure in a system procedure library and it changes (relative to MPM) what is specified in the RMAN ALLOCATE CHANNEL statement. In addition, interpretation of backup file specifications and various other backup and recovery considerations are different with OSDI and EDM.

MPM RMAN users do not necessarily have to convert all of their RMAN procedures prior to moving to OSDI, but they must do so before attempting RMAN backup and recovery operations with OSDI. Oracle recommends that these procedures be examined and modified as necessary as early in the conversion as possible. Refer to Chapter 6, "Database Backup and Recovery" for details on the OSDI implementation of RMAN.

Database Security Considerations

Several SAF-based OSDI security features relate directly to the database server. Depending on whether your installation chooses to use these features, they must be configured or activated before or during the move to OSDI. Complete details on these features are provided in Chapter 8, "Security Considerations". Highlights of the areas of interest follow.

Bind Security This mechanism controls which OS/390 users are permitted to establish cross-memory connections to an OSDI server. There are two categories of bind: the normal application bind used by an OS/390 batch job, TSO user, or USS process, and a "managed" bind used by Access Managers and by Oracle Net (operating on behalf of incoming remote clients). The two bind schemes are distinguished because the managed scheme carries special privileges that should not be made available to normal applications.

MPM had no equivalent security mechanism, so there are no conversion issues per se with bind security. If your installation elects to activate bind security (by defining the required resources in a security subsystem such as RACF), appropriate access rights must be granted to OS/390 userids that will connect to local Oracle servers. This includes the OS/390 userid that will perform the Oracle STARTUP during the migration or upgrade.

Note that beginning with Oracle9*i*, R2, OS/390 clients who connect to remote Oracle instances via Oracle Net do *not* require bind authority to the Oracle Net service.

SYSOPER/SYSDBA Security This mechanism controls which OS/390 users can connect to a local Oracle server using the AS SYSOPER or AS SYSDBA option. These options enable certain operating privileges such as the ability to perform an Oracle STARTUP. This type of connection replaces the CONNECT INTERNAL used in older Oracle releases. CONNECT INTERNAL is not supported in Oracle9*i*.

MPM, via the PRIVUSER parameter/command, maintained its own list of OS/390 userids that were permitted to CONNECT INTERNAL. With OSDI, the equivalent control is implemented by defining resources in the security server (e.g. RACF) and then granting access to the resource names. Examine your MPM instance's PRIVUSER list and discuss your requirements with your system security administrator.

To bring up Oracle9*i*,R2 during migration or upgrade, an OS/390 user must connect to the server using CONNECT / AS SYSDBA. Make sure that the required security configuration work has been done before you reach that step.

Oracle Logon Authentication With MPM, Oracle logon password verification for Oracle users defined as IDENTIFIED EXTERNALLY could be delegated to a security subsystem such as RACF using an exit routine. Although the exit capability is still provided with OSDI, the most common form of external verification is provided as a built-in feature: no exit is required. The feature uses RACROUTE REQUEST=VERIFY, a SAF request, which works with any

SAF-compatible security subsystem. You activate this check by specifying LOGON_AUTH(SAF) in the OSDI database region parameter file.

If your installation requires additional processing beyond this RACROUTE check during Oracle logon, you need to convert your MPM logon exit to OSDI. The logon exit interface provided by OSDI is different from MPM's, so you may find it easier to move a portion of your old exit's logic into the sample exit that is shipped with OSDI (member RACFSMPO of the SRCLIB data set).

If you are using either the built-in SAF check or a converted logon exit, there are differences in processing in OSDI that you need to be aware of. Foremost of these is that the SAF check or logon exit is called only for userids defined to Oracle as IDENTIFIED EXTERNALLY, and only when the user or application attempts to connect with an explicit userid/password string. No SAF check is performed (or no call is made to the exit) when such a user connects with a null userid and password string, usually signified by a standalone slash (as in "CONNECT /"). In this case, the assumption is that the user was authenticated when logging on to OS/390 and need not be authenticated again.

Under MPM, if a logon exit was specified and the init.ora parameter OS_AUTHENT_PREFIX=OPS$ was specified or taken as the default, then users connecting with "/" (standalone slash) were logged on without the OPS$ prefix. For example, if user JLSMITH was created in Oracle and was IDENTIFIED EXTERNALLY, and then if a "CONNECT /" was used with the user logged onto OS/390 as JLSMITH, then the user was logged on to Oracle as JLSMITH, ignoring the OPS$ prefix which should have been used. With the OSDI, the OS_AUTHENT_PREFIX setting is always used. In the example just given, Oracle userid OPS$JLSMITH would have to be created (with IDENTIFIED EXTERNALLY), and a "CONNECT /" would connect OS/390 user JLSMITH to Oracle as user OPS$JLSMITH.

If this change in behavior presents a problem, then the init.ora parameter OS_AUTHENT_PREFIX should be specified as OS_AUTHENT_PREFIX="" (an empty string) to get the old behavior. With the parameter set in this manner, the user would be created as JLSMITH and IDENTIFIED EXTERNALLY, and "CONNECT /" would connect to Oracle as Oracle userid JLSMITH.

Another change in OSDI concerns any use of any non-null OS_AUTHENT_PREFIX value, OPS$ or otherwise. If LOGON_AUTH(SAF), LOGON_AUTH(RACFSMPO), or LOGON_AUTH(<user-written-exit-name>) is specified, and if a connect with an explicit userid and password is attempted, the SAF check (for LOGON_AUTH(SAF)) or the logon exit (for the other two cases) will be called with the OS_AUTHENT_PREFIX value removed. For example, if user OPS$JLSMITH is created and IDENTIFIED EXTERNALLY, and if a "CONNECT

OPS$JLSMITH/SKIFFEL" is then used, then the SAF check or logon exit will be called with JLSMITH as the userid.

Oracle SMF Recording Considerations

Like MPM, OSDI will write records to OS/390 System Management Facility (SMF) for session statistics and for the database audit trail (when init.ora AUDIT_TRAIL=OS is specified). The format of both types of records has changed, however, as have the parameters used to control SMF recording.

MPM used the SMFRECN parameter to specify the record number for the session statistics record, and SMFRECN defaulted to 199. OSDI uses the SMF_STAT_RECNO database region parameter for this, and it defaults to zero which means SMF session records are not written. You must specify SMF_STAT_RECNO if you want session statistics records from an OSDI server. Do not use the same record number you are using with MPM servers: the OSDI record has a different format, so you need to be able to distinguish MPM records from OSDI records.

For the database audit trail, when AUDIT_TRAIL=OS is specified, MPM obtains the SMF record number from the AUDIT_FILE_DEST init.ora parameter. With OSDI, the audit record number is a database region parameter SMF_AUDIT_RECNO. You must specify a nonzero record number for this parameter to enable writing of server audit trail records to SMF. As with the session statistics record, do not use the same number you are using for MPM audit records. The format of the audit record is changed with OSDI.

The changes in record formats mean that programs which process these records will need changes for OSDI. Sample formatting programs are included in the SRCLIB data set (members ORAFMTO and ORAFMTAO). For complete information on Oracle SMF recording refer to Chapter 9, "Oracle SMF Data".

Configuring Network Service

Beginning with Oracle9*i*, R2, the OSDI network service is involved only in inbound network operations, that is, when remote clients connect to an OS/390 Oracle instance. Outbound clients on OS/390--ones who are connecting to remote Oracle instances--interact directly with the TCP/IP protocol rather than using the network service. This is true for all Oracle9*i*, R2 clients: TSO, batch, USS, Access Managers, and even the Oracle database server when it operates as a client (e.g. when opening database links). If you were using the TNS implementation of Oracle Net only for outbound client operations, you may not need to configure and run OSDI network service at all. You do need to run OSDI network service for inbound client support and in some cases for providing backward compatibility.

The Oracle cross-memory protocols used by MPM and OSDI for local connections between OS/390 address spaces are not compatible. This means that an MPM client cannot connect via cross-memory to an OSDI server and an OSDI client cannot connect via cross-memory to an MPM server. It also means that cross-memory database links between MPM and OSDI servers are not supported. In these situations, you may need to use OSDI and TNS network services to enable connections between the dissimilar architectures...even within a single OS/390 system, where no real networking is involved. Refer to the section "Coexistence and Compatibility" on page 18-19 to see if you need to run OSDI network service to provide connection compatibility.

The OSDI network service is simple to configure. You must issue an OSDI DEFINE SERVICE command with TYPE(NET) to create the OSDI data structures used to manage the service. As with the database service, a JCL procedure must be installed in a system procedure library at some point before you attempt to start the service. Rather than reading a parameter data set, network service region parameters are specified directly in the PARM string of DEFINE SERVICE. Only TCP/IP is supported. A port number must be supplied; this is the port on which the service listens for inbound remote clients.

Beginning with Oracle9*i*, R2, the SID of the network service is irrelevant because outbound clients do not use the network service. If you have Oracle8*i* or Oracle9*i*, R1 clients making outbound network connections, the network service SID is significant since those clients must identify the network service they want to use.

As discussed earlier, OSDI database servers require no specific action or configuration to be accessible to network clients. All remote clients specify the same hostname (or IP address) and port number in the Oracle network address string or tnsnames.ora entry. The target database instance is specified using the SID parameter of CONNECT_DATA in the address string.

Configuring Oracle Access Managers

Oracle Access Managers for Oracle9*i*, R2 can access an Oracle9*i*, R2 local or remote database server. When accessing a remote server, Oracle Access Managers utilize the Oracle Net open network protocol with the OMVS dubbing requirement described under "Network Client Operations" on page 18-7.

In the case where a local database server is upgraded to Oracle9*i*R2, earlier versions of Oracle Access Managers can access the local database if the connection in the thread table (CICS) or RTT table (IMS TM) is changed from a local definition to a remote definition using TCP/IP.

Oracle Access Manager for CICS

Oracle9*i*, R2 Access Manager for CICS thread definitions remain consistent with earlier versions. To access this version, customer applications must be re-linked with a new application stub (ORACSTUB). Refer to Chapter 11, "Oracle Access Manager for CICS" for more information.

Operating Considerations

OSDI changes some of the fundamental operating considerations for Oracle database instances and network services. Foremost is the fact that services must be started with an OSDI START command and run as OS/390 system address spaces. In one of the later steps in your migration or upgrade to Oracle9*i*, R2 you will start the OSDI database service and then perform an Oracle STARTUP. This is significantly different from the comparable operations with MPM, as discussed in the following section.

Database STARTUP and SHUTDOWN

During migration or upgrade you will issue a STARTUP command to bring up the new release of the Oracle server. Several aspects of database startup and shutdown are different with OSDI. Some of these differences may require you to change operating procedures you have been using with MPM.

First, the database service you defined to OSDI must be started with an OSDI START command before you attempt an Oracle STARTUP. This will create the address space (or spaces, if so configured) for running the Oracle server, but it will not automatically perform an Oracle STARTUP as was possible with MPM. Oracle STARTUP and SHUTDOWN are performed separately by executing a utility such as SQL*Plus in batch, TSO, or USS. Normally the utility will connect to the service via OS/390 cross-memory services, so it must execute on the same system as the database service.

Since STARTUP and SHUTDOWN are performed from a separate utility process, progress messages appear in the standard output of the utility (e.g. SYSOUT, or the TSO terminal). They do not appear in the OS/390 system log as they did when STARTUP and SHUTDOWN were executed internally by MPM. Moreover, no OSDI messages are normally issued during Oracle STARTUP and SHUTDOWN, so the system log is essentially quiet during this process.

Unlike MPM, Oracle SHUTDOWN does not terminate the database service address space(s). These can and should be left running so they are available for the next Oracle STARTUP. You can STARTUP and SHUTDOWN Oracle as many times as you like without ever stopping service address spaces. The only time service address spaces must be stopped is when installing software maintenance, changing the service JCL, and in certain problem or failure situations (such as when Oracle SHUTDOWN is unable to complete).

Order of Database and Network Service Starts

Because of the architectural changes in OSDI, in particular the elimination of the MPMTNS task, the order in which database and network services are started does not matter. The network service must be started in order for remote clients to access Oracle database instances on OS/390, but there is no dependence on the network service in the Oracle server or in OS/390 clients.

Coexistence and Compatibility

This section discusses considerations in having both MPM/TNS and OSDI software on the same OS/390 system. This is permitted, of course, but components from one implementation may not work with components from the other.

Cross-Memory Protocol

The cross-memory protocol used between local clients and servers on OS/390 is not compatible between MPM/TNS and OSDI. This means that clients running from a pre-OSDI CMDLOAD library cannot connect to an OSDI database instance using either SQL*Net V1 cross-memory (e.g. the "M:" and "Z:" drivers) or with SQL*Net V2 or Net8 PROTOCOL=XM. It also means that client applications built with or upgraded to OSDI libraries cannot connect to an MPM-based Oracle instance via OSDI PROTOCOL=XM.

There are two ways to ensure compatibility for client applications. The simplest approach, for applications that only need to connect to one or the other architecture, is to have the client built with the correct architecture. In the case of a server upgrade to OSDI, this means relinking customer-written applications with the OSDI stub and running them with the OSDI version of the Oracle interface code (OSDI's CMDLOAD). For procedures or scripts that execute Oracle tools or utilities it just means using the OSDI CMDLOAD.

The other approach is to switch the client from cross-memory protocol to TCP/IP. This is easiest with applications that use a TNSNAMES file (e.g. TNSNAMES DD) to obtain the address of the target server. Such applications can be switched to TCP/IP by changing the relevant entry in the TNSNAMES file from PROTOCOL=XM to PROTOCOL=TCP along with the correct hostname or IP address, port number, and SID for the target OSDI server. If the connection address data (including a SQL*Net V1 type of address such as "M:" or "Z:") is embedded in the application program, or if the program relies only on an ORA@ssn DD statement to designate the target server, it must be modified and recompiled to switch it to TCP/IP.

Pre-OSDI applications that are switched to TCP/IP protocol can connect to both OSDI and MPM database instances by varying the target server address in the TNSNAMES file. When connecting to an MPM instance, the TNS subsystem must be running. When connecting to an OSDI instance, both the TNS subsystem and the OSDI network service must be running: the client goes out via TNS and comes in (to the OSDI server) via OSDI net.

Refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* for additional information on migrating OS/390-based applications from MPM/TNS to OSDI.

Database Links

Database links are a special category of Oracle connection where the "client" is one Oracle instance connecting to another. Database links use Oracle Net or SQL*Net just like normal client-server connections and they are subject to the same considerations. This means that cross-memory database link connections between OSDI and MPM servers are not supported for the reasons discussed above.

To enable database links between the two architectures requires using TCP/IP protocol for the link. If the database link's USING clause specifies a TNSNAMES identifier, only the entry in the associated server's TNSNAMES file needs to be changed to use TCP/IP. If the USING clause contains an explicit Oracle Net or SQL*Net address, the database link must be dropped and recreated with a different USING clause. In the latter case Oracle recommends changing to a TNSNAMES identifier in the USING clause and using the server's TNSNAMES file to control the network address particulars of database link targets.

To use a TCP/IP database link in an MPM server to access an OSDI server, both the TNS subsystem and OSDI network service must be running. To use one in an OSDI server to access an MPM server only the TNS subsystem must be running: starting with Oracle9*i*, R2, outbound network connections from the server do not use the OSDI network service.

Network Component Considerations

The Oracle Net components are closely coupled to the database server in architectural terms. This means that the OSDI network service can connect inbound remote clients only to OSDI database instances, and the TNS subsystem can connect inbound remote clients only to an MPM database instance. Assuming you are providing network client access to all OS/390 database instances, you must run both TNS and the OSDI network service for as long as you have both MPM and OSDI database instances.

The TNS subsystem also provides network services to outbound OS/390 clients, including outbound database links from MPM database instances. You must continue to run TNS as long as you have pre-OSDI client applications and/or MPM databases that open network database links.

OSDI client applications prior to Oracle9*i*, R2 (that is, client applications from OSDI Oracle8*i* and Oracle9*i*, R1) use the OSDI network service for outbound network services. Similarly, outbound database link connections from OSDI Oracle8*i* and Oracle9*i*, R1 use the OSDI network service. As long as you have either of these you must run an OSDI network service. Clients running with Oracle9*i*, R2 libraries and database links from Oracle9*i*, R2 servers no longer use the network service for outbound operations. However, the OSDI network service shipped with Oracle9*i*, R2 still contains the outbound services required by pre-Oracle9*i*, R2 clients and servers. This means you can upgrade your network service to Oracle9*i*, R2 and still support pre-Oracle9*i*, R2 OSDI clients and databases.

Access Managers

As with normal clients, the OSDI versions of the Access Manager products must be used for cross-memory access to local OSDI servers and cannot access an MPM server via cross-memory. As with other components, access between the different architectures requires using TCP/IP protocol.

Oracle Enterprise Manager Intelligent Agent and Data Gatherer

The OSDI version of the Intelligent Agent and Data Gatherer supports local OSDI Oracle instances only and is not compatible with local MPM Oracle instances. The tnsnames.ora file should be updated with the appropriate TNSNAMES entry using the PROTOCOL=XM specification to access local Oracle for OS/390 instances. The customization script must be rerun to configure the Oracle Enterprise Manager Intelligent Agent and Data Gatherer for OSDI.

Oracle for OS/390 Migration and Upgrade

This section describes OS/390-specific considerations in performing the migration or upgrade process documented in *Oracle9i Database Migration: Release 2 (9.2)*. If your migration or upgrade includes a move from MPM/TNS to OSDI, all of the preceding sections in this chapter must be read before beginning. Various aspects of the move to OSDI are mentioned here but the details are not. If you are already running an OSDI release of Oracle, you can disregard steps or considerations for moving to OSDI.

Certain parts of this material pertain only to migration, which is the process of converting from Oracle7 to Oracle9*i*, R2 as discussed at the beginning of this chapter. If your Oracle database is Oracle8, Release 8.0 or higher, you can disregard steps or considerations that pertain to migration.

Alternatives to Migration

Transforming an installed version of an Oracle7 database with MPM to Oracle9*i*, R2 with OSDI is possible by creating a new database under the new release, then copying data from the old database to the new database using either of the following two methods:

Export and Import

Export the desired data from the old database (using Oracle7 EXP) and import the data into the new database (using Oracle9*i*, R2 IMP).

SQL*Plus COPY

Use the SQL*Plus COPY command to copy data from the old database to the new one. This requires using Oracle Net to access either the old or new database, because no version of SQL*Plus can connect to both MPM and OSDI servers using cross-memory.

The advantage to these methods of transformation is that they do not disturb your Oracle7 database. They do, however, require that the two databases exist simultaneously, which might be a problem if your database is very large. Also, both of the copying methods that are described above are relatively slow. Migration, using the migrate (MIG) utility, does not move database data from one version to another, and is therefore much faster. It does, however, require that you back up your Oracle7 database completely.

Migration and Upgrade Steps

Step 1: Configure OSDI Database Service

Do this only if you are moving from MPM/TNS to OSDI. Refer to earlier sections of this chapter and Chapter 2, "Configuring and Initializing the Subsystem", Chapter 3, "Configuring a Database Service and Creating a New Database", and Chapter 4, "Defining OS/390 Data Sets for the Oracle Database" for detailed information on initializing an OSDI subsystem and configuring an OSDI database service.

Step 2: Shutdown Your Old Oracle Instance Cleanly

Perform an Oracle SHUTDOWN NORMAL (or IMMEDIATE) and allow it to complete. If it cannot complete because user sessions remain connected, do a SHUTDOWN ABORT or forcibly terminate the MPM subsystem or OSDI service, then do a fresh STARTUP followed by SHUTDOWN NORMAL. It is important for SHUTDOWN NORMAL to complete, ensuring that no outstanding redo data remains in the database.

If your old instance is OSDI, stop the associated service to cause its address space(s) to terminate.

Step 3: Back Up the Database

This step is recommended, though not mandatory, for an upgrade from Oracle 8i, Release 8.1.7.3 and Oracle9*i*, R1. It provides an extra measure of assurance that no data can be lost in the upgrade process.

If you are migrating from Oracle7 or upgrading Oracle8, Release 8.0.6, a cold backup is mandatory because there is no downgrade procedure. For an Oracle7 migration, the backup can be taken now, before running the MIG utility, or afterward. There are slightly different considerations for this backup depending on when the backup is done.

In particular, if you restore a backup taken after MIG is run (in order to fall back to Oracle7), you must rerun the Oracle7 catalog scripts to build the Oracle dictionary views, which MIG drops. Refer to Appendix D of *Oracle9i Database Migration: Release 2 (9.2)* for details on backing up before or after running MIG.

If you are upgrading Oracle8, Release 8.0.6, the backup must be taken now, before starting up Oracle9*i*, R2.

The backup should be of the entire database including control files, online log files, and data files. The backup can be taken with a fast physical data mover such as IBM's DF/DSS. If you use DF/DSS, refer to the following example JCL:

```
//BACKUP EXEC PGM=ADRDSSU
//SYSPRINT DD SYSOUT=*
//DUMPDS DD DSN=ORACLE.ORA1BK.FULL,
// DISP=(NEW,CATLG,DELETE,)
// UNIT=3480,
// VOL=(,,,99,SER=(vol001,vol002,vol003....)),
// LABEL=(1,SL,EXPDT=98000)
//SYSIN DD *
DUMP DATASET(INCLUDE(ORACLE.ORA1.**) ) -
OUTDD(DUMPDS)
/*
```

This example assumes that all of the Oracle7 database files begin with"ORACLE.ORA1".

Step 4: Oracle7 Migration Steps (Oracle7 Migration Only)

Do this major step only if migrating from Oracle7 to Oracle9*i*, R2. Otherwise, skip ahead to "Step 5: Prepare To Start the New Oracle Release".

The Oracle migrate (MIG) utility from the new release is run against your database to extract data from your control files and write it to a file called a "convert" file which is read later when starting up the new Oracle release. Refer to Appendix D of *Oracle9i Database Migration: Release 2 (9.2)* for more information on the MIG utility and its parameters. Here we cover only the OS/390-specific considerations in running MIG.

Step 4.1: Change Oracle7 Control File(s) to VSAM LDS

If your Oracle7 control file(s) are VSAM ESDS clusters, you must change them to VSAM Linear Data Set (LDS) clusters before running the MIG utility. You can determine the cluster type of your control files by doing an IDCAMS LISTCAT ALL on them. If NONINDEXED appears in the data component ATTRIBUTES section of the catalog listing, the cluster is an ESDS and must be changed to an LDS. (The access method MIG uses to read the control files requires this.)

The change to LDS is an ICF catalog update operation done with IDCAMS; it does not move or modify any of the data in the control file. This operation cannot be reversed--you can't ALTER a cluster back to an ESDS--but having them be LDS does not affect your ability to fallback to MPM.

Replace the names in the following example with the correct name(s) of your control file(s).

```
//AMS   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 ALTER ORACLE.ORA1.CONTROL1 TYPE(LINEAR)
 ALTER ORACLE.ORA1.CONTROL2 TYPE(LINEAR)
/*
```

Step 4.2: Preallocate the Convert File

The convert file is a VSAM LDS that will be written by the MIG utility. You can preallocate it with an IDCAMS DEFINE as illustrated here or you can use JCL allocation in the MIG job itself. Size requirements for the convert file are difficult to estimate; making it at least twice the size of your control file and including a secondary allocation quantity are recommended. The file can be deleted after migration is complete.

This example is designed to be re-run: it deletes any existing convert file before doing the DEFINE. Choose a data set name for the convert file that meets your installation's naming requirements and replace 'volser' with a suitable DASD volume serial, or replace the VOLUMES parameter with SMS parameters as appropriate for your installation.

```
//AMS   EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN DD *
 DELETE (ORACLE.ORA1.CNVFILE)
 SET MAXCC=0
 DEFINE CLUSTER( LINEAR -
  NAME(ORACLE.ORA1.CNVFILE) -
  VOLUMES(volser ) -
  TRACKS(100 10) -
  SHR(3 3) )
/*
```

Step 4.3: Prepare Control File(s)

Migration from Oracle7 to Oracle9*i*, R2 replaces your control files: when you first run the new release, an ALTER DATABASE CONVERT statement causes the convert file to be read and new control file data to be written. These will be new files--it is not possible to reuse (overwrite) your Oracle7 control files. The control files used by Oracle9*i*, R2 are substantially larger (for an equivalent database) than those used by Oracle7. You can reuse the same data set names for your control files if you delete or rename your Oracle7 control files prior to starting up the new release.

There is no option to preallocate the new control files in a migration; the server creates them with a dynamic call to the IBM IDCAMS utility. This means the server ORA$FPS file parameters for the DBCT (control file) group are critical for a successful migration.

Make sure that either unit and volume serial or SMS classes are supplied for the DBCT group so that allocation of the control files will succeed. Refer to Chapter 4, "Defining OS/390 Data Sets for the Oracle Database" for details on ORA$FPS parameters.

The name(s) of your control files are specified using the CONTROL_FILES init.ora parameter. Make sure the init.ora parameters for the new release contain the data set name(s) you want to use. If you were using "/DD/" notation for these in Oracle7 you must change to data set names in the new release. The old (MPM) data set name syntax is accepted for compatibility, but Oracle recommends using the new LE-style syntax as in the following example:

```
CONTROL_FILES = "//ORACLE.ORA1.CONTROL1","//ORACLE.ORA1.CONTROL2"
```

If you are using conventional IBM-compatible DASD devices, where a given volume serial equates to a single real device, Oracle recommends using at least two control files and placing them on separate devices and, if possible, separate channel and control unit paths. This separation can be difficult or impossible to arrange when the server is generating the IDCAMS DEFINE CLUSTER commands. If your control files end up on the same device, you can move one of them to another volume after the migration using IDCAMS EXPORT/IMPORT or REPRO. You must shut down the Oracle instance to do this.

Step 4.4: Prepare JCL for the MIG Utility

Refer to Appendix D of *Oracle9i Database Migration: Release 2 (9.2)* for general documentation on the Oracle MIG utility. There are several aspects of the MIG utility that are unique to OS/390, and those are covered here.

The MIG utility comes from the new Oracle release (the one to which you are migrating) but it connects to your Oracle7 instance. Because the OSDI cross-memory protocol is not compatible with MPM, MIG must use Oracle Net TCP/IP to connect to the MPM instance. This means your Oracle7 instance must be able to accept a TCP/IP connection. Both the Oracle7 instance and the TNS subsystem it uses must be running, and the MPMTNS subtask initialized, when you run the MIG job. Because Oracle9*i*, R2 clients do not use the network service for outbound connections, the OSDI network service is not required when you run MIG.

In the MIG utility JCL, supply a TNSNAMES file containing an address entry for your Oracle7 instance. The entry will have a distinct TCP/IP port number since that is how MPM/TNS distinguished target servers. The port number is that which is specified in the MPMTNS parameter file in the MPM Oracle7 instance. You will need to supply an environment variable, TWO_TASK, with the name of the TNSNAMES entry when you run MIG.

The MIG utility also requires the password for Oracle userid SYS be supplied via environment variable SYSPASSW. This, and the TWO_TASK environment variable are both supplied via the ORA$ENV DD statement.

Other required DD statements for the MIG utility are:

ORA$LIB - specifies the MESG data set from Oracle9*i*, R2

INITORA - specifies your Oracle7 init.ora file parameters; this is the default unless overridden with the MIG PFILE parameter

MIGRATE - specifies the MIGRATE SQL script from the new release (normally this is the MIGRATE member of the .SQL PDS)

CNVFILE - specifies the convert file to be written

SYSIN - can be empty or DUMMY

In addition to the above, you must include with the MIG utility *all* Oracle database file DD statements that you are currently using with your Oracle7 instance. This includes any control file(s) specified in init.ora with "/DD/" prefixes and any Oracle log and tablespace files that were added to the server using "/DD/" notation. Refer to the JCL used by your MPM Oracle7 instance to see what database file DD statements are supplied. These can be copied verbatim into the MIG job JCL. If all of your control, log, and tablespace files were specified to your MPM

Oracle7 database using data set names instead of DD names, no additional DD statements are required by MIG.

Note: Only the Oracle7 control files are actually opened and accessed by the MIG utility. The DD statements for Oracle log and tablespace files are used to determine the underlying data set names, which are required by the new Oracle instance.

In our example MIG JCL, we have assumed that two control files and three log files (and no tablespace files) are known to the Oracle7 server as "/DD/" files and that the current password for Oracle user SYS is CHANGE_ON_INSTALL.

```
//MIG   EXEC PGM=MIG,REGION=0M,
//    PARM='DBNAME=ORA1 MULTIPLIER=60 SPOOL=DD:SPOOL'
//STEPLIB  DD DISP=SHR,DSN=ORACLE.V920.CMDLOAD
//         DD DISP=SHR,DSN=SYS1.SCEERUN   IBM LE Runtime
//ORA$LIB  DD DISP=SHR,DSN=ORACLE.V920.MESG
//INITORA  DD DISP=SHR,DSN=ORAN.ORAV.PARMLIB(INITORA1)
//MIGRATE  DD DISP=SHR,DSN=ORACLE.V920.SQL(MIGRATE)
//CNVFILE  DD DISP=OLD,DSN=ORACLE.ORA1.CNVFILE
//SPOOL    DD SYSOUT=*
//*** The following were copied from the Oracle7 (MPM) instance JCL:
//CONTROL1 DD DISP=SHR,DSN=ORACLE.ORA1.CONTROL1
//CONTROL2 DD DISP=SHR,DSN=ORACLE.ORA1.CONTROL2
//LOG1     DD DISP=SHR,DSN=ORACLE.ORA1.LOG1
//LOG2     DD DISP=SHR,DSN=ORACLE.ORA1.LOG2
//LOG3     DD DISP=SHR,DSN=ORACLE.ORA1.LOG3
//*** End of DDs from Oracle7 instance JCL.
//SYSIN    DD DUMMY
//TNSNAMES DD *
ORA7MPM1 =
 (DESCRIPTION =
  (ADDRESS = (PROTOCOL=TCP) (HOST=localhost) (PORT=1568) ))
/*
//ORA$ENV  DD *
TWO_TASK=ORA7MPM1
SYSPASSW="CHANGE_ON_INSTALL"
/*
```

Step 4.5: Prepare Your Oracle7 Database for the MIG Utility

The OS/390 version of MIG operates differently from that of other Oracle platforms: it does not start up or shut down the Oracle7 database instance. Those operations are done separately on OS/390.

Start up your Oracle7 instance using STARTUP RESTRICT to ensure that no normal user sessions can be started. Ensure that the required TNS subsystem and MPMTNS task are initialized so that MIG can make a TCP/IP connection to the instance. Perform checks of various database components as discussed in *Oracle9i Database Migration: Release 2 (9.2)*. If you want, you can run MIG with 'CHECK_ONLY=true' to check for sufficient space in your SYSTEM tablespace.

If you take any actions at this point to change the Oracle7 database in preparation for migration, such as enlarging the SYSTEM tablespace or cleaning up pending transactions found in DBA_2PC_PENDING, you must perform a clean SHUTDOWN (NORMAL or IMMEDIATE) and another STARTUP RESTRICT of the Oracle7 instance before running MIG.

Step 4.6: Run the MIG Utility

Run the MIG job with JCL as discussed in "Step 4.4: Prepare JCL for the MIG Utility". The utility does not take long to run, even with large databases. Examine the SPOOL output from the utility carefully for error messages. If there are problems you must correct them, shut down the Oracle7 instance cleanly, start it back up with STARTUP RESTRICT, and rerun MIG before proceeding to the next step.

Step 4.7: Shut Down Oracle7

Bring down your Oracle7 instance and MPM address space with SHUTDOWN NORMAL or IMMEDIATE. From this point you must not re-open the database with Oracle7 before bringing up Oracle9*i*, R2. If you do start up Oracle7 with this database you must go back and rerun the MIG utility, creating a new convert file, before trying to bring up Oracle9*i*, R2.

Step 4.8: Back Up the Database

If you opted not to back up the database in "Step 3: Back Up the Database" (before running MIG), you must do it now.

Step 4.9: Make the Convert File Available to the New Release

Modify the JCL procedure you created for the database service in "Step 1: Configure OSDI Database Service" to include a CNVFILE DD statement that specifies the convert file written by MIG. It should look exactly the same as the DD statement included with the MIG utility, e.g.

```
//CNVFILE DD DISP=SHR,DSN=ORACLE.ORA1.CNVFILE
```

This DD will be used only during the migration startup of the new release and should be removed from the procedure afterward.

Step 5: Prepare To Start the New Oracle Release

At this point you are about to do a STARTUP of the new release of Oracle. If your old release was an MPM/TNS release, you configured the OSDI subsystem and database service in"Step 1: Configure OSDI Database Service" and all of the things it needs (JCL procedure, parameters, etc.) are ready. You will start the database service for the first time in the next step.

If your old release was an OSDI release and the associated database service is still active, stop it with an OSDI STOP command or an OS/390 STOP (P) command. After the service terminates, you must modify the service to use the new release code. There are two ways to do this:

1. Modify the existing JCL procedure for the service to specify the new release library in STEPLIB.
2. Create a new JCL procedure for the service with a different name from the old procedure, and use the OSDI ALTER SERVICE command to switch the service to the new procedure. If you take this approach, make sure that the new procedure will be assigned the correct OS/390 userid when started. If in doubt, talk to your system security administrator.

It also is possible to define a new OSDI service (specifying a new JCL procedure) and use this to bring up the new release. However, the new service can't have the same SID as the old (assuming both are on the same OS/390 system) because SIDs must be unique. In most cases you will want the new release to run using the same SID as the old.

Make sure that your init.ora parameter file for the new release is ready. If you are upgrading from Oracle8*i*, Release 8.1.7 or Oracle9*i*, make sure you have specified the COMPATIBLE parameter and replaced any "/DD/" CONTROL_FILES values with control file data set names. If you are migrating from Oracle7, make sure that your CONTROL_FILES parameter specifies the new data set names for your

Oracle9*i*, R2 control files. Refer to the section"RDBMS Parameter Considerations" on page 18-12.

Step 6: Start the New Oracle Release

First, start the OSDI database service with an OSDI START command. Make sure the service initializes successfully, signified by message MIR0002I in the system log.

Next, do an Oracle STARTUP using SQL*Plus from the new release. You can do this using a batch job, from TSO, or from USS. Our examples use batch jobs and identify the target instance with an ORA@sid DD statement. With OSDI, this DD supplies an OSDI SID rather than an MPM subsystem name. However you choose to run SQL*Plus, if you have configured OSDI security features the associated OS/390 userid must have the required authorizations: it must be authorized to bind to the database service and it must be able to CONNECT using AS SYSDBA.

Specify RESTRICT on the STARTUP command, if desired, to prevent normal client access to the database while you are completing the migration or upgrade.

Other details of this startup depend on whether you are migrating from Oracle7 or upgrading a later release. In the migration case, the STARTUP must specify NOMOUNT and is followed by two ALTER DATABASE statements that cause the convert file to be read and processed, populating the new control files and completing the migration:

```
//PLUS    EXEC PGM=SQLPLUS,REGION=0M,PARM='/nolog'
//STEPLIB  DD DISP=SHR,DSN=ORACLE.V920.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORACLE.V920.MESG
//INITORA  DD DISP=SHR,DSN=ORACLE.V920.PARMLIB(INITORA1)
//ORA@ORA1 DD DUMMY  <-- OSDI SID is 'ORA1'
//SYSIN    DD *
CONNECT / AS SYSDBA
STARTUP PFILE='//DD:INITORA' RESTRICT NOMOUNT
ALTER DATABASE CONVERT;
ALTER DATABASE OPEN RESETLOGS MIGRATE;
/*
```

In the upgrade case, use STARTUP with the MIGRATE option, which mounts and opens the database:

```
//PLUS    EXEC PGM=SQLPLUS,REGION=0M,PARM='/nolog'
//STEPLIB  DD DISP=SHR,DSN=ORACLE.V920.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORACLE.V920.MESG
//INITORA  DD DISP=SHR,DSN=ORACLE.V920.PARMLIB(INITORA1)
```

```
//ORA@ORA1 DD DUMMY  <-- OSDI SID is 'ORA1'
//SYSIN    DD *
CONNECT / AS SYSDBA
STARTUP PFILE='//DD:INITORA' RESTRICT MIGRATE
/*
```

In either case, check the output from SQL*Plus to make sure that the STARTUP (and ALTER statements, if migrating) ran successfully. Normally no messages are issued to the system log during these operations, even if there are errors. If there are error indications in the SQL*Plus output, it may be helpful to examine the instance alert log. Unless you supplied a SYSPRINT DD statement in the database service JCL procedure, the alert log is a SYSOUT data set that can be browsed with a tool like TSO SDSF.

Errors during STARTUP or ALTER DATABASE must be diagnosed and corrected before proceeding.

Step 7: Run Upgrade Scripts

Whether you are migrating from Oracle7 or upgrading a later release of Oracle, you now need to run SQL scripts to upgrade the Oracle dictionary structures to the new release. The scripts are provided in the .SQL data set installed with the new release; which one you run depends on the Oracle release you are migrating or upgrading from. Refer to *Oracle9i Database Migration: Release 2 (9.2)* to determine which script(s) you must run.

Our example is JCL for a batch SQL*Plus job that executes the upgrade script to complete a migration from Oracle7. This job uses the OS/390 FNA feature provided with most Oracle tools and utilities to allow the various SQL scripts and sub-scripts to be read as members of a PDS, which is how they are installed. In the Oracle7 migration case, although you are migrating from Oracle7, Release 7.3.3.6, the script named U0703040 is run. (There is no U0703030 script.) If you are upgrading from a later Oracle release, specify the exact script(s) for your old release as discussed in the Migration manual--do not run the U0703040 script. In all cases, the release-specific upgrade script is followed by the CMPDBMIG script as shown below.

```
//PLUS    EXEC PGM=SQLPLUS,REGION=0M,PARM='/nolog'
//STEPLIB  DD DISP=SHR,DSN=ORACLE.V920.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORACLE.V920.MESG
//SQL      DD DISP=SHR,DSN=ORACLE.V920.SQL
//ORA@ORA1 DD DUMMY  <-- OSDI SID is 'ORA1'
//ORA$FNA  DD *
```

```
 FSA (FTYPE(SQL) FNAME('//DD:SQL(+)'))
 FSA (FTYPE(PLB) FNAME('//DD:SQL(+)'))
/*
//SYSIN    DD *
CONNECT / AS SYSDBA
@U0703040
@CMPDBMIG
/*
```

Examine the SQL*Plus output from these scripts and the Oracle instance alert log for significant error indications. Certain DROP statements in the scripts may produce "object not found" errors which are not considered significant. Other errors, particularly ORA-0060x (internal) errors, must be resolved before proceeding.

Step 8: Shut Down and Restart the Instance

This is required at this point to perform internal housekeeping activities. On conclusion of the STARTUP, your database is migrated or upgraded and is usable. Do this STARTUP without RESTRICT, if desired, to enable normal client access.

If you migrated from Oracle7, you can edit the database service JCL procedure to remove the CNVFILE DD statement. The convert file data set will remain allocated to the service address space, however, until you stop and restart the service.

Our example is a SQL*Plus batch job:

```
//PLUS    EXEC PGM=SQLPLUS,REGION=0M,PARM='/nolog'
//STEPLIB  DD DISP=SHR,DSN=ORACLE.V920.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORACLE.V920.MESG
//INITORA  DD DISP=SHR,DSN=ORACLE.V920.PARMLIB(INITORA1)
//ORA@ORA1 DD DUMMY  <-- OSDI SID is 'ORA1'
//SYSIN    DD *
CONNECT / AS SYSDBA
SHUTDOWN IMMEDIATE
STARTUP PFILE='//DD:INITORA'
/*
```

Step 9: Recompile Database Procedures

Some number of stored procedures in the database will have been invalidated during migration or upgrade due to changes in underlying database objects. Although these will recompile dynamically as needed, Oracle recommends forcing the recompiles now. This involves running a supplied script and using FNA, as in the dictionary upgrade step:

```
//PLUS    EXEC PGM=SQLPLUS,REGION=0M,PARM='/nolog'
//STEPLIB  DD DISP=SHR,DSN=ORACLE.V920.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORACLE.V920.MESG
//SQL      DD DISP=SHR,DSN=ORACLE.V920.SQL
//ORA@ORA1 DD DUMMY  <-- OSDI SID is 'ORA1'
//ORA$FNA  DD *
 FSA (FTYPE(SQL) FNAME('//DD:SQL(+)'))
 FSA (FTYPE(PLB) FNAME('//DD:SQL(+)'))
/*
//SYSIN    DD *
CONNECT / AS SYSDBA
@UTLRP
/*
```

For additional information on this script refer to *Oracle9i Database Migration: Release 2 (9.2)*.

Step 10: Run CFUTIL To Modify Control Files

Do this step only if you upgraded from an Oracle8 or Oracle8*i* MPM/TNS implementation to OSDI. This step changes the syntax of all file names stored in the control files from MPM "/DSN/" to OSDI LE-style. (If you migrated from Oracle7, this change was made for you by the MIG utility.) Although the database will run without this change, it must be done to enable certain file integrity checks in the server. Refer to "CFUTIL Parameter Syntax for Control Files" on page 18-37 for details.

Note: You must shut down the Oracle instance before running CFUTIL. You can start it back up immediately afterward.

Archive Logs and Recovery

This discussion pertains only to upgrades of Oracle8, Release 8.0 or the MPM/TNS implementation of Oracle8*i* to OSDI. If you encounter a database recovery situation under OSDI and you did not take a complete database backup during the upgrade, then you may have archive logs created under MPM that must be processed under OSDI. OSDI requires archive logs to be VSAM LDS data sets. For that reason, the DSORG=PS archive logs that were created under MPM must be converted in order to be usable. The conversion is done using the CFUTIL program documented in section "CFUTIL (Convert File Utility) Reference" on page 18-35. Each archive log that is called for during recovery must be converted before being specified to Oracle database server.

Fallback and Downgrade Considerations

Problems in OSDI or in Oracle9*i*, R2 might force you to return to the MPM/Oracle release that you were running. If that release was Oracle7, Release 7.3.3.6 or Oracle8, Release 8.0.6, you must restore the backup of the database that was taken during migration or upgrade. If your old database was Oracle8*i*, Release 8.1.7.3, then you can fall back to that older release without restoring the database, provided you have used the COMPATIBLE database parameter to prevent writing new data structures into your database.

To fall back, first shut down the Oracle9*i*, R2 instance (if possible) and stop the OSDI service, waiting for all service address spaces to terminate. When this is complete, if you are falling back to the MPM implementation of Oracle8*i*, run CFUTIL to convert database file names in the control file back to their MPM form. For more information, refer to "CFUTIL (Convert File Utility) Reference" on page 18-35. When CFUTIL has completed converting your control file, or files, to MPM format, you can start the MPM address space and warmstart the instance. Use STARTUP RESTRICT, because the database is not yet ready to use.

After starting the old release, you must run downgrade scripts to return Oracle dictionary objects to their older state. Refer to *Oracle9i Database Migration: Release 2 (9.2)* for information on the sequence of scripts to run when downgrading.

If you have created archive logs or have added tablespace datafiles or additional online log files to your database while running under OSDI, then a patch to your MPM/Oracle kernel is required in order for those files to be usable under MPM. Contact Oracle Support Services if you need this patch.

CFUTIL (Convert File Utility) Reference

OSDI requires that certain files be in a different format from MPM. When converting an existing MPM-based Oracle database server to an OSDI database server, or during fallback from an database server to an MPM-based Oracle database server, CFUTIL is needed to convert files to the corresponding format.

One of the functions of the CFUTIL program is to convert database file names (that are stored in Oracle8 MPM-based control files) to the form used by OSDI. This step is not required when migrating an Oracle7 instance to this release of OSDI database server because control files are created and formatted during the migration step. This utility can also convert OSDI-format control files back to MPM-format if circumstances require you to fall back to the MPM implementation. Conversion is "in-place" in either direction. If you want to preserve a copy of a control file before conversion, you can use IDCAMS EXPORT or other system utilities such as DFDSS to do so. The conversion is relatively non-invasive and easily reversed, so such backups are usually unnecessary.

Although Oracle9*i* under OSDI is capable of running with unconverted MPM-format control files, the presence of old-format file names in the control file creates an integrity exposure because the Oracle database server may fail to recognize a second attempt to add a file that is already part of the database. Oracle recommends that you convert your control files.

Note: In a fallback situation (from OSDI back to MPM), Oracle8 under MPM is not capable of running with OSDI-format control files. Reverse conversion is mandatory during fallback.

CFUTIL can only process VSAM linear data set (LDS) clusters. If your control file(s) are VSAM ESDS clusters you must change them to LDS before running CFUTIL. This is done using the OS/390 IDCAMS utility ALTER command or by issuing ALTER directly from TSO. The ALTER command does not move or change data in the affected files; it only changes the associated ICF catalog entry. Once a cluster is altered from ESDS to LDS, it cannot be altered back. If you want to preserve the ESDS form of your control files, you must back them up with a facility that also backs up the catalog entry, such as IDCAMS EXPORT. This is not necessary, however, for fallback to Oracle7.

To convert an ESDS control file to LDS form, log on to TSO and issue the ALTER command specifying the cluster name and TYPE(LINEAR). You can also execute the IDCAMS utility in batch, as shown in the following example:

```
//MIGLDS   JOB (0000,OR),'ORACLE INSTALL'
//AMS      EXEC PGM=IDCAMS
//SYSPRINT DD SYSOUT=*
//SYSIN    DD *
  ALTER ORA3B.ORAV70A.CONTROL1 TYPE(LINEAR)
  ALTER ORA3B.ORAV70A.CONTROL2 TYPE(LINEAR)
/*
//
```

Run CFUTIL against your MPM-format control files after a successful normal shutdown of the Oracle (MPM) system and before you attempt a warm start of Oracle9*i* for OS/390. If your instance uses multiple copies of the control file, then you must convert all copies. The control files must first be converted to VSAM Linear Data Sets, as discussed above. After the files are converted, a warm start can be performed under OSDI using the converted control files. If possible, the same procedure (in other words, a normal shutdown followed by the CFUTIL run) should be used when moving from OSDI back to MPM. If the circumstances that cause you to fall back also prevent a normal shutdown under OSDI, then stop the associated service and make sure that all of its address spaces have terminated before attempting the conversion.

CFUTIL runs in three environments:

1. OS/390 Unix System Services (USS, formerly called OpenEdition)
2. TSO (using CALL)
3. Batch job

When CFUTIL is invoked as a command under USS, command parameters are specified on the command line immediately after the command name. When CFUTIL is invoked using TSO CALL, parameters are specified inside a pair of apostrophes as the second positional parameter of CALL. When CFUTIL is run as a batch job, parameters are specified in the PARM field of the EXEC statement. In all cases, the parameter syntax is the same.

CFUTIL Parameter Syntax for Control Files

```
CVTFOROSDI | CVTFORMPM
control-file1 [control-file2 [... control-filen]]
```

where `control-file1/2/n` specifies the data set name, or names, of control files to convert.

`CVTFOROSDI` specifies conversion of control files from MPM format to OSDI.

`CVTFORMPM` specifies conversion of control files from OSDI format to MPM.

Notes: Before running CFUTIL, you must convert the input control files to VSAM linear data set (LDS) files. Refer to the example under "CFUTIL (Convert File Utility) Reference" on page 18-35 for further information on converting to VSAM LDS files.

To fall back to MPM using CVTFORMPM, you must be running MPM/Oracle, Release 8.1.7.3 or higher.

If a warmstart is done with OSDI and the COMPATIBLE parameter is set higher than 8.1.7.3, CVTFORMPM cannot be used.

The OS/390 userid in effect for the USS, TSO, or batch environment must have read and write access to the control file data sets that are to be updated.

Under USS, the number of control files that can be listed is unlimited. With TSO CALL and batch, the conversion type and data set names are specified using an OS/390 PARM and are therefore limited to 100 characters total, including the type keyword and the data set names. This limits the total number of data set names to whatever will fit in the 100 characters after the type.

The letter case of the command and data set names is not significant. All data set names are converted to uppercase characters before being used.

If the utility fails or is canceled while it is running, then it can safely be rerun until it completes successfully. Any partial changes made in earlier attempts are accepted, and the utility runs from that point to completion. The utility can even be run again, and if the control files are already in the correct format, then no changes will be made.

Examples

USS:

```
cfutil cvtforosdi oracle.orax.control1 oracle.orax.control2
```

TSO CALL:

```
CALL 'ORAN.ORAV.CMDLOAD(CFUTIL)' -
   'CVTFOROSDI ORACLE.ORAX.CONTROL1 ORACLE.ORAX.CONTROL2'
```

Batch:

```
//CFUEXEC EXEC PGM=CFUTIL,
// PARM='CVTFOROSDI ORACLE.ORAX.CONTROL1 ORACLE.ORAX.CONTROL2'
//STEPLIB  DD DISP=SHR,DSN=ORAN.ORAV.CMDLOAD
//ORA$LIB  DD DISP=SHR,DSN=ORAN.ORAV.MESG
//SYSPRINT DD SYSOUT=*          STDOUT OUTPUT
//SYSOUT   DD SYSOUT=*          STDERR OUTPUT
```

Note: Because the PARM= field for this utility can easily be longer than a line, you may need to split the parameter (up to 100 characters) across two or more lines. To do this, start the PARM= on the first line with the opening apostrophe followed by as much as will fit through record position 71. In position 72, put any non-blank character. On the records that follow, put "//" in positions one and two followed by 13 blanks, starting the continuation in position 16.

The following example shows a long, continued EXEC parameter with record position indicators:

```
----5---10---15---20---25---30---35---40---45---50---55---60---65---70--
//CFUEXEC EXEC PGM=CFUTIL,PARM='CVTFOROSDI ORACLE.ORAX.CONTROL1 ORACLE.*
//             ORAX.CONTROL2'
----5---10---15---20---25---30---35---40---45---50---55---60---65---70--
```

CFUTIL for Archive Log Files

Another usage of CFUTIL is to convert archived logs that are generated under MPM-based Oracle database server to the VSAM LDS format required by OSDI database server before they can be accessed in a recovery situation. This is an unlikely situation, but the utility is provided for the situation as a precaution. The syntax is as follows:

```
UFUTIL CVTARCLOG   <source_dsn>   <target_dsn>
```

The *source_dsn* is an MPM-style physical sequential archive log, and the *target_dsn* is a pre-allocated VSAM LDS.

A
OSDI Subsystem Command Reference

OSDI provides a set of system commands for defining and controlling instances of Oracle products. This appendix is the primary reference for all OSDI commands.

The topics in this appendix include:

OSDI Command Reference

OSDI provides a set of system commands for defining and controlling instances of Oracle products. This appendix is the primary reference for all OSDI commands.

Command Types and Processing

Commands are broadly divided into two groups: definition and operating.

1. Definition commands are used to create and manipulate data structures that describe service groups and services. These commands commonly appear in the subsystem configuration file.
2. Operating commands are used to manage execution of services.

Three mechanisms exist for issuing OSDI commands:

1. Subsystem configuration file - processed during subsystem initialization.
2. OS/390 system operator command interfaces-system consoles, TSO SDSF, SYS1.PARMLIB (COMMND*xx*), Netview, and others.
3. OSDI program interface - used internally by Oracle products.

When an OSDI command is issued using an OS/390 system operator command interface, the target subsystem is specified by using the command prefix associated with the subsystem. When the program interface is used to issue an OSDI command, the target is identified by its subsystem name rather than a command prefix. Commands in the configuration file always apply to the subsystem (service group) being initialized, and they must omit the prefix.

Commands generally result in synchronous response messages ranging from simple acknowledgment to multiline displays. Responses to commands that are issued via system operator facilities normally are directed to the issuing console. Various operating commands can result in subsequent, asynchronous messages. These messages are not necessarily directed to the console or session that issued the original command.

System Symbols in Commands

In order to meet the requirement that the particulars of a service (that runs on multiple systems in a sysplex) can be tailored by system, OS/390 system symbols (alphanumeric names prefixed with the ampersand character, "&") can be used in the specification of certain OSDI command parameters. These command parameters resolve to system-specific or IPL-specific values set by OS/390 or by the installation. If a command parameter can include system symbols, this capability is noted in the parameter description.

Definition Commands

Definition commands are used to create, modify, and display the data structures of the service group. An initial set of commands in the configuration file directs the building of these structures during subsystem initialization. Subsequent definition commands can be used to add new service definitions, modify existing definitions, and so forth. The overall data structure persists for the life of the IPL.

The definition commands operate only on data structures; they do not directly affect the operation of services.

Three definition commands are supported:

- DEFINE - Create a logical structure.
- ALTER - Modify the definition of an existing logical structure.
- SHOW - Display contents of an existing logical structure.

Note: **SHOW** deals with definition data only and is distinct from the operating command, **DISPLAY**. Refer to "SHOW" on page A-5, "SHOW" on page A-11, and "DISPLAY" on page A-13.

Structures

The definition commands operate on the following structures:

- SERVICEGROUP - The primary structure of the subsystem.
- SERVICE - A structure representing an instance of an Oracle product.

The various parts of these structures are generally referred to as attributes. Definition commands use keywords to identify attributes being set or modified.

Service Group Definition Commands

DEFINE

DEFINE SERVICEGROUP must be the first command issued to a newly-initialized subsystem; normally it appears in the configuration file just after the bootstrap (INIT) record. DEFINE SERVICEGROUP must be processed successfully in order for any other OSDI commands or functions to be usable.

The command structure for defining a service group is shown in the following example:

```
DEFINE SERVICEGROUP
   [ SSID( string ) ]
   [ DESCRIPTION( string ) ]
   [ MODE( LOCAL | SYSPLEX | *SYS ) ]
   [ SYSTEMS( sysname [ sysname... ] | *ALL ) ]
```

Define Parameters

SSID	Specifies the 1-character to 4-character OS/390 subsystem name associated with the service group. If coded, then it must match the subsystem identifier specified in the IEFSSN*xx* parmlib member or the SETSSI operator command. This parameter defaults to the correct value. It is therefore coded only to confirm that the correct configuration file is in use. *String* cannot contain system symbols.
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service group. This can be used to supply installation-specific identification for the service group. The default value is 'Service Group *ssn*' where *ssn* is the subsystem name. *String* can contain system symbols but should not contain non-printable characters or control characters.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

ALTER

ALTER SERVICEGROUP is used to modify attributes of a service group. It can be included in the configuration file and it can be issued after initialization. Not all attributes can be altered. The subsystem ID, for example, is constant for the life of the IPL.

The command structure for altering a service group is shown in the following example:

```
ALTER SERVICEGROUP
    [ DESCRIPTION( string ) ]
    [ MODE( LOCAL | SYSPLEX ) ]
    [ SYSTEMS( sysname [ sysname... ] | *ALL ) ]
```

Alter Parameters

DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service group. This can be used to supply installation-specific identification for the service group. *String* can contain system symbols but should not contain non-printable or control characters.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

SHOW

SHOW SERVICEGROUP is used to display the current definition of the service group. It can be included in the configuration file and it can be issued after initialization. The command for displaying a service group definition is shown in the following example:

```
SHOW SERVICEGROUP
   [ LONG ]
```

Show Parameters

LONG	Specifies that the name, type, and description of each service in the service group be included in the display.

Service Definition Commands

DEFINE

DEFINE SERVICE is used to create a structure for executing an installed Oracle product. It can be included in the configuration file and it can be issued after initialization. Once a service is defined, it can be started, stopped, etc. using operating commands.

The command structure for defining a service is shown in the following example:

```
DEFINE SERVICE
     name
     TYPE( string )
     PROC( string )
   [ DESCRIPTION( string ) ]
   [ PARM( string ) ]
   [ MAXAS( number ) ]
   [ SID( string ) ]
   [ JOBNAME( string ) ]
   [ JOBACCT( string ) ]
   [ MODE( SYSPLEX | LOCAL ) ]
   [ SYSTEMS( sysname [ sysname... ] | *SVG ) ]
```

Define Parameters

name	Specifies the name of the service. The name is used in other commands that operate on the service or its definition and by program functions that interact with the service. It must be 1 character to 8 characters long, must consist of upper case alphabetic, numeric, and/or national characters, and must begin with an alphabetic character. It must be unique within the service group. *Name* cannot contain system symbols. **Note:** Unless you specify JOBNAME, *name* is used as the job identifier when the service is started. In this case, *name* must not be the same as any subsystem name in your system.

TYPE	Specifies the Oracle product being configured. *`String`* is a 1-character to 4-character alphabetic and/or numeric identifier that designates an installed Oracle for OS/390 product managed under this architecture. Current supported values are ORA (Oracle database) and NET (Oracle Net). *`String`* cannot contain system symbols.
PROC	Specifies the member name in a system JCL procedure library used to start an address space for the service. Usually this is a procedure that is created during installation of the associated Oracle for OS/390 product. *`String`* cannot contain system symbols.
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service. This can be used to supply installation-specific identification for the service. The default value is '`Service` *`svc`* `Type` *`type`*', where *`svc`* is the service name and *`type`* is the type. *`String`* can contain system symbols but should not contain non-printable or control characters.
PARM	Specifies a parameter string passed to the service when an address space is started. *`String`* can be from 0 character to 64 characters. If it contains characters other than alphabetic, numeric, and national characters, then it must be enclosed in single apostrophes. To indicate a zero-length (empty) parameter, specify `PARM('')`. Content requirements for this string depend on the service type and are discussed in Chapter 3, "Configuring a Database Service and Creating a New Database" and Chapter 10, "Oracle Net". The default is a null string. The value specified can contain system symbols.
MAXAS	Specifies the maximum number of address spaces that can be started for the service. This is meaningful only for a database (TYPE(ORA)) service. If specified for any other type, then *`number`* must be 1. For a database service, *`number`* must be between 1 and 255 inclusive. The default for this parameter is 1. The value cannot include system symbols.

SID	Specifies a unique identifier for the service that is used in client- or application-supplied service addressing. `String` is a 1-character to 8-character identifier that conforms to the same rules as those for service names. The value supplied must be unique within the OS/390 image: it cannot duplicate the SID of any other service in this or any other service group. This parameter defaults to the service name. If you accept the default, it means that the service name must be unique within the OS/390 image. **Note:** If you migrate an MPM-based database to OSDI and are supporting pre-OSDI local client applications, you probably want SID to match the subsystem name you were using with MPM.
JOBNAME	Specifies an OS/390 jobname to use when starting address spaces for the service. `String` is either a 1-character to 8-character identifier that conforms to OS/390 jobname requirements, or it is a 1-character to 5-character identifier followed by an asterisk. The asterisk form can be used with multiple address space services to provide unique jobnames; it is replaced with a 3-digit address space counter (001, 002, and so on) as each address space is started. The jobname should conform to any rules or requirements specific to your installation. This parameter has no default. If you omit it, service address spaces are started without a jobname but with the service name as the address space identifier. System symbols cannot be used in this parameter.
JOBACCT	Specifies an installation-specific string containing job accounting fields. `String` can be from 1 character to 64 characters. If it contains characters other than alphabetic, numeric, and national characters, then it must be enclosed in single apostrophes. Use this parameter if your installation requires job accounting data to be included with started tasks (STCs). If this parameter is omitted or specified as a null string (a pair of adjacent apostrophes), then no job accounting data is supplied when service address spaces are started. `String` can include system symbols.
MODE	This parameter is not yet supported.

SYSTEMS	This parameter is not yet supported.

Refer to "Examples" on page 2-6.

ALTER

ALTER SERVICE is used to modify attributes of a defined service. It can be included in the configuration file and it can be issued after initialization. The name, type, maximum address spaces, and SID of a service cannot be altered.

OSDI does not prohibit altering the definition of a running service. This enables some useful capabilities but it may also be harmful if misused. For example, changing the JCL procedure of a running multiple address space service would have unpredictable consequences when additional auxiliary address spaces are started.

The command structure for altering a service is shown in the following example:

```
ALTER SERVICE
      name
    [ DESCRIPTION( string ) ]
    [ PROC( string ) ]
    [ PARM( string ) ]
    [ MAXAS( number ) ]
    [ JOBNAME( string ) ]
    [ JOBACCT( string ) ]
    [ MODE( SYSPLEX | LOCAL ) ]
    [ SYSTEMS( sysname [ sysname... ] | *SVG ) ]
```

Alter Parameters

name	Specifies the name of the service to be altered. It must be the name of an existing service in the service group. *Name* cannot contain system symbols. **Note:** The name of a service cannot be altered.
DESCRIPTION	Specifies an arbitrary text string of up to 64 characters that appears in certain displays associated with the service. This can be used to supply installation-specific identification for the service. *String* can contain system symbols but should not contain non-printable or control characters.

PROC	Specifies the member name in a system procedure library used to start an address space for the service. Usually this is a procedure that is created during installation of the associated Oracle for OS/390 product. `String` cannot contain system symbols. **Note:** Altering PROC while a multiple address space or multiple system service is running can have unpredictable effects.
PARM	Specifies a parameter string passed to the service when an address space is started. Requirements for this string depend on the service type and are discussed in the associated Oracle for OS/390 product documentation. The value specified can contain system symbols. **Note:** Altering PARM while a multiple address space or multiple system service is running can have unpredictable effects.
MAXAS	Specifies the maximum number of address spaces that can be started for the service. This is meaningful only for a database (TYPE(ORA)) service. MAXAS is accepted on an ALTER SERVICE command only when the service is not active. If the service is active, starting, or stopping, an ALTER SERVICE command with MAXAS specified will be rejected.
JOBNAME	Specifies a jobname to use when starting service address spaces, as discussed under DEFINE SERVICE. You can nullify (remove) a service jobname specification by coding JOBNAME(''). The value specified cannot contain system symbols.
JOBACCT	Specifies job accounting data to be included when service address spaces are started, as discussed under DEFINE SERVICE. You can nullify (remove) job accounting data by coding JOBACCT(''). The value specified can contain system symbols.
MODE	This parameter is not yet supported.
SYSTEMS	This parameter is not yet supported.

SHOW

The SHOW SERVICE command is used to display the current definition of a service.

The command for displaying a service definition is shown in the following example:

```
SHOW SERVICE
     name
```

Show Parameters

name	Specifies the name of the service whose definition is to be displayed. It must be the name of an existing service in the service group. *Name* cannot contain system symbols.

Operating Commands

Operating commands manage the execution of services. They are normally issued via the OS/390 command interface, either automatically (for example, COMMND*xx* member of SYS1.PARMLIB) or by a real operator. They might also be issued through the OSDI program interface by a management component such as Oracle Enterprise Manager Agent. Operating commands are also permitted in the service group configuration file.

Available Commands

Five operating commands are provided:

- START - Start execution of a service.
- DISPLAY - Display operating status of a service.
- DRAIN - Place a service in quiescent state.
- RESUME - Restore a service to normal operation.
- STOP - Stop execution of a service (see note below).

All of the operating commands take a service name as the first positional parameter. This service name must be the name of a defined service.

Commands

START

The START command initiates execution of an address space for a specified service. For a database service, this can be the first address space (service not previously started) or an auxiliary address space (service previously started and initialized successfully but not yet at its maximum address spaces). For other types of services the service must not already be running.

The command structure for starting a service is shown in the following example:

```
START
    name
   [ PARM( string ) ]
```

START Parameters

name	`Name` specifies the name of the service to start. It must be a defined service whose current state is **inactive** or, if **active**, must not already have its maximum address spaces running.
PARM	Specifies a parameter string passed to the service when an address space is started. This overrides any PARM value established by DEFINE or ALTER SERVICE. Requirements for this string depend on the service type and are discussed in Chapter 3, "Configuring a Database Service and Creating a New Database" and Chapter 10, "Oracle Net". `String` can contain system symbols.

Note: A PARM override must not be used when starting auxiliary address spaces for a database service.

DISPLAY

The DISPLAY command displays execution status information for services. OSDI displays the current operating state of the service. If the service state is "active" or "drained", then the command is also posted to the running service for further processing at the service's discretion.

The command structure for displaying a service is shown in the following example:

```
DISPLAY
     name
   [ LONG ]
```

DISPLAY Parameters

name	Specifies the name of the service to be displayed.
LONG	Specifies that a more detailed display is desired. This provides information about each active address space of the service.

DRAIN

The DRAIN command places a running service in a quiescent state in which it no longer accepts new connection (bind) requests. Existing connections or sessions are not affected. The command is also posted to the running service for further discretionary processing. This command has no effect when the service is not running.

The command structure for draining a service is shown in the following example:

```
DRAIN
     name
```

DRAIN Parameters

name	Specifies the name of the service to be made quiescent. This must be the name of a running service whose current state is **active**.

RESUME

The RESUME command reverses the effect of a drain, allowing a service to begin accepting new connection requests. The command is also posted to the running service for further discretionary processing. This command has no effect when the service is not running.

The command structure for resuming a service is shown in the following example:

```
RESUME
    name
```

RESUME Parameters

name	Specifies the name of the service to be resumed. This must be the name of a running service whose current state is **drained**.

STOP

The STOP command requests termination of a running service. The normal mode of stop changes the service state to **stopping** and posts the stop request to the service; it is up to the service to comply, presumably after allowing current requests to complete and performing required cleanup. A **force** option causes the service address space(s) to be terminated involuntarily. The **force** form of stop requires that a normal stop be issued first. This command has no effect when the service is not running.

The command structure for stopping a service is shown in the following example:

```
STOP
    name
  [ FORCE ]
```

STOP Parameters

name	Specifies the name of the service to be stopped. This must be the name of a running service whose current state is **active**, **drained**, or (if the FORCE option is specified) **stopping**.
FORCE	Specifies that an involuntary stop is requested.

OSDI Command Keyword Abbreviations

The abbreviated or alternate forms that can be used for OSDI command verbs and parameter keywords are as follows:

ALTER	ALT
DEFINE	DEF
DESCRIPTION	DESC
DISPLAY	DIS, D
DRAIN	DR
FORCE	(none)
JOBACCT	ACCT
JOBNAME	JOB
LONG	L
MAXAS	MXA
MODE	MD
PARM	P
PROCEDURE	PROC
RESUME	RES
SERVICE	SRV, SVC
SERVICEGROUP	SVG, SG
SHOW	SH
SID	IDENTIFIER, ID
SSID	(none)
START	ST, S
STOP	P
SYSTEMS	SYS
TYPE	TY

B
Operating System Dependent Variables

This appendix contains the OS/390-specific allowed, default, and maximum values for Oracle initialization and storage parameters. For a complete discussion of these parameters, refer to the *Oracle9i Database Reference*. To find out which parameters described in *Oracle9i Database Reference* are supported or not supported by Oracle9*i* for OS/390, refer to the *Oracle9i Enterprise Edition Installation Guide for OS/390.*

The topics in this appendix include:

Initialization Parameters with OS/390-Specific Defaults or Limits

Table B–1 Initialization Parameters

Parameter	Default	Maximum	Minimum
COMMIT_POINT_STRENGTH	1	N/A	N/A
DB_BLOCK_SIZE	4096	32768	4096
DB_FILE_MULTIBLOCK_READ_COUNT	Derived	256	1
LOG_BUFFER	16384	Limited by address space size	N/A
NLS_LANGUAGE	AMERICAN	N/A	N/A
NLS_TERRITORY	AMERICA	N/A	N/A
PROCESSES	Derived	Limited by address space size	N/A
SORT_AREA_SIZE	65536	Limited by available session memory	N/A
TRANSACTIONS_PER_ROLLBACK_SEGMENT	5	34	N/A

Note: N/A means that the platform-specific value is the same as the generic value.

You can separate the keywords with commas or blanks.

Database Limits

Except for control files, Oracle database files (including log, tablespace, and other files) are limited to 4 gigabytes each in size. Control files are limited to 20,000 logical blocks of size DB_BLOCK_SIZE.

A maximum of 255 redo log files can be specified for a database.

A database can contain up to 65, 534 datafiles over all tablespaces. A single tablespace is limited to 4094 datafiles.

SQL Language Parameters

CREATE CONTROLFILE

Table B–2 CREATE CONTROL FILE Parameters

Parameter	Default	Maximum	Minimum
MAXDATAFILES	32	65534*	N/A
MAXINSTANCES	1	15	N/A
MAXLOGFILES	32	256	N/A
MAXLOGMEMBERS	2	5	N/A
MAXLOGHISTORY	100	65534	N/A

*Specifying MAXDATAFILES larger than 40,000 requires using a database default blocksize (DB_BLOCK_SIZE INITORA parameter value) of 8K or larger.

Note: N/A means that the platform-specific value is the same as the generic value.

CREATE DATABASE

Table B–3 CREATE DATABASE Parameters

Parameter	Default	Maximum	Minimum
DATAFILE	Refer to "Oracle Database Files" on page 4-2.	N/A	16K or one VSAM control area, whichever is larger
LOGFILE	Refer to "Oracle Database Files" on page 4-2.	N/A	N/A
MAXDATAFILES	32	65534*	N/A
MAXINSTANCES	1	15	N/A
MAXLOGFILES	32	256	2
MAXLOGMEMBERS	2	5	N/A
MAXLOGHISTORY	N/A	N/A	N/A

*Specifying MAXDATAFILES larger than 40,000 requires using a database default blocksize (DB_BLOCK_SIZE INITORA parameter value) of 8K or larger.

Note: N/A means that the platform-specific value is the same as the generic value.

Storage Parameters

The maximum values for DATAFILE, MINEXTENTS, NEXT, OPTIMAL, and PCTINCREASE are all limited by file size, which is limited by the size of the disk device.

C

Oracle9*i* for OS/390 System Symbols

This appendix documents Oracle9*i* for OS/390 system symbols for use in generating unique filenames.

System Symbols

Oracle9*i* for OS/390 system symbols can be incorporated into SQL scripts and parameter files to guarantee that unique filenames are generated with values specific to the server instance or session.

Symbol	**Description**
&ORAPREFD	A high level (leftmost) dsname qualifier for use in database filenames. It is derived from the database region parameter DSN_PREFIX_DB or ORAPREFD For more information, refer to the description on page 3-10.
&ORASESST	A distinct session identifier for use in data set names. It is based on the OSDI pid (process id) in hexadecimal format. However, if the hexadecimal representation of the pid begins with 0-9, it is converted to G-P, respectively.
&ORASRVN	The OSDI service name without trailing blanks.

D

National Language Support

This appendix documents the National Language Support (NLS) information specific to Oracle9*i* for OS/390. Information about the product-specific operation of language-specific features is provided in the *Oracle9i Database Globalization Support Guide*.

The topics in this appendix include:

Overview

Oracle's globalization support enables you to store, process, and retrieve data in native languages. National Language Support (NLS) is a subset of globalization support. It enables Oracle applications to interact with users in their native language, using their specific cultural conventions for displaying data.

The Oracle NLS architecture is data-driven, enabling support for specific languages and character encoding schemes to be added without requiring any changes in source code.

Supported Languages

Oracle9*i* for OS/390 currently supports 24 languages. This table lists the languages that are supported and the default territories for each.

Language	Default Territory
American	America
Arabic	United Arab Emirates
Bengali	Bangladesh
Brazilian Portuguese	Brazil
Bulgarian	Bulgaria
Canadian French	Canada
Catalan	Catalonia
Croatian	Croatia
Czech	Czechoslovakia
Danish	Denmark
Dutch	Netherlands
Egyptian	Egypt
English	United Kingdom
Estonian	Estonia
Finnish	Finland
French	France
German	Germany

Language	Default Territory
German DIN	Germany
Greek	Greece
Hebrew	Israel
Hungarian	Hungary
Icelandic	Iceland
Indonesian	Indonesia
Italian	Italy
Japanese	Japan
Korean	Korea
Latin American Spanish	America
Latvian	Latvia
Lithuanian	Lithuania
Malay	Malaysia
Mexican Spanish	Mexico
Norwegian	Norway
Polish	Poland
Portuguese	Portugal
Romanian	Romania
Russian	CIS
Simplified Chinese	China
Slovak	Czechoslovakia
Slovenian	Slovenia
Spanish	Spain
Swedish	Sweden
Thai	Thailand
Traditional Chinese	Taiwan
Turkish	Turkey
Ukranian	Ukraine

Language	Default Territory
Vietnamese	Vietnam

Overview of Character Set Support

Oracle automatically converts these types of data as they transfer between client and server, if required:

1. CHAR, VARCHAR, and LONG database columns
2. SQL and PL/SQL statements
3. host variables containing character data

Both the client and the server have associated character sets. The client declares its character set before connecting to the server through the NLS_LANG environment variable. On OS/390, this parameter is in the ORA$ENV DD statement.

If NLS_LANG is not specified, the default character set is assigned.

The character set for the server is declared when a database is created in Oracle and it cannot be changed once established. The default database character set is assigned if one is not explicitly declared.

When the client character set matches the server character set, character data is sent between client and server without any conversion. If the two character sets differ, all character data is converted from one character set to the other as it is transferred. It is important to be aware that all data contained in a database, whether in user-specified tables or in Oracle-specified data dictionary tables, is stored in the database character set.

Server-Side NLS

The character set in which the data is stored in the Oracle database is specified in the CHARACTER SET clause of the CREATE DATABASE statement. Refer to the *Oracle9i Database Administrator's Guide* for more information about the CREATE DATABASE statement.

When creating a database, the character set you choose depends on the language(s) to be supported and the character set(s) of the clients connecting to the server. Many languages can be supported by the default OS/390 character set of WE8EBCDIC1047, but many others cannot.

The following chart allows you to select an appropriate value for the CHARACTER SET clause of the CREATE DATABASE statement, based on the language to be supported and on the character set of the OS/390 client. If you need to support languages that are not on this chart, please contact Oracle Support Services.

Language	OS/390 Client Character Set	OS/390 Server Character Set
Arabic	AR8EBCDICX	AR8EBCDIC420S
Baltic (1)	BLT8EBCDIC1112	BLT8EBCDIC1112S
Cyrillic (1)	CL8EBCDIC1025	CL8EBCDIC1025R*
Eastern European (1)	EE8EBCDIC870	EE8EBCDIC870S
Greek	EL8EBCDIC875	EL8EBCDIC875R*
Hebrew	IW8EBCDIC424	IW8EBCDIC424S
Icelandic	WE8EBCDIC871	WE8EBCDIC871S
Japanese	JA16DBCS	JA16DBCS (2)
Korean	KO16DBCS	KO16DBCS (2)
Simplified Chinese	ZHS16DBCS	ZHS16DBCS (2)
Thai	TH8TISEBCDIC	TH8TISEBCDICS
Traditional Chinese	ZHT16DBCS	ZHT16DBCS (2)
Turkish	TR8EBCDIC1026	TR8EBCDIC1026S
Western European (a)	(3)	WE8EBCDIC1047

*OS/390 server character sets CL8EBCDIC1025S (Cyrillic) and EL8EBCDIC875S (Greek) have some errors. Oracle Corporation recommends converting from those character sets to CL8EBCDIC1025R and EL8EBCDIC875R, respectively.

Default Character Set Changed

With Oracle7 for MVS Version 7.3.2.3.50, the default character set changed from WE8EBCDIC37C to WE8EBCDIC1047. The change and potential impact are discussed here.

Reason for Change

Oracle Corporation changed the default character set from WE8EBCDIC37C to WE8EBCDIC1047 to conform to prior changes in IBM standard character sets.

Two characters are changed in the transition from WE8EBCDIC37C to WE8EBCDIC1047:

- the circumflex, which is sometimes called caret or hat
- the logical not, which is sometimes called step

This chart highlights the change:

	Circumflex	**Logical Not**
WE8EBCDIC37C	X’B0’	X’5F’
WE8EBCDIC1047	X’5F’	X’B0’

Potential Impact

Databases Created by Oracle7 -- No Impact

A database created by releases of Oracle7 for OS/390 prior to Version 7.3.2.3.50 has its character set assigned to WE8EBCDIC37C, assuming the default character set is used to create the database. Even though the data is stored in WE8EBCDIC37C and the Oracle kernel normally requires data to be stored in WE8EBCDIC1047, the change to the character set is such that the newer kernel can successfully process an existing Oracle database. However, clients accessing these databases might be affected.

Databases Created by Oracle Version 6 and Migrated to Oracle Version 7 -- Minimal Impact

A database created by Oracle Version 6 uses character set WE8EBCDIC500C. This character set is slightly different from both WE8EBCDIC37C and WE8EBCDIC1047.

When an Oracle Version 6 database migrates to Oracle7 using the migrate utility, the database character set is forced to become WE8EBCDIC37C. If a database view was created by Oracle Version 6 and was migrated to Oracle7, it can contain the obsolete WE8EBCDIC500C form of a special delimiter character known as a newline.

The presence of this character can cause unwanted results in certain circumstances. An attempt to import the view might cause an ORA-911 error if all of these are true:

1. The view was created in Oracle Version 6.
2. The view has never been exported and imported under an Oracle7 release prior to Release 7.3.

The work-around for the ORA-911 problem is to recreate the views. If the views are part of a larger export file (such as a user export), then only the views are affected. All other data within the export file is properly imported.

OS/390 Clients Using Default NLS_LANG Value with Database Character Set of WE8EBCDIC37C -- Moderate Impact

Any OS/390 client (whether precompiler, Oracle Call Interface (OCI), or SQL*Plus application) that accepts the default NLS_LANG value can be impacted in one of these ways:

- If database columns contain one or both of the two changed characters (that is, circumflex and the logical "not"), the change of default client character set means that an attempt to insert a circumflex actually causes a logical "not" to be inserted. Similarly, existing logical "nots" in the database are retrieved and converted to circumflexes. If this occurs, specifying the database character set WE8EBCDIC37C in NLS_LANG forces both client and server to declare the same character set and thereby eliminate all conversion. Doing this reverts to Oracle Version 7 behavior prior to release 7.3.2.3.50.
- If neither of the two reassigned characters is used in an application, then proper application function can be achieved by letting the client default. However, adopting the default causes the client to specify WE8EBCDIC1047 and causes the server to specify WE8EBCDIC37C, which triggers character conversion where it was not used previously. In applications that move a considerable amount of character data to or from the database, CPU use might increase. Specifying the database character set WE8EBCDIC37C in NLS_LANG forces both client and server to declare the same character set. This eliminates all conversion overhead.
- If one of the previous two factors requires addition of NLS_LANG, the entire NLS_LANG parameter (`language, territory,` and `character_set`)

might have to be specified. If `language` and `territory` are omitted from NLS_LANG, then they default to American and America, respectively, even if the server specified a `language` or `territory` default in its initialization parameters. If American and America are suitable defaults, then NLS_LANG=.WE8EBCDIC37C can be used.

Client-Side NLS

The client character set is determined by the data that is sent to the Oracle server. This is typically determined by the type of terminal that is used by the client. See your system administrator for more information about the character set used by your terminal.

Supported OS/390 Client Character Sets

The default client character set has changed from WE8EBCDIC37C to WE8EBCDIC1047, but most existing Oracle7 for OS/390 databases are WE8EBCDIC37C. When the client and server character sets differ, the CPU overhead increases. If you set NLS_LANG so that the client and server character sets match, the CPU overhead decreases.

The following character sets are supported for OS/390 clients:

Character Set	**Support**
AR8EBCDICX	XBASIC Code Page 420 Arabic
BLT8EBCDIC1112	EBCDIC Code Page 1112 Baltic/Multilingual
CL8EBCDIC1025	EBCDIC Code Page 1025 Cyrillic/Multilingual
D8EBCDIC273	EBCDIC Code Page 273 Austrian /German
DK8EBCDIC277	EBCDIC Code Page 277 Danish
EE8EBCDIC870	EBCDIC Code Page 870 East European
EL8EBCDIC875	EBCDIC Code Page 875 Greek
F8EBCDIC297	EBCDIC Code Page 297 French
I8EBCDIC280	EBCDIC Code Page 280 Italian
IW8EBCDIC424	EBCDIC Code Page 424 Hebrew
JA16DBCS	EBCDIC DBCS Japanese

Character Set	Support
KO16DBCS	EBCDIC DBCS Korean
S8EBCDIC278	EBCDIC Code Page 278 Swedish
TH8TISEBCDIC	EBCDIC Code Page 838 Thai
TR8EBCDIC1026	EBCDIC Code Page ISO 8859 Turkish
WE8EBCDIC37	EBCDIC Code Page 37 West European
WE8EBCDIC37C	EBCDIC Code Page 37 West European with extensions
WE8EBCDIC284	EBCDIC Code Page 284 Spanish (Spain)
WE8EBCDIC285	EBCDIC Code Page 285 English, UK
WE8EBCDIC500	EBCDIC Code Page 500 West European
WE8EBCDIC871	EBCDIC Code Page 871 Icelandic
WE8EBCDIC1047	EBCDIC Code Page 1047 Latin 1/Open Systems
ZHS16DBCS	EBCDIC DBCS Simplified Chinese
ZHT16DBCS	EBCDIC DBCS Traditional Chinese

Setting the OS/390 Client NLS Parameters

Use the following language parameters and environment variables to specify the OS/390 client NLS language, territory, and character set for the following components and tools:

Component	Parameter or Environment Variable
Export/Import	NLS_LANG
Oracle Access Manager for CICS	ENAME
Oracle Access Manager for IMS TM	NLS_LANG
Server Manager	NLS_LANG
SQL*Loader	NLS_LANG
SQL*Plus	NLS_LANG
Oracle Reports	ORACLE.LANGUAGE and NLS_LANG

ENAME Parameter

For Oracle Access Manager for CICS, specify the following thread table parameters:

```
TYPE=ENV,ENAME=(NLS_LANG=language_territory.charset)
```

This thread table specification is used for Oracle Access Manager for CICS messages and messages from Oracle tools executing under Oracle Access Manager for CICS.

NLS_LANG Environment Variable

To specify the character set with the NLS_LANG environment variable, use the following syntax:

```
NLS_LANG = "language_territory.charset"
```

Refer to the *Oracle9i Enterprise Edition User's Guide for OS/390* for more information about environment variables.

ORACLE.LANGUAGE Environment Variable

To specify the character set with the ORACLE.LANGUAGE environment variable, use the following syntax:

```
ORACLE.LANGUAGE = "language_territory.charset"
```

Message Availability

Availability of the supported language message modules depends on which modules are installed in the Oracle9*i* for OS/390 product set. If you do not have message modules for a particular language set installed, then specifying that language with a language parameter does not display messages in the requested language.

Refer to panel ORLANG in the *Oracle9i Enterprise Edition Installation Guide for OS/390* for a list of currently supported language message modules.

Customized Character Sets (LXINST)

This section provides OS/390-specific information required to customize your character sets as described in Appendix B, "Customizing Locale Data" in *Oracle9i Database Globalization Support Guide*. Customize a character set only if it is absolutely necessary.

The process for customizing a character set is accomplished primarily on OS/390 Unix System Services. You will need to be familiar with OS/390 UNIX in general, as well as the material in Appendix B of the *Oracle9i Enterprise Edition User's Guide for OS/390* describing the particular environment variables that are required by Oracle products in the OS/390 UNIX environment.

The OS/390 implementation of Oracle products uses the same file names and directory structure that generic Oracle product documentation uses, so as you review the "Customizing Locale Data" Appendix in the *Oracle9i Database Globalization Support Guide* and plan for customizing your character sets. Log on to your OS/390 UNIX environment and verify that you can locate the directories and files described in the book. If you have any difficulty locating these files, ensure that the ORACLE_HOME environment value is properly set and that the OS/390 UNIX products are properly installed. If the difficulty persists, contact Oracle Support Services for assistance.

To customize your character set, follow the steps outlined in "Example of Character Set Customization" in the *Oracle9i Database Globalization Support Guide*. When you are directed to create text files by using "vi",

```
vi /tmp/lx0boot.nlt
```

you might find it more convenient to use the oedit command under OS/390 UNIX. You can use "vi" if you rlogin to OS/390 UNIX. Bear in mind that UNIX is case sensitive. Command, directory, and file names must be typed exactly as they appear.

After you complete Step 5, "Generate and install the .nlb files", you must update the .nlb files that are used by the OS/390 UNIX utilities as they run in the OS/390 UNIX environment.

Oracle products running in the traditional OS/390 environment will not access the .nlb files that you have just created. To make these files available to the OS/390 environment, the nlb2mvs command must be executed.

To run nlb2mvs, the following environment variables must be set or must be allowed to default to:

- ORACLE_HOME

nlb2mvs will convert the value of this environment variable to a form suitable for OS/390 and will use it as the high-level qualifier(s) of several OS/390 data sets needed by nlb2mvs. For example, if ORACLE_HOME is `"/oran/orav"`, the value "*`ORAN.ORAV`*" will be used as the high-level qualifier. If your naming conventions do not follow this assumption, then specify the rest of the following environment variables:

ORACLE_NLB_PDS	overrides the name of the PDS used to hold the output NLS objects. The default is $ORACLE_HOME.OCOMMON.NLS.ADMIN.DATA
ORACLE_NLB_OBJ	overrides the name of a work data set. The default is $ORACLE_HOME.OCOMMON.NLB.OBJ
ORACLE_NLB_LIST	overrides the name of a work data set. The default is $ORACLE_HOME.OCOMMON.NLB.LIST

Oracle Corporation recommends you thoroughly test your new NLS objects before moving them into production.

NLS Calendar Utility (LXEGEN)

The setup for running this utility is similar to that for running customized character sets (LXINST). Refer to the previous section, "Customized Character Sets (LXINST)" on page D-10. Whereas the LXINST utility is used to customize character sets, the LXEGEN utility is used in a similar manner to customize calendar data.

Index