Oracle9*i*

Database Migration

Release 1 (9.0.1)

July 2001

Part No. A90191-02

Oracle9*i* Database Migration, Release 1 (9.0.1)

Part No. A90191-02

Copyright © 1996, 2001, Oracle Corporation. All rights reserved.

Primary Author: Tony Morales

Graphic Artist: Valarie Moore

Contributors: Nipun Agarwal, Rick Anderson, Vikas Arora, Neerja Bhatt, Bill Bridge, Rae Burns, Thong Bui, Ben Chang, Debashish Chatterjee, Lakshminaray Chidambaran, Eugene Chong, George Claborn, David Colello, Jay Davison, Alan Downing, Sreenivas Gollapudi, Brajesh Goyal, Michael Hartstein, Jeffrey Hebert, Thuvan Hoang, Wei Huang, Robert Jenkins, Sanjeev Jhala, Christopher Jones, Sanjay Kaluskar, Dhiraj Kapoor, Vishwanath Karra, Mark Kennedy, Susan Kotsovolos, Viswanathan Krishnamurthy, Muralidhar Krishnaprasad, Thomas Kurian, Paul Lane, Gordon Larimer, Simon Law, Jing Liu, Juan Loaiza, Neil Le, J. Bill Lee, Bill Maimone, Raghu Mani, Shailendra Mishra, Ari Mozes, Kannan Muthukkaruppan, Subramanian Muralidhar, Ravi Murthy, Karuna Muthiah, Miranda Nash, Mark Niebur, Anil Nori, Peter Ogilvie, Irene Paradisis, Rosanne Park, Elizabeth Pitt, Greg Pongracz, Franco Putzolu, N. C. Ramesh, Paul Raveling, Ann Rhee, Richard Sarwal, Ajay Sethi, Carol Sexton, Helen Slaterry, James Stamos, Debbie Steiner, Alex Tsukerman, Randy Urbano, Guhan Viswanathan, Steven Wertheimer, Rick Wessman, Andrew Witkowski, Lik Wong, Aravind Yalamanchi, Qin Yu

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

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

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

Restricted Rights Notice Programs delivered subject to the DOD FAR Supplement are "commercial computer software" and use, duplication, and disclosure of the Programs, including documentation, shall be subject to the licensing restrictions set forth in the applicable Oracle license agreement. Otherwise, Programs delivered subject to the Federal Acquisition Regulations are "restricted computer software" and use, duplication, and disclosure of the Programs shall be subject to the restrictions in FAR 52.227-19, Commercial Computer Software - Restricted Rights (June, 1987). Oracle Corporation, 500 Oracle Parkway, Redwood City, CA 94065.

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

Oracle is a registered trademark, and Oracle8*i*, Oracle9*i*, PL/SQL, Pro*Ada, Pro*COBOL, Pro*C, Pro*C/C++, SQL*Plus, and Trusted Oracle are trademarks or registered trademarks of Oracle Corporation. Other names may be trademarks of their respective owners.

Contents

Send Us Your Comments

Oracle9*i* Database Migration, Release 1 (9.0.1)

Part No. A90191-02

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

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

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

- Electronic mail: infodev_us@oracle.com
- FAX: (650) 506-7227   Attn: Server Technologies Documentation Manager
- Postal service:
  Oracle Corporation
  Server Technologies Documentation
  500 Oracle Parkway, Mailstop 4op11
  Redwood Shores, CA 94065
  USA

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

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

Preface

This manual guides you through the process of planning and executing migrations, upgrades, and downgrades on the Oracle database system. In addition, this manual provides information about compatibility, about upgrading applications to the current release of Oracle, and about important changes in the current release, such as initialization parameter changes and data dictionary changes.

Oracle9*i* Database Migration contains information that describes the features and functionality of the Oracle9*i* (also known as the standard edition) and the Oracle9*i* Enterprise Edition products. Oracle9*i* and the Oracle9*i* Enterprise Edition have the same basic features. However, several advanced features are available only with the Enterprise Edition, and some of these are optional. For example, to use application failover, you must have the Enterprise Edition with the Oracle9*i* Real Application Clusters option.

> **See Also:** *Oracle9i Database New Features* for information about the differences between Oracle9*i* and the Oracle9*i* Enterprise Edition and the features and options that are available to you.

This preface contains these topics:

- Audience
- Organization
- Related Documentation
- Conventions
- Documentation Accessibility

Audience

Oracle9*i* Database Migration is intended for database administrators (DBAs), application developers, security administrators, system operators, and anyone who plans or executes migration, upgrade, or downgrade operations on Oracle databases.

To use this document, you need to be familiar with the following:

- Oracle database management system (DBMS) concepts
- Your current release of the Oracle database server
- Your operating system environment

Organization

This document contains:

Chapter 1, "Introduction"

This chapter contains definitions for terms used throughout this manual. This chapter also provides information about running different versions and releases of Oracle on the same computer system.

Chapter 2, "Overview of Migration"

This chapter summarizes migration procedures and the responsibilities of database administrators and application developers.

Chapter 3, "Preparing to Migrate"

This chapter describes the steps to complete before migrating a database.

Chapter 4, "Migrating from Oracle7 Using the Migration Utility"

This chapter provides step-by-step instructions for using the Migration utility to migrate an Oracle7 database to Oracle9*i*.

Chapter 5, "Migrating from Oracle7 Using the Oracle Data Migration Assistant"

This chapter provides step-by-step instructions for using the Oracle Data Migration Assistant to migrate an Oracle7 database to Oracle9*i*.

Chapter 6, "Migrating Using Export/Import"

This chapter describes how to migrate a version 6 or Oracle7 database to Oracle9*i* using the Export and Import utilities.

Chapter 7, "Upgrading from an Older Release of Oracle to the New Oracle9i Release"

This chapter provides step-by-step instructions for performing the following actions:

- Upgrading a database from a version 8 release to the new Oracle9*i* release
- Upgrading specific components of Oracle software to the current release
- Changing the word size of your database (switching between 32-bit and 64-bit software)

Chapter 8, "After Migrating or Upgrading the Database"

This chapter describes the actions to complete after migrating or upgrading the database to the new Oracle9*i* release.

Chapter 9, "Compatibility and Interoperability"

This chapter contains information about compatibility and interoperability between different releases of Oracle, including detailed information about the `COMPATIBLE` initialization parameter. This chapter also lists the Oracle9*i* features that require a 9.0.0 or higher compatibility level and discusses specific issues relating to compatibility and interoperability.

Chapter 10, "Upgrading Your Applications"

This chapter provides general information about upgrading Oracle7 applications and tools for use with Oracle9*i*.

Chapter 11, "Migrating from Server Manager to SQL*Plus"

This chapter describes modifying your Server Manager line mode scripts for use with SQL*Plus.

Chapter 12, "Migration Issues for Physical Rowids"

This chapter covers issues associated with ROWIDs in release 8.0 and higher, including specific information about migrating columns containing ROWIDs to release 8.0 and higher.

Chapter 13, "Downgrading to Release 8.1"

This chapter provides instructions for downgrading a database from Oracle9*i* to release 8.1.

Chapter 14, "Removing Incompatibilities Before Downgrading to Release 8.1"

This chapter provides instructions for removing incompatibilities before downgrading to release 8.1.

Chapter 15, "Downgrading to an Older Release of Oracle"

This chapter provides instructions for downgrading a database from Oracle9*i* to Oracle7 or release 8.0.

Appendix A, "Troubleshooting Migration Problems"

This appendix describes common migration problems and the actions required to correct these problems. In addition, this appendix lists the messages displayed by the Migration utility, and includes an explanation of each message. If the message is an error message, then this appendix discusses probable causes of the error, and suggests corrective action for the error.

Appendix B, "Changes to Initialization Parameters"

This appendix lists initialization parameters that are important for migration. Specifically, this appendix describes initialization parameters that have been added, renamed, or obsoleted in version 8 and Oracle9*i*. In addition, this appendix describes compatibility issues relating to specific initialization parameters.

Appendix C, "Changes to Static Data Dictionary Views"

This appendix lists static data dictionary views that have been added, changed, or obsoleted in version 8 and Oracle9*i*. This appendix also lists static data dictionary views with added columns, dropped columns, and renamed columns. In addition, this appendix lists columns in static data dictionary views that may return NULLs in release 8.1 but did not return NULLs in release 8.0 and earlier.

Appendix D, "Changes to Dynamic Performance Views"

This appendix lists dynamic performance views (V$ views) that have been added, changed, or obsoleted in version 8 and Oracle9*i*. This appendix also lists dynamic performance views with added columns and dropped columns.

Appendix E, "New Internal Datatypes and SQL Functions"

This appendix lists new internal datatypes and SQL functions.

Appendix F, "Migration and Compatibility for Oracle Net Services"

This appendix discusses considerations for migrating SQL*Net from Oracle7 to Oracle9*i*. This appendix also discusses considerations for upgrading SQL*Net or Oracle Net from version 8 to Oracle9*i*.

Appendix G, "Migration and Compatibility for Replication Environments"

This appendix provides step-by-step instructions for migrating an Oracle replication system on an Oracle7 database to Oracle9*i*. This appendix also discusses compatibility issues between different versions of Oracle replication.

Related Documentation

For more information, see these Oracle resources:

- *Oracle9i Database Concepts* for a comprehensive introduction to the concepts and terminology used in this manual
- *Oracle9i Database Administrator's Guide* for information about administering the Oracle database server
- *Oracle9i SQL Reference* for information on Oracle's SQL commands and functions
- *Oracle9i Database Utilities* for information about the utilities bundled with the Oracle database server, including Export, Import, and SQL*Loader
- *Oracle Net Services Administrator's Guide* for information about Oracle Net Services

Many of the examples in this book use the sample schemas of the seed database, which is installed by default when you install Oracle. Refer to *Oracle9i Sample Schemas* for information on how these schemas were created and how you can use them yourself.

In North America, printed documentation is available for sale in the Oracle Store at

```
http://oraclestore.oracle.com/
```

Customers in Europe, the Middle East, and Africa (EMEA) can purchase documentation from

```
http://www.oraclebookshop.com/
```

Other customers can contact their Oracle representative to purchase printed documentation.

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

`http://technet.oracle.com/membership/index.htm`

If you already have a username and password for OTN, then you can go directly to the documentation section of the OTN Web site at

`http://technet.oracle.com/docs/index.htm`

Conventions

This section describes the conventions used in the text and code examples of this documentation set. It describes:

- Conventions in Text
- Conventions in Code Examples

Conventions in Text

We use various conventions in text to help you more quickly identify special terms. The following table describes those conventions and provides examples of their use.

Convention	Meaning	Example
Bold	Bold typeface indicates terms that are defined in the text or terms that appear in a glossary, or both.	When you specify this clause, you create an **index-organized table**.
Italics	Italic typeface indicates book titles or emphasis.	*Oracle9i Database Concepts* Ensure that the recovery catalog and target database do *not* reside on the same disk.
`UPPERCASE monospace (fixed-width font)`	Uppercase monospace typeface indicates elements supplied by the system. Such elements include parameters, privileges, datatypes, RMAN keywords, SQL keywords, SQL*Plus or utility commands, packages and methods, as well as system-supplied column names, database objects and structures, usernames, and roles.	You can specify this clause only for a `NUMBER` column. You can back up the database by using the `BACKUP` command. Query the `TABLE_NAME` column in the `USER_TABLES` data dictionary view. Use the `DBMS_STATS.GENERATE_STATS` procedure.

Convention	Meaning	Example
`lowercase monospace (fixed-width font)`	Lowercase monospace typeface indicates executables, filenames, directory names, and sample user-supplied elements. Such elements include computer and database names, net service names, and connect identifiers, as well as user-supplied database objects and structures, column names, packages and classes, usernames and roles, program units, and parameter values. **Note:** Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.	Enter `sqlplus` to open SQL*Plus. The password is specified in the `orapwd` file. Back up the datafiles and control files in the `/disk1/oracle/dbs` directory. The `department_id`, `department_name`, and `location_id` columns are in the `hr.departments` table. Set the `QUERY_REWRITE_ENABLED` initialization parameter to `true.` Connect as `oe` user. The `JRepUtil` class implements these methods.
`lowercase monospace (fixed-width font) italic`	Lowercase monospace italic font represents placeholders or variables.	You can specify the *`parallel_clause`*. Run `U`*`old_release`*`.SQL` where *`old_release`* refers to the release you installed prior to upgrading.

Conventions in Code Examples

Code examples illustrate SQL, PL/SQL, SQL*Plus, or other command-line statements. They are displayed in a monospace (fixed-width) font and separated from normal text as shown in this example:

```
SELECT username FROM dba_users WHERE username = 'MIGRATE';
```

The following table describes typographic conventions used in code examples and provides examples of their use.

Convention	Meaning	Example
[ ]	Brackets enclose one or more optional items. Do not enter the brackets.	`DECIMAL (`*`digits`*` [ , `*`precision`*` ])`
{ }	Braces enclose two or more items, one of which is required. Do not enter the braces.	`{ENABLE | DISABLE}`

|	A vertical bar represents a choice of two or more options within brackets or braces. Enter one of the options. Do not enter the vertical bar.	`{ENABLE | DISABLE}`
		`[COMPRESS | NOCOMPRESS]`

Convention	Meaning	Example

`. . .`	Horizontal ellipsis points indicate either:
	■ That we have omitted parts of the code that are not directly related to the example
	■ That you can repeat a portion of the code
		`CREATE TABLE ... AS` *`subquery`*`;`
		
		`SELECT` *`col1`*`,` *`col2`*`, ... ,` *`coln`* `FROM employees;`

. . .	Vertical ellipsis points indicate that we have omitted several lines of code not directly related to the example.	

Other notation	You must enter symbols other than brackets, braces, vertical bars, and ellipsis points as shown.	`acctbal NUMBER(11,2);`
		`acct    CONSTANT NUMBER(4) := 3;`

`Italics`	Italicized text indicates placeholders or variables for which you must supply particular values.	`CONNECT SYSTEM/`*`system_password`*
		`DB_NAME =` *`database_name`*

`UPPERCASE`	Uppercase typeface indicates elements supplied by the system. We show these terms in uppercase in order to distinguish them from terms you define. Unless terms appear in brackets, enter them in the order and with the spelling shown. However, because these terms are not case sensitive, you can enter them in lowercase.	`SELECT last_name, employee_id FROM employees;`
		`SELECT * FROM USER_TABLES;`
		`DROP TABLE hr.employees;`

`lowercase`	Lowercase typeface indicates programmatic elements that you supply. For example, lowercase indicates names of tables, columns, or files.
	Note: Some programmatic elements use a mixture of UPPERCASE and lowercase. Enter these elements as shown.
		`SELECT last_name, employee_id FROM employees;`
		`sqlplus hr/hr`
		`CREATE USER mjones IDENTIFIED BY ty3MU9;`

Documentation Accessibility

Oracle's goal is to make our products, services, and supporting documentation accessible to the disabled community with good usability. 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 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/`

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.

1 Introduction

This chapter includes definitions for words used throughout this manual and information about changing the word-size of your database. This chapter also provides information about running different versions and releases of Oracle on the same computer system. In addition, this chapter covers other topics that relate to migration, upgrading, and downgrading operations.

Specifically, this chapter covers the following topics:

- Terminology
- Running Multiple Versions and Releases of Oracle
- Using Optimal Flexible Architecture (OFA)
- Running Scripts
- Changing Word-Size
- Rolling Upgrades for Oracle9i Real Application Clusters
- Deinstalling Options

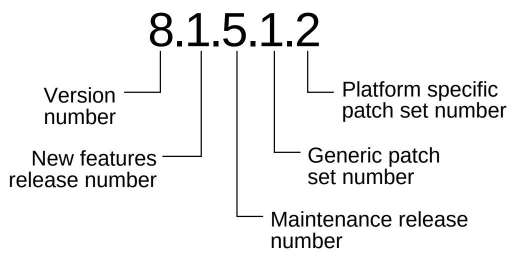
Terminology

The following terms are used throughout this document:

Versions and Releases

The instructions in this document describe moving between different **versions** and **releases** of Oracle. Figure 1–1 describes what each part of a release number represents.

Figure 1–1 Example of an Oracle Release Number


8.1.5.1.2
Version number
New features release number
Maintenance release number
Generic patch set number
Platform specific patch set number

See Also: *Oracle9i Database Administrator's Guide* for more information about Oracle release numbers.

When a statement is made in this book about a version of Oracle, the statement applies to all releases in that version. References to version 8 include all releases in release 8.0 and release 8.1; references to Oracle7 include all version 7 releases, including release 7.0, 7.1, 7.2, and 7.3.

Similarly, when a statement is made in this book about a maintenance release, the statement applies to all production patch releases and port specific patch releases within that maintenance release. So, a statement about release 8.1 applies to all production releases within release 8.1, including release 8.1.5, 8.1.6, and 8.1.7, but not necessarily to release 8.1.4, because that release was a beta release.

The same logic applies to patch releases. When a statement is made in this book about a patch release, the statement applies to all port-specific patch releases within that patch release. So, a statement about release 8.1.5 applies to release 8.1.5.0, 8.1.5.1, and all other port-specific patch releases within release 8.1.5.

Migration vs. Upgrading

Migration is the process of transforming an installed version of an Oracle7 database into a later version. For example, transforming an Oracle7 database into an Oracle9*i* database requires a migration.

> **See Also:** Chapter 2, "Overview of Migration" to begin the migration process.

Upgrading is the process of transforming an installed version or release of an Oracle database into a later version or release, whenever a migration is not required. For example, transforming a version 8 database into an Oracle9*i* database requires an upgrade.

> **See Also:** Chapter 7, "Upgrading from an Older Release of Oracle to the New Oracle9i Release", for information about upgrading.

Downgrading is the process of transforming an installed version or release of an Oracle database into an earlier version or release. For example, transforming a release 8.1.7 database back into a release 8.1.5 database is downgrading, and transforming an Oracle9*i* database back into an Oracle7 database is downgrading.

> **See Also:** Chapter 13, "Downgrading to Release 8.1", for information about downgrading a release 9.0.1 database to an 8.1 release. See Chapter 15, "Downgrading to an Older Release of Oracle", for information about downgrading to Oracle7 or release 8.0.

Source Database and Target Database

The **source database** is the database to be migrated to Oracle9*i*; during migration, the source database uses an older version of Oracle software, such as version 6 or Oracle7. The **target database** is the database into which you are migrating the source database; during migration, the target database uses new Oracle9*i* software.

Running Multiple Versions and Releases of Oracle

You can run different versions of Oracle on the same computer system at the same time. However, each version can only access a database that is consistent with its version. For example, if you have version 7 and version 8 installed on the same computer system, then the version 7 server can access version 7 databases but not version 8 databases, and the version 8 server can access version 8 databases but not version 7 databases. The following sections provide general information about running multiple versions and releases of Oracle.

Caution: It is not possible to install release 9.0.1 products into an existing Oracle7 or version 8 Oracle home. This functionality was only available for certain previous releases and has not been continued. An Oracle9*i* release must be installed in an Oracle home that is separate from previous releases of Oracle. Also, you cannot have more than one release per Oracle home. Oracle Corporation recommends that you adopt an Optimal Flexible Architecture (OFA) when creating multiple Oracle homes. See "Using Optimal Flexible Architecture (OFA)" on page 1-11 for more information.

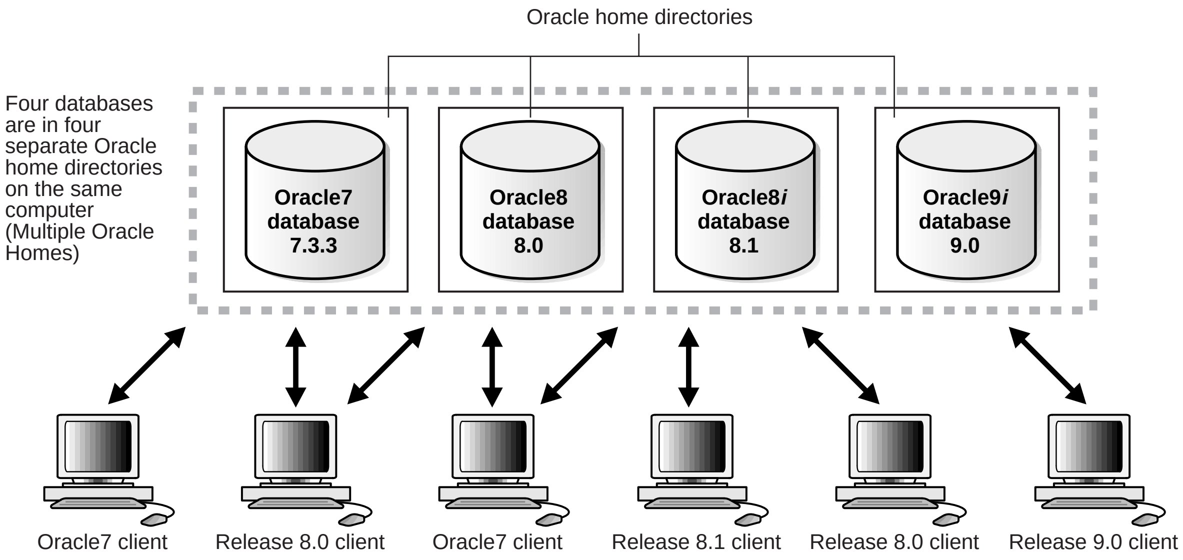
See Also: Your operating system-specific Oracle documentation for more information about running multiple versions and releases of Oracle on your operating system. Restrictions may apply on some operating systems.

Install Version 7 and Version 8 Databases in Multiple Oracle Homes

You can install version 7 and version 8 databases in multiple (separate) Oracle homes and have both version 7 and version 8 clients connecting to version 7 and version 8 databases.

Figure 1–2 illustrates this network configuration:

Figure 1–2 Version 7 and Version 8 Databases in Multiple Oracle Homes

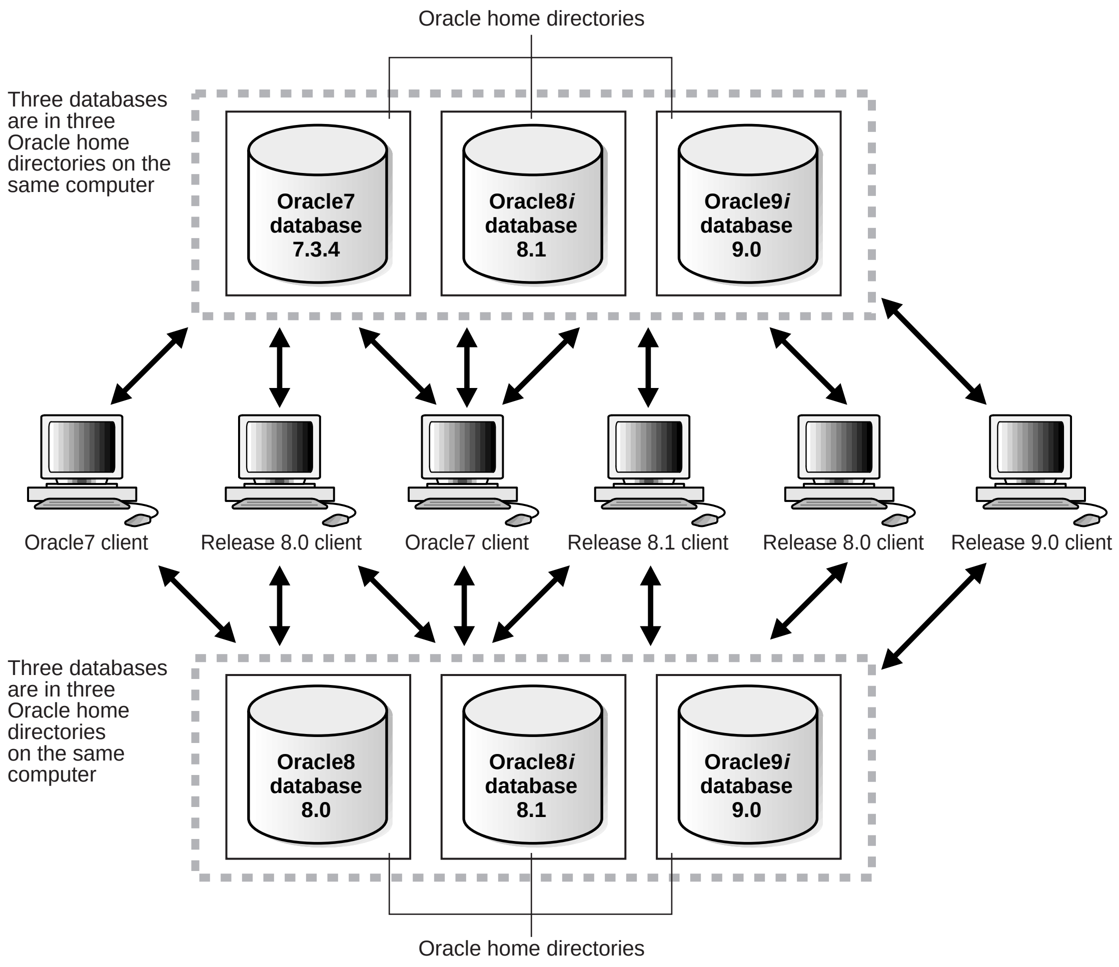


Install Version 7 and Version 8 Databases on Separate Computers

You can install version 7 and version 8 databases on separate computers and have both version 7 and version 8 clients connecting to both databases.

Figure 1–3 illustrates this network configuration:

Figure 1–3 Version 7 and Version 8 Databases on Separate Computers

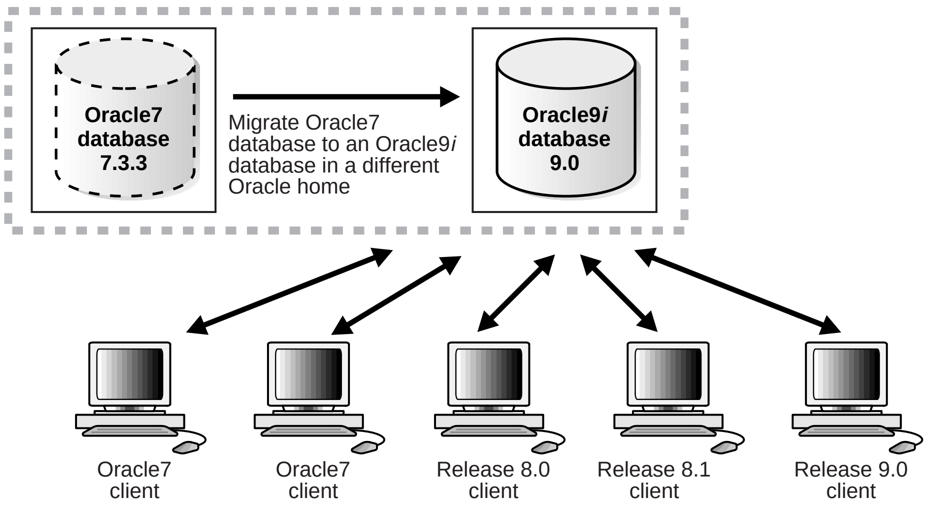


Migrate a Version 7 Database to a Version 8 Database

You can migrate your version 7 database to a version 8 database and have both version 7 and version 8 clients connecting to the version 8 database. You cannot migrate your version 7 database to a version 8 database in the same Oracle home.

Figure 1–4 illustrates this network configuration:

Figure 1–4 Migrate a Version 7 Database to a Version 8 Database

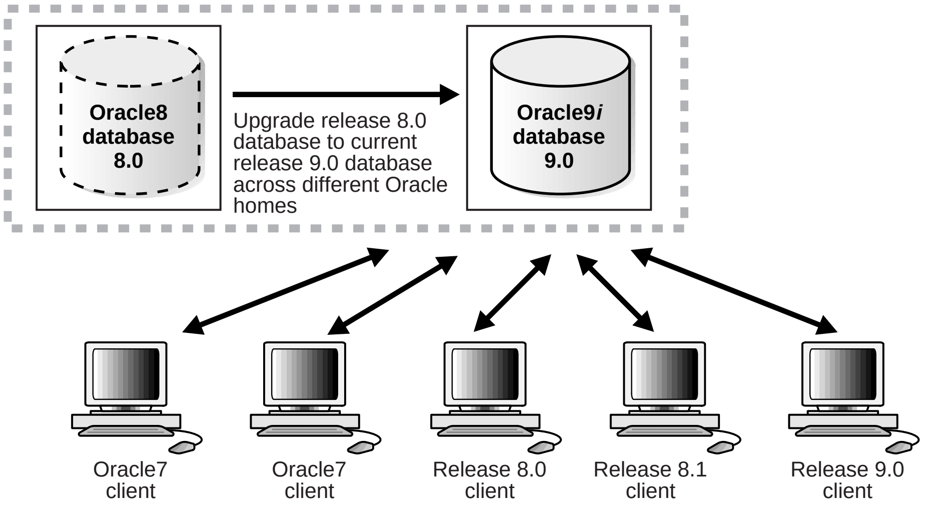


Upgrade a Release 8.0 Database to a Release 9.0.0 Database

You can upgrade your release 8.0 database, for example release 8.0.6, to release 9.0.1 and have both version 7 and release 8.0 clients connecting to the release 9.0.1 database. You cannot upgrade your release 8.0 database to a release 9.0.1 database in the same Oracle home.

Figure 1–5 illustrates this network configuration:

Figure 1–5 Upgrade a Release 8.0 Database to Release 9.0.0

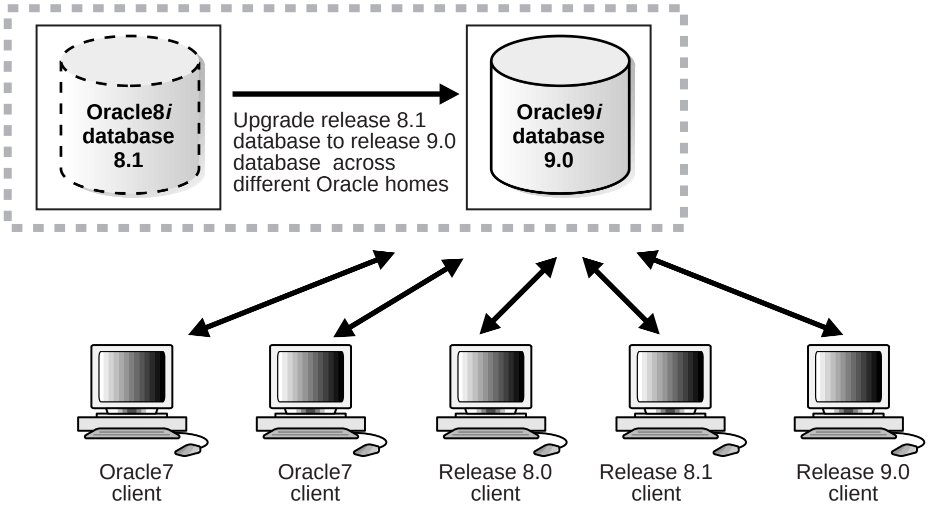


Upgrade a Release 8.1 Database to the Current Release

You can upgrade a release 8.1 database, for example release 8.1.5, to the current 9.0.0 release and have version 7, release 8.0, release 8.1, and release 9.0.1 clients connecting to the release 9.0.1 database. You must use a separate Oracle home directory for the new 9.0.0 release.

Figure 1–6 illustrates this network configuration:

Figure 1–6 Upgrade a Release 8.1 Database to the Current Release

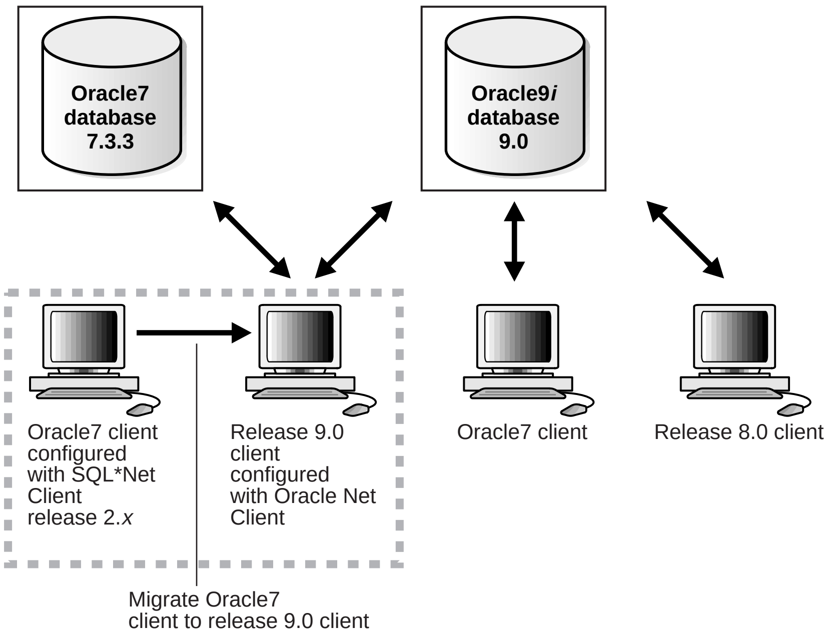


Migrate Version 7 Clients to Version 8 Clients

You can migrate some or all of your version 7 clients to release 8.1. You can also migrate your version 7 database to a release 8.1 database or upgrade your release 8.0 database to the current release 8.1 database at a later date.

Figure 1–7 illustrates this network configuration:

Figure 1–7 Migrate Version 7 Clients to Version 8 Clients


Oracle7 database 7.3.3
Oracle9*i* database 9.0
Oracle7 client configured with SQL*Net Client release 2.*x*
Release 9.0 client configured with Oracle Net Client
Oracle7 client
Release 8.0 client
Migrate Oracle7 client to release 9.0 client

Using Optimal Flexible Architecture (OFA)

Whether you are migrating a version 7 database or upgrading a version 8 database, Oracle Corporation recommends the Optimal Flexible Architecture (OFA) standard for your Oracle9*i* installations. The OFA standard is a set of configuration guidelines for efficient and reliable Oracle databases that require little maintenance.

OFA provides the following benefits:

- Organizes large amounts of complicated software and data on disk to avoid device bottlenecks and poor performance
- Facilitates routine administrative tasks, such as software and data backup functions, which are often vulnerable to data corruption
- Alleviates switching among multiple Oracle databases
- Adequately manages and administers database growth
- Helps to eliminate fragmentation of free space in the data dictionary, isolate other fragmentation, and minimize resource contention

If you are not currently using the OFA standard, then switching to the OFA standard involves modifying your directory structure and relocating your database files.

> **See Also:** Your Oracle operating system-specific documentation for more information about OFA, and *Oracle9i Database Administrator's Guide* for information about relocating your database files.

Running Scripts

You need to run various scripts when you perform migration, upgrade, and downgrade operations. When you run a script, the script may report `ORA-` errors. In general, you should look for errors that alert you to insufficient space, and for errors that alert you that a script failed to run. If you see these types of errors, then the operation may not be completely successful. However, you typically can ignore errors about the failure to alter or drop an object that does not exist.

> **See Also:** *Oracle9i Database Error Messages* for more information about `ORA-` errors

Changing Word-Size

You can change the word-size of your Oracle database server during a migration, upgrade, or downgrade operation. A change in word-size includes the following scenarios:

- You have 32-bit Oracle software installed on 64-bit hardware and want to change to 64-bit Oracle software.
- You have 64-bit Oracle software installed on 64-bit hardware and want to change to 32-bit Oracle software.

If you are changing word-size during a migration, upgrade, or downgrade operation, then no additional action is required. The word-size is changed automatically during any of these operations. However, if you want to change the word-size within the same release, then follow the instructions in "Changing the Word-Size of Your Current Release" on page 7-40. For example, if you have the 32-bit version of Oracle release 9.0.1 and you want to switch to the 64-bit version of Oracle release 9.0.1, then you must complete this procedure.

The following information applies if you are upgrading or downgrading your hardware from 32-bit to 64-bit or from 64-bit to 32-bit:

- If you want to upgrade your hardware, then you should be able to switch from 32-bit hardware to 64-bit hardware and still use your existing 32-bit Oracle software without encountering any problems.
- If you want to downgrade your hardware from 64-bit to 32-bit, then you must first downgrade your Oracle software to 32-bit software before downgrading your hardware.

The on-disk format for database data, redo, and undo is identical for the 32-bit and 64-bit installations of Oracle. The only internal structural differences between the 32-bit and 64-bit Oracle installations are the following:

- The compiled format of PL/SQL is different. The instructions for how and when to recompile PL/SQL are provided in the appropriate chapters of this book.
- The storage format of user-defined types is based on the release of Oracle that created the database. The existing storage format will be converted to the correct format transparently when necessary. User-defined types include object types, REFs, varrays, and nested tables.

Rolling Upgrades for Oracle9*i* Real Application Clusters

Rolling upgrades are not supported with Oracle9*i* Real Application Clusters. A rolling upgrade is one in which multiple instances of the Oracle database server, having different release levels, simultaneously access the same database during the upgrade process. If you are using Oracle9*i* Real Application Clusters, then you must upgrade all instances at the same time.

Note: Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

Deinstalling Options

If you want to deinstall old options when you migrate or upgrade to a new release of Oracle, then use the installer to deinstall them. You can deinstall them before or after you upgrade or migrate, but you must use the version of the installer that corresponds with the items you want to remove.

For example, if you are running release 8.1 of Oracle with Oracle Parallel Server installed, and you decide that you do not need this option when you upgrade to Oracle9*i*, then you should deinstall Oracle Parallel Server in one of the following ways:

- Before you upgrade to Oracle9*i*, use the Oracle Universal Installer in your 8.1 release to deinstall Oracle Parallel Server. Then, do not install Oracle9*i* Real Application Clusters when you install Oracle9*i*.
- When you upgrade to Oracle9*i*, install and upgrade Oracle9*i* Real Application Clusters. Then, use the Oracle Universal Installer in Oracle9*i* to deinstall Oracle9*i* Real Application Clusters.

Note: After you deinstall an option, extraneous data dictionary tables may remain.

See Also: Your operating system-specific Oracle documentation for information about using the Oracle Universal Installer.

2 Overview of Migration

This chapter includes an overview of the major steps required to migrate a version 7 or version 6 database to Oracle9*i*. These migration procedures transform an existing version 7 or version 6 database system (including associated applications) into an Oracle9*i* database system. Oracle9*i* is compatible with all earlier Oracle versions and releases. Therefore, databases transformed using the migration procedures described in this book can work in the same manner as in earlier versions and, optionally, can leverage new Oracle9*i* functionality.

Several preparatory steps are required before you migrate the current production database. After migrating the database, you should perform several additional test steps to test the migration. Other procedures enable you to add new Oracle9*i* functionality to existing applications.

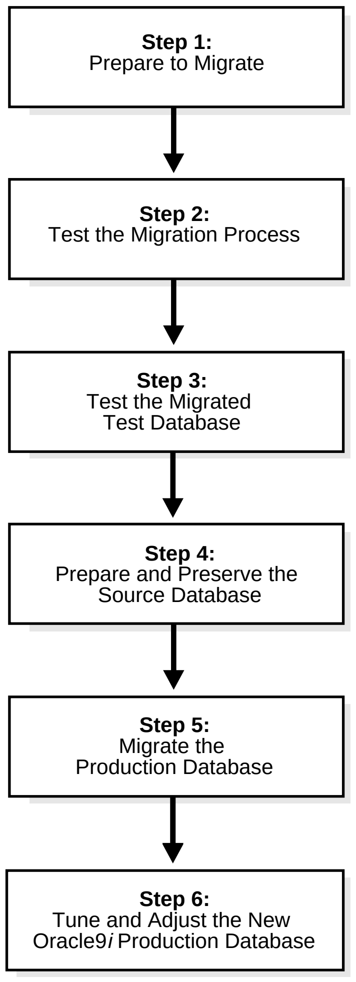
This chapter covers the following topics:

Overview of Migration Steps

Before you perform a database migration, you should understand the major steps in the migration process. These major steps apply to all operating systems, with the possible exception of a few operating system-specific details identified in your operating system-specific Oracle documentation.

Note: The rest of this chapter describes migration. If you plan to perform an operation other than migration, such as upgrading or downgrading, then you can proceed to the appropriate chapter for the operation.

Figure 2–1 Major Migration Steps


Step 1:
Prepare to Migrate
Step 2:
Test the Migration Process
Step 3:
Test the Migrated Test Database
Step 4:
Prepare and Preserve the Source Database
Step 5:
Migrate the Production Database
Step 6:
Tune and Adjust the New Oracle9*i* Production Database

Careful planning and use of Oracle9*i* tools can ease the process of migrating a database to Oracle9*i*. You can use one of the following migration methods to migrate your database:

- The Oracle Data Migration Assistant is the easiest way to migrate an entire database.
- The Migration utility is more complicated to use but provides more control over the process of migrating an entire database.
- Export/Import and SQL copy utilities enable piecemeal migration of parts of a database.

The following sections contain a brief outline of the major steps shown in Figure 2–1. The purpose of these descriptions is to familiarize you with the major steps in the migration process. For detailed instructions, refer to the appropriate chapters and sections later in this book.

Step 1: Prepare to Migrate

- Become familiar with the features of the Oracle9*i* database. See *Oracle9i Database New Features* for an overview of these features.
- Decide which migration method to use, based on considerations involving the current production database, your migration objectives, and the behavior and capabilities of available migration methodologies.
- Estimate and secure the system resources required for the migration.
- Develop a plan for testing the migration with an Oracle9*i* test database and a plan for testing the migrated Oracle9*i* production database.
- Prepare a backup strategy so that you can recover quickly from any unexpected problems or delays.

Step 2: Test the Migration Process

- Perform a test migration using a test database. The test migration should be conducted in an environment created for migration testing and should not interfere with the actual production database.

Step 3: Test the Migrated Test Database

- Perform the tests you planned in Step 1 on the version 7 or version 6 test database and on the version 7 or version 6 test database that was migrated to Oracle9*i*.
- Compare results, noting anomalies between test results on the version 7 or version 6 test database and on the migrated Oracle9*i* database.
- Investigate ways to correct any anomalies you find and then implement the corrections.
- Repeat Step 1, Step 2, and the first parts of Step 3, as necessary, until the test migration is completely successful and works with any required applications.

Chapter 3, "Preparing to Migrate", provides detailed information about Steps 1 through 3.

Step 4: Prepare and Preserve the Source Database

- Prepare the current production database as appropriate to ensure that its migration to Oracle9*i* will be successful.
- Schedule the downtime required for backing up and migrating the version 7 or version 6 production database to Oracle9*i*.
- Perform a full backup of the current production database. This step is required only if either the Oracle Data Migration Assistant or the Migration utility is used for the migration.

Step 5: Migrate the Production Database

- Migrate the version 7 or version 6 production database to Oracle9*i*.
- After the migration, perform a full backup of the production database and other post-migration tasks.

Chapter 4 describes Steps 4 and 5 using the Migration utility; Chapter 5 describes Steps 4 and 5 using the Oracle Data Migration Assistant; and Chapter 6 describes Steps 4 and 5 using the Export/Import utilities. Chapter 8 describes the backup procedure after the migration and other post-migration tasks.

> **See Also:** Appendix G, "Migration and Compatibility for Replication Environments", if you are migrating a database system that has Oracle replication installed.

Step 6: Tune and Adjust the New Production Database

- Tune the new Oracle9*i* production database. The Oracle9*i* production database should perform as good as, or better than, the Oracle database prior to migration. Chapter 8 describes these tuning adjustments.
- Determine which new features of the Oracle9*i* database you want to use and update your applications accordingly.
- Develop new database administration procedures as needed.
- Do not migrate production users to the Oracle9*i* database until all applications have been tested and operate properly. Chapter 10 describes considerations for updating applications.

During migration, multi-versioning can be a useful feature because you can keep multiple copies of the same database on one computer system. You can use the existing version as your production environment while you test the new version.

Role of the Database Administrator During Migration

Typically, the database administrator (DBA) is responsible for ensuring the success of the migration process. The DBA is usually involved in each step of the process, except for steps that involve testing applications on the migrated database.

The specific DBA duties typically include the following:

- meeting with everyone involved in the migration process and clearly defining their roles during migration
- performing test migrations
- scheduling the test and production migration process
- performing backups of the version 7 or version 6 production database
- completing the production database migration
- performing backups of the newly migrated Oracle9*i* production database

Role of the Application Developer During Migration

The application developer is responsible for ensuring that applications designed for the version 7 or version 6 database work correctly with the migrated Oracle9*i* database. Application developers often test applications against the migrated Oracle9*i* database and decide which new features of Oracle9*i* should be used.

Before migrating the Oracle7 production database, the DBA or application developer should install an Oracle9*i* test database. Then, the application developer can test and modify the applications, if necessary, until they work with their original (or enhanced Oracle9*i*) functionality.

The following references provide information about identifying differences in the migrated Oracle9*i* database that could affect particular applications. Application developers can use these differences to guide modifications to existing applications.

- Chapter 9, "Compatibility and Interoperability", describes compatibility and interoperability issues that may result because of differences in releases of Oracle.
- Chapter 10, "Upgrading Your Applications", describes the changes required to enable existing applications (that access an Oracle7 database) to access an Oracle9*i* database and provides guidance for upgrading Oracle7 applications to take advantage of Oracle9*i* functionality.

- Appendix B, "Changes to Initialization Parameters", lists new, renamed, and obsolete initialization parameters in version 8 and Oracle9*i*.
- Appendix C, "Changes to Static Data Dictionary Views", lists new, changed, and obsolete static data dictionary views in version 8 and Oracle9*i*.
- Appendix D, "Changes to Dynamic Performance Views", lists new, changed, and obsolete dynamic performance views (V$ views) in version 8 and Oracle9*i*.
- Appendix E, "New Internal Datatypes and SQL Functions", lists new internal datatypes and SQL functions added in version 8 and Oracle9*i*.
- Appendix F, "Migration and Compatibility for Oracle Net Services", provides instructions for migrating and upgrading SQL*Net and Net8.
- Appendix G, "Migration and Compatibility for Replication Environments", provides instructions for a database system that has Oracle replication installed.
- *Oracle9i Database New Features* describes the new features available in Oracle9*i*.
- *Oracle9i Real Application Clusters Concepts* and *Oracle9i SQL Reference* contain descriptions of changes and new Oracle9*i* functionality.
- *Oracle9i Application Developer's Guide - Fundamentals*, *Oracle9i Application Developer's Guide - Large Objects (LOBs)*, and *Oracle9i Application Developer's Guide - Advanced Queuing* provide information about planning and implementing applications.

Oracle9*i* includes features that aid in upgrading existing applications to Oracle9*i*, for example:

- Net8 and SQL*Net V2 support communication between Oracle versions.
- The programming interface is unchanged between Oracle versions.
- Oracle's backward compatibility accommodates small incompatibilities between different versions and releases.

3

Preparing to Migrate

This chapter covers the steps that must be completed before you migrate a production database. This chapter covers in detail Steps 1 through 3 of the migration process, which were outlined in Chapter 2.

This chapter covers the following topics:

- Prepare to Migrate
- Test the Migration Process
- Test the Migrated Test Database

The information in this chapter is generic and applies generally to Oracle7 and version 6 production databases.

See Also:

- Appendix F, "Migration and Compatibility for Oracle Net Services", for information about migrating SQL*Net to Oracle Net.
- Appendix G, "Migration and Compatibility for Replication Environments", if you are migrating a database system that has Oracle replication installed.
- Some aspects of migration are operating system-specific. See your operating system-specific Oracle documentation for additional information about preparing to migrate.

Prepare to Migrate

Complete the following tasks to prepare to migrate:

- Become Familiar with the Features of the New Database
- Choose a Migration Method
- Assess System Requirements vs. Resources Available
- Choose an Oracle Home Directory for the New Release
- Avoid Common Migration Problems
- Prepare a Backup Strategy
- Develop a Testing Plan

Become Familiar with the Features of the New Database

Before you plan the migration process, become familiar with the new features of the Oracle9*i* database. *Oracle9i Database New Features* is a good starting point for learning the differences between release 9.0.1, release 8.1, release 8.0, and release 7.3. Also, check specific books in the Oracle documentation library to find information about new features for a certain component; for example, see *Oracle9i Real Application Clusters Concepts* for changes in Oracle9*i* Real Application Clusters.

Note: Oracle9*i* training classes are an excellent way to learn how to take full advantage of the functionality available with Oracle9*i*. Connect to the following web page for more information:

```
http://education.oracle.com
```

Choose a Migration Method

Choose one of the following methods to migrate your database to Oracle9*i*:

- Use the Migration utility to migrate an Oracle7 database to Oracle9*i*. See your operating system-specific Oracle documentation for information about the earliest release that the Migration utility can migrate on your operating system. In general, the Migration utility supports migrations of the last 7.3 release and higher databases on your operating system. The exact maintenance release number of the last 7.3 release varies from operating system to operating system.

  The Migration utility is a command-line utility for migration of a complete database from Oracle7 to Oracle9*i*. It changes datafile headers but leaves actual data unchanged. It does not copy data.

- Use the Oracle Data Migration Assistant to migrate an Oracle7 database to Oracle9*i*. See your operating system-specific Oracle documentation for information about the earliest release that the Oracle Data Migration Assistant can migrate on your operating system. Typically, the earliest release supported by the Oracle Data Migration Assistant is the same as the earliest release supported by the Migration utility.

  The Oracle Data Migration Assistant has a graphical user interface (GUI) for migration or upgrade of a complete database. It changes datafile headers but leaves actual data unchanged. It does not copy data.

- Perform a full or partial export of an Oracle7 (or version 6) source database, followed by a full or partial import into an Oracle9*i* target database.

  Export/Import can migrate parts of the database. Export/Import leaves datafile headers and actual data unchanged, and makes a new copy of the data.

- Copy data from a source database into an Oracle9*i* database using the `COPY` command or the `AS` clause of the `CREATE TABLE` statement.

  Data copying can migrate parts of the database. Data copying leaves datafile headers and actual data unchanged, and makes a new copy of the data.

Table 3–1 summarizes the advantages of each of these methods. Table 3–2 summarizes the disadvantages of each of these methods.

Table 3–1 Advantages of Different Migration Methods

Migration Utility	Oracle Data Migration Assistant	Export/Import	Data Copying
Automatic, requires minimal interaction by the DBA when compared with Export/Import or data copying. Relatively fast, whatever the size of the database, because the data dictionary objects are the only objects that are changed. Imposes essentially no limit on the size of the database it can migrate. Requires relatively little additional disk space, when compared with other migration methods. Provides more control over the migration process than the Oracle Data Migration Assistant.	Guides you through the migration with an easy-to-use GUI. Automatic, requiring minimal interaction by the DBA when compared with Export/Import or data copying. Relatively fast, whatever the size of the database, because the data dictionary objects are the only objects that are changed. Imposes essentially no limit on the size of the database it can migrate. Requires relatively little additional disk space, when compared with other migration methods. Can be used for release-to-release upgrades, for example, upgrading from release 8.1.7 to release 9.0.1.	Can migrate version 6 databases to Oracle9*i*, as well as any production Oracle7 database, including release 7.0. Can migrate specific parts of a database. Can be used to downgrade between versions of Oracle, for example, downgrading from Oracle9*i* to Oracle7. Can be used for release-to-release upgrade or downgrade operations, for example, upgrading from release 8.1.7 to release 9.0.1. Datafiles can be defragmented, and migrated data compressed, to improve performance. A database can be restructured with modified or new tablespaces, or by table partitioning. Can be used to migrate to a different operating system and hardware platform.	Datafiles can be defragmented, and migrated data compacted, to improve performance. A database can be restructured with modified or new tablespaces. Can migrate version 6 databases to Oracle9*i*, as well as any production Oracle7 database, including release 7.0. Can migrate specific parts of a database. Can be used for release-to-release upgrade or downgrade operations, for example, upgrading from release 8.1.7 to release 9.0.1. Can be used to migrate to a different operating system and hardware platform.

Table 3–2 Disadvantages of Migration Methods

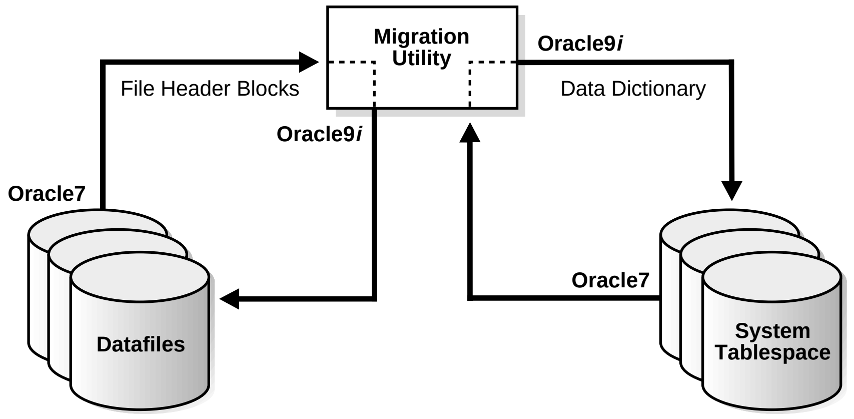
Migration Utility	Oracle Data Migration Assistant	Export/Import	Data Copying
Performs only Oracle7 to Oracle9*i* migrations, and cannot downgrade back to Oracle7. Cannot perform direct migrations on release 7.0, 7.1, and 7.2 databases, nor on databases below a specific 7.3 release. The 7.3 release requirement is operating system-specific. Cannot perform release-to-release upgrades, for example, cannot upgrade from release 8.1.7 to release 9.0.1. Cannot migrate selected parts of a database; migrates only the entire database. Cannot migrate to a different operating system or hardware platform.	Provides less flexibility than other methods because the migration process is highly automated. The GUI covers only the most essential migration choices. Provides less control over the migration than other methods. Cannot downgrade back to Oracle7. Cannot perform direct migrations on release 7.0, 7.1, and 7.2 databases, nor on databases below a specific 7.3 release. The specific 7.3 release requirement is operating system-specific. Cannot migrate selected parts of a database; migrates only the entire database. Cannot migrate to a different operating system or hardware platform. Cannot migrate systems with Oracle Parallel Server installed.	Extremely slow except for very small databases. Time required increases with the amount of data and use of LONG datatypes. Very large databases of several gigabytes may take many hours, and terabyte databases may take days. Requires large amounts of disk space for copying data into export files.	Extremely slow except for very small databases. Time required increases with the amount of data and use of LONG datatypes. Very large databases of several gigabytes may take many hours, and terabyte databases may take days. Requires that both source and target databases be available at once during copying operations.

The following sections describe each of the migration methods in detail, covering the relative amounts of time and space the methods require and the situations in which the methods are most appropriate.

Migration Utility

The Migration utility is a command-line utility that converts files and structures in the Oracle7 source database to Oracle9*i* format, changing only the file headers and, if necessary, the definitions of the data in the files. The Migration utility does not change the data portions of the datafiles, nor their format or content.

Figure 3–1 Migration Utility



The primary advantages of using the Migration utility are speed and relative ease of use. The Migration utility takes significantly less time than Export/Import, and its use entails a standardized series of specific, easy steps. In addition, the time required to migrate a database with the Migration utility depends less on the size of the database than on the number of objects in the data dictionary.

The Migration utility is especially useful for quickly migrating an entire source database. Unlike Export/Import, the Migration utility cannot selectively migrate specific datafiles. However, for databases with large amounts of data, large datatypes, and some other Oracle7 features, Export/Import may not be feasible, and the only practical options may be either the Migration utility or the Oracle Data Migration Assistant.

The Migration utility requires only enough temporary space in the `SYSTEM` tablespace to hold both the Oracle7 (source) and Oracle9*i* (target) data dictionaries simultaneously.

The Migration utility converts the entire database, including database files, rollback segments, and the control files. At any point before actually migrating the Oracle7 database, you can open and access data with the Oracle7 instance. However, after the Migration utility has migrated the Oracle7 source database to Oracle9*i*, you can go back to Oracle7 only by restoring a full backup of the Oracle7 source database.

The Migration utility cannot perform direct migrations on release 7.0, 7.1, and 7.2 databases, nor on databases below a specific 7.3 release. In general, the Migration utility supports migrations of the last 7.3 release and higher databases on your operating system. The exact maintenance release number of the last 7.3 release varies from operating system to operating system.

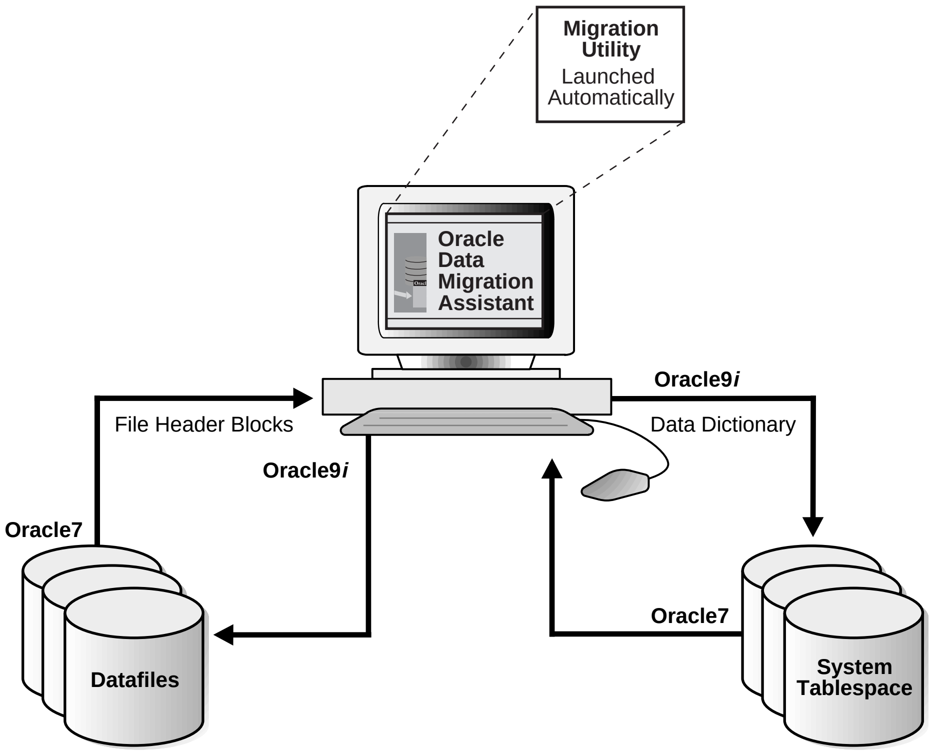
If you are using a release below the release supported by the Migration utility on your operating system, then you first must migrate or upgrade your database to a supported Oracle7 release before using the Migration utility to migrate to Oracle9*i*. See your operating system-specific Oracle documentation for information about the earliest release supported by the Migration utility on your operating system.

> **See Also:** Chapter 4, "Migrating from Oracle7 Using the Migration Utility", for detailed information about using the Migration utility.

Oracle Data Migration Assistant

The Oracle Data Migration Assistant provides a user-friendly, graphical user interface (GUI) that guides you through the migration process. The Oracle Data Migration Assistant calls the Migration utility and runs it in the background, which means that you avoid running the Migration utility manually from a command-line.

Figure 3–2 Oracle Data Migration Assistant



The primary advantage of the Oracle Data Migration Assistant is that it is easy to use. Because the Oracle Data Migration Assistant calls the Migration utility, most of the advantages and disadvantages of the Migration utility also apply to the Oracle Data Migration Assistant. The section "Choosing Between the Oracle Data Migration Assistant and the Migration Utility" on page 3-9 provides information about the differences between the Oracle Data Migration Assistant and the Migration utility.

See Also: Chapter 5, "Migrating from Oracle7 Using the Oracle Data Migration Assistant", for detailed information about using the Oracle Data Migration Assistant.

Choosing Between the Oracle Data Migration Assistant and the Migration Utility

When choosing between the Oracle Data Migration Assistant and the Migration utility, consider these differences:

- The Oracle Data Migration Assistant provides a GUI that guides you through the migration process, while the Migration utility is a command-line utility. The Oracle Data Migration Assistant also provides extensive online help. Therefore, the Oracle Data Migration Assistant is easier to use than the Migration utility.
- The Oracle Data Migration Assistant is less flexible than the Migration utility. To avoid complexity, the Oracle Data Migration Assistant automates many of the steps in the migration process. In contrast, if you use the Migration utility, then you must perform each step manually, which enables you to make adjustments during the migration process if they are necessary. Of course, the migration process usually takes longer if you use the Migration utility because you are performing more steps manually, and you must make more decisions and complete more steps during the process.
- The Oracle Data Migration Assistant performs all of the steps in a migration each time it is run. Therefore, if you have to exit the Oracle Data Migration Assistant for any reason during a migration, then you must restore the backup of your Oracle7 database and start over from the beginning. You may need to exit the Oracle Data Migration Assistant if, for example, you receive an unexpected error. In contrast, if you use the Migration utility, then there are several clearly-defined major steps in the process. If you have to abort a particular step, then you can start at the end of the previous step, without starting over from the very beginning.
- The Oracle Data Migration Assistant automatically removes obsolete initialization parameters from the initialization parameter file. See "Initialization Parameters Obsolete in Oracle9i" on page B-3 for lists of the obsolete parameters that are removed by the Oracle Data Migration Assistant. In contrast, the Migration utility does not alter your initialization parameter file, and you should remove the obsolete parameters manually if you use the Migration utility.

- The Oracle Data Migration Assistant does not support migration of systems with Oracle Parallel Server installed. However, you can use the Migration utility to migrate systems with Oracle Parallel Server installed.

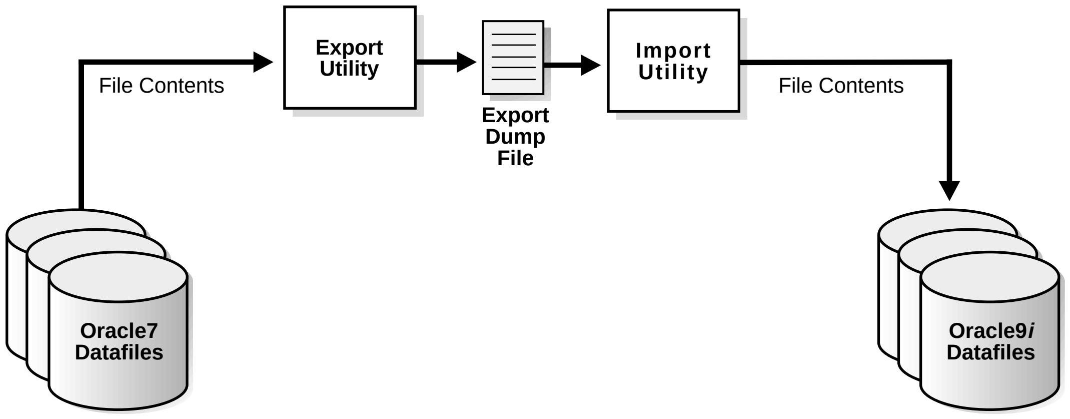
> **Note:** Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

In general, if you prefer a graphical user interface (GUI) over a command-line interface, and you like highly automated processes with few choices, then use the Oracle Data Migration Assistant. If, on the other hand, you prefer a command-line interface over a GUI, and you like to have more control over the migration process, then use the Migration utility.

Export/Import

Unlike the Migration utility, the Export/Import operation physically copies data in the source database to a new database. The source database's Export utility copies specified parts of the source database into an export file. Then, the Oracle9*i* Import utility loads the exported data into the new Oracle9*i* database. However, the new Oracle9*i* target database must already exist before the export file can be migrated into it.

Figure 3–3 Export/Import



The following sections highlight aspects of Export/Import that may help you to decide whether to use Export/Import for migrating your database.

> **See Also:** Chapter 6, "Migrating Using Export/Import", and *Oracle9i Database Utilities*, for more information about using Export/Import for migration.

Export/Import Effects on Migrated Databases The Export/Import method of migration does not change the source database, which enables the source database to remain available throughout the migration process. However, if a consistent snapshot of the database is required (for data integrity or other purposes), then the source database must run in restricted mode or must otherwise be protected from changes during the export procedure. Because the source database can remain available, you can, for example, keep an existing Oracle7 production database running while the new Oracle9*i* database is being built at the same time by Export/Import. During this migration, to maintain complete database consistency, changes to the data in the Oracle7 database cannot be permitted without the same changes to the data in the Oracle9*i* database.

The Export/Import method can also be used to upgrade or downgrade a database. For example, the transformation of an Oracle9*i* database back into an Oracle7 database can be accomplished using Export/Import.

Most importantly, the Export/Import operation results in a completely new database. Although the source database ultimately contains a copy of the specified data, the migrated database may perform differently from the original source database. For example, although Export/Import creates an identical copy of the migrated database, other factors, such as disk placement of data and unset tuning parameters, may cause unexpected performance problems.

As a result of data defragmentation, database restructuring by the DBA, and the new Oracle9*i* software, expect changes in the following areas:

- performance
- data growth patterns
- shared resource usage
- data dictionary size
- object organization

Careful planning, expert implementation, and rigorous testing are required to take advantage of the possible positive effects of Export/Import on the database; otherwise, the database changes may create problems. If the database was restructured during migration, and the migrated database behaves differently, then it may be difficult to determine the causes of the differences.

Export/Import Benefits Data migration by Export/Import offers the following benefits:

- Defragments the data - you can compress the imported data to improve performance.
- Restructures the database - you can create new tablespaces or modify existing tables, tablespaces, or partitions to be populated by imported data.
- Enables the migration of specified database objects or users - you can import only the objects, users, and other items that you wish.
- Serves as a backup archive - you can use a full database export as an archive of the source database.

Export/Import Limitations Data migration by Export/Import has the following limitations:

- Migrating a database by Export/Import requires an expert DBA. The combination of required planning and complicated execution typically requires multiple stagings and a great deal of practice before the final migration can be attempted.
- For a large database, a full database Export/Import can require a substantial amount of temporary storage space for the export dump file.
- You may need to partition the export into multiple jobs if the operating system does not support a file size as large as the database.
- Export/Import creates an entirely new database. To keep the source database in place, and to import its export dump file/data into an Oracle9*i* target database, you must create target data files on the system *before* you import.
- Making multiple changes to the database at the same time, such as migrating to Oracle9*i* and defragmenting/restructuring the database simultaneously, can hinder troubleshooting.
- To keep data in the source database and the target database synchronized during migration, either prohibit any data change in the source or make full provisions to mirror the changes in the migrated data in the target database.

Time Requirements for Export/Import Migrating an entire database by using Export/Import can take a long time, especially compared to using the Migration utility or the Oracle Data Migration Assistant. Therefore, you may need to schedule the migration during non-peak hours or make provisions for propagating to the new target database any changes that are made to the source database during the migration.

The time and system resources (particularly disk space) required for Export/Import migration depend on DBA skill, database size, and the type of data to be migrated, particularly the number, size, and type of indexes that must be rebuilt.

For example, a relatively simple 6-gigabyte, Oracle7 database was migrated to Oracle9*i* using the Migration utility in about an hour. The same Oracle7 database was exported, producing a single 2-gigabyte export dump file. To import that one export dump file took 20 hours. The complete migration using the steps described in "Migrate the Source Database Using Export/Import" on page 6-3 took two days.

Consider the following factors related to the extended time required to migrate a database by Export/Import:

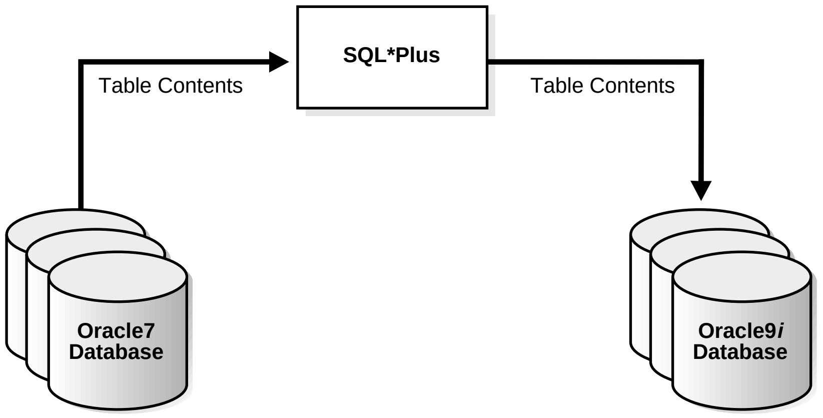
- Migration time easily exceeds the non-peak or off-production hours available in a typical daily schedule. Therefore, making the database unavailable for production tasks for the duration of the migration process might be impractical.
- If you make the source database read-only or do not allow changes to be made in it after the export, then applications will be unavailable until after the import and final migration steps are completed.
- For larger databases, consider operating parallel export streams to reduce the time and optimize the process.

Data Definition Conversion by Oracle9*i* Import When importing data from an earlier version, the Oracle9*i* Import utility makes appropriate changes to data definitions as it reads earlier versions' export dump files. That is, it handles dump files produced by the Export utilities of Oracle version 6, version 7, and version 8. If the export source database is earlier than version 6, then the source database *must* first be upgraded to at least version 6 before the export is performed.

Copying Data

You can copy data from one Oracle database to another Oracle database using database links. For example, you can copy data from a source database table to a target database table with the SQL*Plus COPY statement, or you can create new tables in a target database and fill the tables with data from the source database by using the `INSERT INTO` statement and the `CREATE TABLE ... AS` statement.

Figure 3–4 Copying Data



Copying data and Export/Import offer the same advantages for migration. Using either method, you can defragment data files and restructure the database by creating new tablespaces or modifying existing tables or tablespaces. In addition, you can migrate only specified database objects or users.

Copying data, however, unlike Export/Import, enables the selection of specific rows of tables to be placed into the target database. Copying data is thus a good method for migrating only part of a database table. In contrast, using Export/Import to migrate data from Oracle7 to Oracle9*i*, you can migrate only entire tables.

For example, to create a new table (NEW_EMP) that contains a subset of the data in an existing table (EMP@V7DB, only the employees in departments 10 and 20), you can use the following SQL statement:

```
CREATE TABLE new_emp AS
     SELECT empno, ename, job, mgr, hiredate, sal, comm, deptno
     FROM emp@v7db WHERE deptno IN (10, 20);
```

Copying data requires less disk space and memory buffer space for migration than Export/Import because copying data requires only that the source database and the target database both are online. There is no need to allocate large amounts of extra space for temporary files or for export dump files.

The SQL*Plus `COPY` command is useful for working with large clustered tables. Further, the SQL*Plus `COPY` command can move portions of the cluster in parallel using Oracle Net (or SQL*Net). For more information about copying data from one database to another, refer to the `CREATE TABLE` statement in the *Oracle9i SQL Reference* and to the `COPY` command in the *SQL*Plus User's Guide and Reference*.

Assess System Requirements vs. Resources Available

Estimate the system resources required for successful migration. Different migration methods may result in different resource requirements; therefore, if you are not certain of the method you want to use, then complete an estimate for each potential method of migrating the existing database to Oracle9*i*.

Consider the following factors in your estimates:

- configuration requirements for both the operating system and hardware
- the size of the existing production database
- possible size adjustments to your database associated with implementing Oracle9*i*

Oracle9*i* binaries may require as much as three times the disk space required by Oracle7 binaries. This threefold increase can require special attention on large batch systems (which may generate dozens or hundreds of executables). The space required for executables also depends on the options you choose for the Oracle9*i* environment, such as the following:

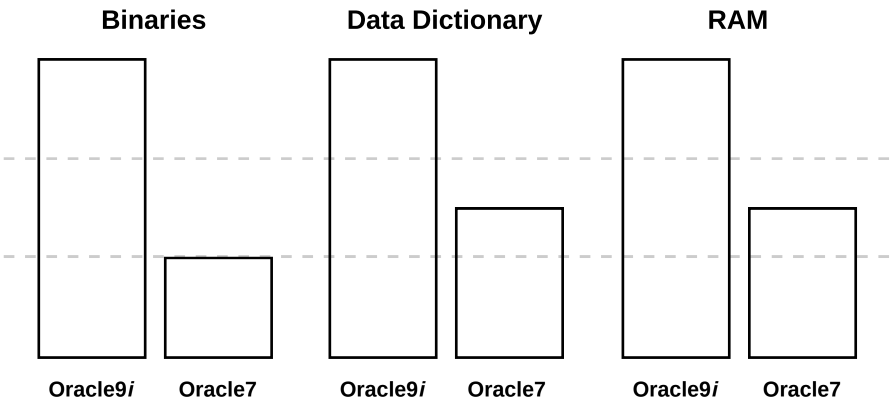
- Oracle9*i* Real Application Clusters (see *Oracle9i Real Application Clusters Installation and Configuration*)
- Oracle Net or SQL*Net use (see *Oracle Net Services Administrator's Guide*)

In addition, the Oracle9*i* data dictionary may require as much as double the space of the Oracle7 data dictionary in the `SYSTEM` tablespace. If you plan to use the Migration utility, then you can estimate space requirements for the `SYSTEM` tablespace by running the Migration utility with the `CHECK_ONLY` option.

Also, Oracle9*i* may require up to twice as much RAM as Oracle7. The amount of RAM required also depends on the options you choose for the Oracle9*i* environment.

Figure 3–5 illustrates the differences in system requirements between Oracle7 and Oracle9*i*.

Figure 3–5 System Requirements for Migration



After you have chosen a migration method and estimated your requirements, secure the necessary resources for a successful migration.

> **See Also:** Your operating system-specific Oracle documentation for detailed information about system requirements.

Assess Memory Requirements for Concurrent Access

The memory size of an Oracle9*i* system depends on concurrent access and the way in which concurrent access is accomplished. Oracle9*i* supports the following connect options:

Option 1: Use local connections in dedicated server architecture (also called "two-task common"). Set this option as it was set in Oracle7.

Option 2: Use remote connections through SQL*Net. Set this option as it was set in Oracle7.

Option 3: Use multithreaded shared servers for local and remote connections. After migrating, set initialization parameters for this option as specified in the *Oracle9i Database Administrator's Guide*.

Option 4: Use transaction processor (TP) monitors.

Option 1 requires more memory than Option 2 or Option 3. With Option 1, if both client application and its Oracle server (or shadow) process reside on the same computer, memory is required for both. For example, 100 client application processes connected to Oracle9*i* results in 100 additional Oracle server processes on the system, totaling 200 in all.

With **Option 2**, only the Oracle processes reside on the system, and the client processes are connected remotely. Thus, you need to consider only to the size of the Oracle server processes and the size of the available memory.

Option 3, using multithreaded server architecture, enables the processes of several local or remote client processes to connect to a single dispatcher process, instead of having a dedicated Oracle shadow process. While not designed as a performance enhancement, multithreaded server configuration enables more concurrent connections on an Oracle9*i* server, thereby improving throughput. Multiple clients can connect to a single dispatcher, so the memory utilization for concurrent user connections decreases.

See Also: *Oracle9i Database Concepts, Oracle9i Database Administrator's Guide*, and *Oracle9i Database Performance Guide and Reference*, for more information on the multithreaded server feature of Oracle9*i*.

Option 4, use of TP monitors, is an alternative for systems requiring a high number of users (greater than several hundred) all performing OLQP / OLTP type transactions. Such transactions are usually short-lived and do not require the user to make a direct connection to the database. All transactions are performed with messages routed by the TP (transaction processor) monitor service. The TP layer provides named services and coordinates service requests with various DBMS systems, including Oracle.

Note: The requirements for using TP monitors vary greatly and are beyond the scope of this manual. Please consult the appropriate TP monitor vendor for system requirements.

In summary, you can estimate system memory requirements, for a single system, by considering the following factors:

- the average number of open cursors and cursors that may cause sorts for a given Oracle application session
- the average size of the Oracle shadow processes that will include open cursors and sort areas
- the peak number of concurrent users on the system
- the average memory size of the Oracle front-end application

Choose an Oracle Home Directory for the New Release

You must choose an Oracle home directory for the new Oracle9*i* release that is separate from the Oracle7 Oracle home directory. You cannot install the Oracle9*i* software into the same Oracle home directory that you used for Oracle7.

Using separate installation directories enables you to keep your Oracle7 software installed along with the Oracle9*i* software. This method enables you to test the migration process on an Oracle7 test database before replacing your production environment entirely.

Avoid Common Migration Problems

You can save time by eliminating common migration problems before you migrate your database. Common problem areas include those in the following table:

Table 3–3 Common Migration Problems **(Page 1 of 2)**

Issues That Affect Migration	Description
1. Running out of space*	Oracle9*i* binaries may require as much as three times the disk space required by Oracle7 binaries. This requirement may cause you to run out of disk space during migration. It is very important that you read "Assess System Requirements vs. Resources Available" on page 3-16 to understand requirements before you migrate. In addition, Chapter 4 and Chapter 5 discuss requirements in more detail. During migration, the data dictionary requires 50% more space to hold both Oracle7 and Oracle9*i* data dictionaries. Actual requirements can be verified by running the Migration utility with the `CHECK_ONLY` option.
2. Duration of migration is unrelated to database size*	The time it takes to migrate is not dependent on the size of the database, but on the number of objects in the data dictionary. For example, actual migration for a 3 1/2 GB database with 25,473 objects on a Sun E6000 with 20 CPUs, with datafiles stripped on the file system on 128 KB slices, can take 1 1/2 hours. Remember to allow extra time for backing up and restoring the database in case of problems.
3. Avoiding problem areas*	Check for usage of ROWIDs in both user columns and application code (including triggers & packaged procedures). These may need to be converted using the DBMS_ROWID package. See Chapter 12, "Migration Issues for Physical Rowids". Because the format for rowids is different in version 8, the old rowids are invalid and must be converted Check the names of any Oracle7 database objects (for example, tables and columns) that use names that are key words or reserved words for Oracle9*i*. Usage of key words and reserved words can cause unexpected failures during migration. See *Oracle9i SQL Reference* for lists of key words and reserved words. Certain version 7 initialization parameters are obsolete in Oracle9*i*. Remove all obsolete parameters from the Oracle7 initialization parameter file that start an Oracle9*i* instance. Obsolete parameters may cause errors if used with an Oracle9*i* database. Also, alter any parameter whose syntax has changed in Oracle9*i*. See Appendix B, "Changes to Initialization Parameters", for lists of new, changed, and obsolete parameters.
4. Compatibility	Make sure that all Oracle product versions, operating system versions, and third-party software versions are certified against Oracle9*i*. See the Oracle documentation for your operating system for information.

Table 3–3 Common Migration Problems **(Page 2 of 2)**

Issues That Affect Migration	Description
5. Invalid objects and lost statistics*	Migration leaves all objects (packages, triggers, views, and so on) invalid except for tables. All other objects must be made valid again by recompilation. You can either do this manually, or you can do this automatically as the objects are first accessed. The latter will of course slow down initial access. All estimated or calculated statistics are lost during migration. These need to be recalculated to ensure proper functionality of the optimizer. Some bitmapped indexes will become invalidated. Check all bitmapped indexes in the DBA_INDEXES table and recreate any that are marked as status unusable after migration.
6. Editing the Windows registry	If you are using a Windows operating system, and you edit the registry for any reason during the migration process, then you need to restart your computer.
7. Read-only tablespace issues*	Oracle7 read-only tablespaces are readable by Oracle9*i* and do not require any conversion. But to prevent Oracle9*i* rowid conversions from taking place every time a table is accessed, the tablespaces in read-only mode should be made read-write. Perform full table scans on all tables in the tablespace. After the full table scans are complete, you can put the tablespaces in read-only mode again.
8. The point of no return*	You can return the database to an Oracle7 version up until the `ALTER DATABASE CONVERT` statement is run. If a failure occurs during `ALTER DATABASE CONVERT` (when it is converting the physical file headers of the datafiles), then you must restore the database from backup and rerun the migration. Do not open the database between running the migration and executing the `ALTER DATABASE CONVERT` statement.
9. Preventing large restores*	To avoid restoring the entire database due to any failures during the `ALTER DATABASE CONVERT` statement, put all tablespaces, except `SYSTEM` and `ROLLBACK` into read-only or offline normal mode. This causes the `ALTER DATABASE CONVERT` statement to only convert the datafile headers for `SYSTEM` and `ROLLBACK`. If any errors occur, then you need only restore the datafiles for `SYSTEM` and `ROLLBACK` and rerun the migration. If the migration is successful, then the headers for the rest of the datafiles will be converted when they are changed to read-write or online.
10. Testing	Most migration problems can be avoided if a test migration is performed first. Performing a test migration helps raise any problems that can occur and provides a basis for the amount of time it will take to migrate your production database. See "Test the Migration Process" on page 3-26 for more information.

* These issues apply only to the Migration utility and the Oracle Data Migration Assistant. They do not apply to Export/Import and data copying.

Prepare a Backup Strategy

The ultimate success of your migration depends heavily on the design and execution of an appropriate backup strategy. To develop a backup strategy, consider the following questions:

- How long can the production database remain inoperable before business consequences become intolerable?
- What backup strategy should be used to meet your availability requirements?
- Are backups archived in a safe, offsite location?
- How quickly can backups be restored (including backups in offsite storage)?
- Have recovery procedures been tested successfully?

Your backup strategy should answer all of these questions and include procedures for successfully backing up and recovering your database.

> **See Also:** *Oracle7 Server Administrator's Guide* for Oracle7 databases and *Oracle9i User-Managed Backup and Recovery Guide* for Oracle9*i* databases.

Develop a Testing Plan

You need a series of carefully designed tests to validate all stages of the migration process. Executed rigorously and completed successfully, these tests ensure that the process of migrating the production database is well understood, predictable, and successful. Perform as much testing as possible before migrating the production database. Do not underestimate the importance of a test program.

> **Caution:** Failing to test rigorously before migration is risky and may lead to unpredictable results.

The testing plan must include the following types of tests:

Migration Testing

Migration testing entails planning and testing the migration path from the source database to the migrated database, whether you use the Migration utility, the Oracle Data Migration Assistant, Export/Import, or other data-copying methods to migrate the production database data to the target database. These methods are discussed in Chapter 4, "Migrating from Oracle7 Using the Migration Utility", Chapter 5, "Migrating from Oracle7 Using the Oracle Data Migration Assistant", and Chapter 6, "Migrating Using Export/Import".

Regardless of the migration method you choose, you must establish, test, and validate a migration plan.

Minimal Testing

Minimal testing entails moving all or part of an application on the source database to the target database and running the application without enabling any new target database features. Minimal testing is a very limited type of testing that may not reveal potential issues that may appear in a "real-world" production environment. However, minimal testing will reveal any application startup or invocation problems immediately.

Functional Testing

Functional testing is a set of tests in which new and existing functionality of the system are tested after migration. Functional testing includes all components of the RDBMS system, networking, and application components. The objective of functional testing is to verify that each component of the system functions as it did before migrating and to verify that new functions are working properly.

Integration Testing

Integration testing examines the interaction of each component of the system. Consider the following factors when you plan your integration testing:

- Pro*C/C++ applications running against the target database instance should be tested to ensure that there are no problems with the new software.
- GUI interfaces should be tested with other components.
- Subtle changes in the target database, such as datatypes, data in the data dictionary (additional rows in the data dictionary, object type changes, and so forth) can have an effect all the way up to the front-end application, regardless of whether the application is directly connected to the Oracle9*i* instance or not.
- If the connection between two components involves Oracle Net or SQL*Net, then those connections should also be tested and stress tested.

Performance Testing

Performance testing of a target database compares the performance of various SQL statements in the target database with the statements' performance in the source database. Before migrating, you should understand the performance profile of the application under the source database. Specifically, you should understand the calls the application makes to the database kernel.

For example, if you are using Oracle Parallel Server, and you want to measure the performance gains realized from using cache fusion when you migrate to the new release, then make sure you record your system's statistics before migrating. For cache fusion, record the statistics from the `V$SYSSTAT`, `V$LOCK_ACTIVITY`, and `V$LOCK_CLASS_PING` views. Doing so enables you to compare pre- and post-cache fusion performance statistics.

For best results, run the SQL scripts `utlbstat.sql` and `utlestat.sql` to collect `V$SYSSTAT` statistics for a specific period. Use a collection timeframe that most consistently reflects peak production loads with consistent transaction activity levels. To obtain data from `V$LOCK_ACTIVITY` and `V$LOCK_CLASS_PING`, use a `SELECT *` statement at the beginning and end of the statistics collection period. Repeat this process after cache fusion is running on the new release and evaluate your system's performance as described in *Oracle9i Real Application Clusters Deployment and Performance*.

> **See Also:** *Oracle9i Database Performance Guide and Reference* for information about tuning. To thoroughly understand the application's performance profile under the source database, enable the SQL trace facility and profile with TKPROF.

Volume and Load Stress Testing

Volume and load stress testing tests the entire migrated database under high volume and loads. Volume describes the amount of data being manipulated. Load describes the level of concurrent demand on the system. The objective of volume and load testing is to emulate how a production system might behave under various volumes and loads.

Volume and load stress testing is crucial, but is commonly overlooked. Oracle Corporation has found that customers often do not conduct any kind of volume or load stress testing. Instead, customers often rely on benchmarks that do not characterize business applications. Benchmarks of the application should be conducted to uncover problems relating to functionality, performance, and integration, but they cannot replace volume and load stress testing.

After you migrate the source database, you should test the data to ensure that all data is accessible and that the applications function properly. You also should determine whether any database tuning is necessary. If possible, you should automate these testing procedures.

The testing plan should reflect the work performed at the site. You should test the functionality and performance of all applications on the source production databases. Gather performance statistics for both normal and peak usage.

Specific Pre-Migration and Post-Migration Tests

Include the following tests in your testing plan:

- timing tests
- data dictionary growth observations
- database resource usage observations, such as rollback and temporary segment usage

Collecting this information will help you compare the source database with the migrated target database.

Use EXPLAIN PLAN on both the source and target databases to determine the execution plan Oracle follows to execute each SQL statement. Use the `INTO` clause to save this information in tables.

After migrating, you can compare the execution plans of the migrated database with the execution plans of the source database. If there is a difference, then execute the statement on the migrated database and compare the performance with the performance of the statement executed on the source database.

See Also: *Oracle9i Database Performance Guide and Reference* for more information about EXPLAIN PLAN.

Test the Migration Process

Create a test environment that will not interfere with the current production database. Your test environment will depend on the migration method you have chosen:

- If you plan to use the Migration utility or the Oracle Data Migration Assistant, then create a test version (typically a subset) of the source database to test migration.
- If you plan to use Export/Import, then export and import small test pieces of the actual production source database.

Practice migrating the database using the test environment. The best migration test, if possible, is performed on an exact copy of the database to be migrated, rather than on a downsized copy or test data.

Caution: Do not migrate the actual production database until after you successfully migrate a test subset of this database and test it with applications, as described in the next step.

Make sure you upgrade any OCI and precompiler applications that you plan to use with your Oracle9*i* database. Then, you can test these applications on a sample Oracle database before migrating your production database. See "Upgrading Precompiler and OCI Applications" on page 10-3 for more information.

Test the Migrated Test Database

Perform the planned tests on the source database and on the test database that you migrated to Oracle9*i*. Compare the results, noting anomalies. Repeat the test migration as many times as necessary.

Test the newly migrated Oracle9*i* test database with existing applications to verify that they operate properly with a migrated Oracle9*i* database. You also might test enhanced functionality by adding features that use the available Oracle9*i* functionality. However, first make sure that the applications operate in the same manner as they did in the source database.

See Also: Chapter 10, "Upgrading Your Applications", for more information on using applications with Oracle9*i*.

4

Migrating from Oracle7 Using the Migration Utility

This chapter guides you through the process of migrating an Oracle7 database to Oracle9*i* using the Migration utility. This chapter covers the following topics:

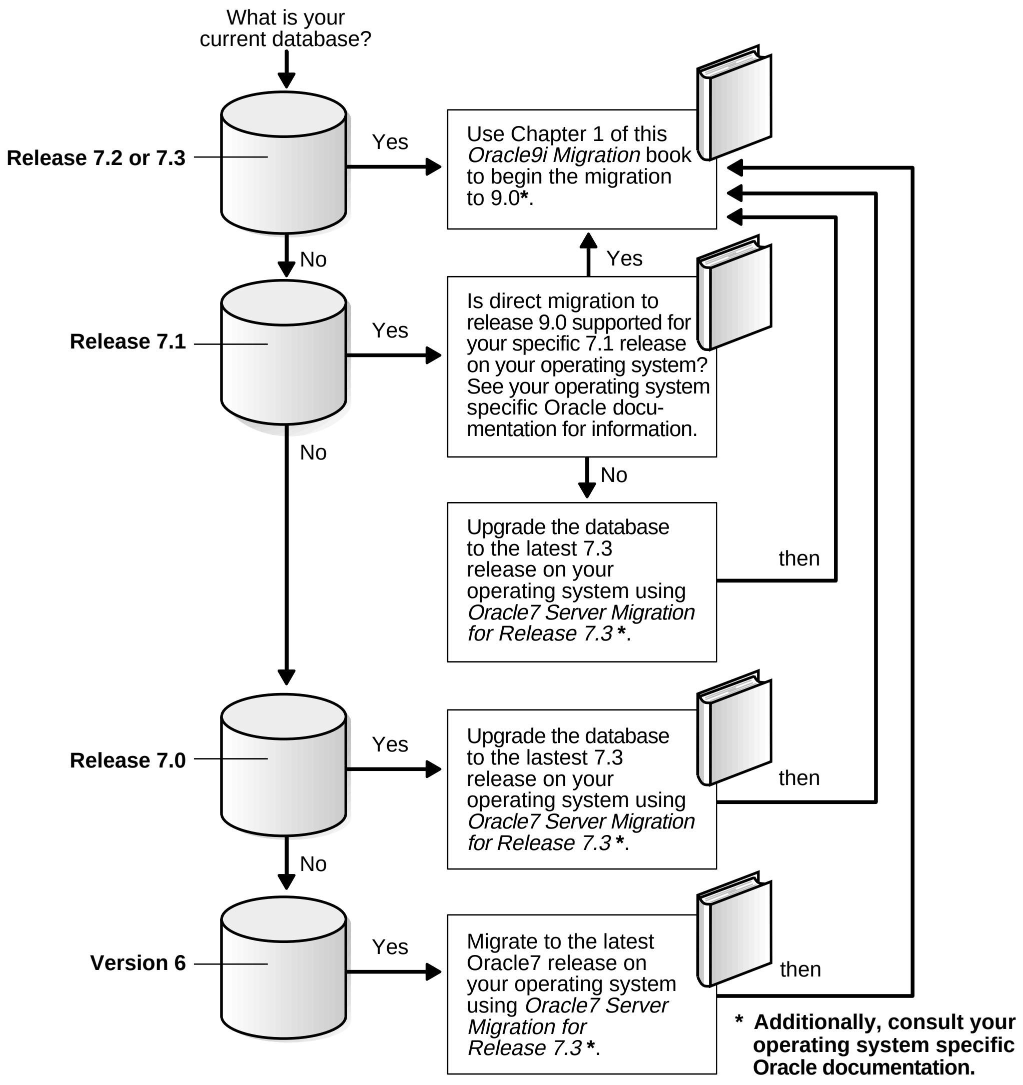
- Documentation Roadmap for Using the Migration Utility
- Overview of Migration Using the Migration Utility
- System Considerations and Requirements
- Prepare the Oracle7 Source Database for Migration
- Install the Release 9.0.1 Oracle Software
- Review Migration Utility Command-Line Options
- Migrate the Oracle7 Source Database
- Troubleshooting Errors During Migration
- Abandoning the Migration

See Also: Some aspects of migration are operating system-specific. See your operating system-specific Oracle documentation for additional information about migrating.

Documentation Roadmap for Using the Migration Utility

Figure 4–1 is a roadmap that specifies the documentation you should use to migrate your database to release 9.0.1 based on your current release of Oracle.

Figure 4–1 Documentation Roadmap for Using the Migration Utility


What is your current database?
Release 7.2 or 7.3
Yes
Use Chapter 1 of this *Oracle9i Migration* book to begin the migration to 9.0*****.
No
Release 7.1
Yes
Is direct migration to release 9.0 supported for your specific 7.1 release on your operating system? See your operating system specific Oracle documentation for information.
Yes
No
Upgrade the database to the latest 7.3 release on your operating system using *Oracle7 Server Migration for Release 7.3* *****.
then
No
Release 7.0
Yes
Upgrade the database to the lastest 7.3 release on your operating system using *Oracle7 Server Migration for Release 7.3* *****.
then
No
Version 6
Yes
Migrate to the latest Oracle7 release on your operating system using *Oracle7 Server Migration for Release 7.3* *****.
then
*** Additionally, consult your operating system specific Oracle documentation.**

Overview of Migration Using the Migration Utility

The Migration utility converts the data dictionary and structures of an Oracle7 database into Oracle9*i* format. To migrate the database, you first install the Oracle9*i* software and run the Migration utility on the Oracle7 database. Then, you execute a series of `ALTER DATABASE` statements on the new Oracle9*i* database and run the `u0703040.sql` conversion script.

The completion of these procedures results in the conversion of the following Oracle7 structures into structures that can be used by Oracle9*i*:

- Data files (file headers only)
- Data dictionary
- Control files
- Rollback segments

Outline of the Migration Process

The following sections provide an outline of the migration process:

In the Oracle7 Environment

- You run the Oracle9*i* Migration utility, which creates and populates a new data dictionary based on the data dictionary of the Oracle7 database, and also creates a binary file based on the control file of the Oracle7 database. This binary file is called the convert file.

Note: You can run the Oracle9*i* Migration utility multiple times (without opening the database in Oracle9*i*) and still be able to return to the Oracle7 database. However, running the Migration utility automatically eliminates the Oracle7 database catalog views (see "Abandoning the Migration" on page 4-35).

In the Oracle9*i* Environment

- You execute an `ALTER DATABASE CONVERT` statement, which creates a new control file based on the convert file generated by the Migration utility, converts all online datafile headers to Oracle9*i* format, and mounts the Oracle9*i* database.

The file headers of offline datafiles and read-only tablespaces are not updated during migration. The file headers of offline datafiles are converted later when they are brought online, and the file headers of read-only tablespaces are converted if and when they are made read-write sometime after migration; however, they never have to be made read-write.

- You execute an `ALTER DATABASE OPEN RESETLOGS` statement, which automatically converts all objects and users defined in the new dictionary to Oracle9*i* specifications, and converts all rollback segments to Oracle9*i* format.

  If a source database rollback segment is in a tablespace that is offline when the Oracle9*i* database is opened, then the rollback segment is not converted immediately to Oracle9*i* database format. Instead, the rollback segment is converted the first time the tablespace is brought online in Oracle9*i*.

- You run the database conversion scripts. The primary conversion script is the `u0703040.sql` script. This script creates and alters certain system tables and drops the `MIGRATE` user. It also runs the `catalog.sql` and `catproc.sql` scripts, which create the system catalog views and all the necessary packages for using PL/SQL.

  Other conversion scripts perform the necessary operations to convert specific components to the current release. For example, the `catrep.sql` script is one of the conversion scripts for Oracle Replication.

Using the Migration Utility

This section contains important considerations for using the Migration utility.

Start with an Oracle7 Database Supported by the Migration Utility

In general, the Migration utility supports migrations of the last 7.3 release and higher databases on your operating system. The exact maintenance release number of the last 7.3 release varies from operating system to operating system.

For example, on some operating systems, the Migration utility can migrate only release 7.3.4 and higher databases, and cannot migrate a release lower than release 7.3.4 (such as release 7.0, 7.1, and 7.2). If your database release number is lower than the release supported by the Migration utility on your operating system, then upgrade or migrate the database to the required release.

See Also:

- Your operating system-specific Oracle documentation for information about the earliest release that is supported by the Migration utility on your operating system.
- Release 7.3 of *Oracle7 Server Migration* for instructions about migrating or upgrading the database to the required release. Then, use this manual to migrate to Oracle9*i*.

Downgrading

Downgrading is the process of transforming an existing Oracle database into a previous version or release. The Migration utility *cannot* transform an Oracle9*i* database back into Oracle7. In some situations, you can use another facility to downgrade, such as using Export/Import, restoring from backups, and possibly using other functions.

See Also: Chapter 13 and Chapter 15 for information about downgrading.

System Considerations and Requirements

The following sections discuss system considerations and requirements for using the Migration utility.

Space Requirements

Oracle9*i* binaries may require as much as three times the disk space required by Oracle7 binaries. This requirement may cause you to run out of disk space during migration. If you are installing Oracle9*i* onto a computer system that already has Oracle7 installed, then ensure that you have enough hard disk space and RAM for both databases. You need to add the system requirements for Oracle9*i* server and Oracle7 server to determine the total system requirements.

The Migration utility requires relatively little temporary space. It needs only enough extra room in the SYSTEM tablespace to hold the new Oracle9*i* data dictionary simultaneously with the existing Oracle7 data dictionary.

The space required to hold an Oracle data dictionary depends on how many objects are in the database. Typically, a new Oracle9*i* data dictionary requires double the space that its Oracle7 source data dictionary required. If necessary, add space to the SYSTEM tablespace.

In addition, running the conversion scripts (such as the u0703040.sql script) to complete the migration may require more space in the SYSTEM tablespace and in the rollback segments. Insufficient space results in an "unable to extend" warning when you run a conversion script. The exact amount of space required to run the conversion scripts varies depending on the number of objects in the database. If you encounter "unable to extend" warnings when you run a conversion script, then try increasing the SYSTEM tablespace and the rollback segments; then, rerun the script.

> **See Also:** See your operating system-specific installation documentation for detailed information about system requirements.

If you need to add more space to the SYSTEM tablespace, then issue a statement similar to the following, substituting the appropriate directory path and name for the new datafile and the amount of space you need to add:

```
ALTER TABLESPACE system
   ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
   SIZE 50M
   AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;

    ALTER ROLLBACK SEGMENT rb
       STORAGE (MAXEXTENTS UNLIMITED);
```

Block Size Considerations

The value of DB_BLOCK_SIZE (an initialization parameter in the initialization parameter file) in the Oracle7 database and in the migrated Oracle9*i* database *must* be the same. Oracle9*i* requires a minimum block size of 2048 bytes (2 KB). Above this amount, integer multiples of your operating system's physical block size are acceptable. However, multiples of 2 KB, especially powers of 2—that is, 2 KB, 4 KB, 8 KB, 16 KB—provide for the most robust operation.

Make sure the Oracle9*i* block size setting meets the following criteria:

- Matches the Oracle7 setting.
- Is at least 2048 bytes (2 KB). The Oracle9*i* Migration utility displays an error message if the Oracle7 block size is less than 2 KB.
- Is an integer-multiple of your operating system's physical block size, preferably a multiple of 2 KB.

Considerations for SQL*Net

There are many issues relating to SQL*Net that you must consider when you migrate your database to Oracle9*i*, not the least of which is deciding whether you will migrate to Oracle Net Services.

See Also: Appendix F, "Migration and Compatibility for Oracle Net Services" for information about these issues and for instructions on migrating from SQL*Net to Oracle Net.

Considerations for Replication Environments

You can migrate an Oracle7 replication environment to Oracle9*i*. Oracle7 sites can coexist and run successfully with version 8 and Oracle9*i* sites within the replication environment. However, take special care to accommodate the various replication features implemented on each system.

See Also: Appendix G, "Migration and Compatibility for Replication Environments" for detailed instructions about migrating systems using replication features

Migrating a System with Oracle Parallel Server Installed

If you are migrating a system with Oracle Parallel Server installed, then most of the actions described in this chapter should be performed on only one node of the system. So, perform the actions described in this chapter on only one node unless instructed otherwise in a particular step. Support for coexistence of different versions of the database is operating system-specific for Oracle Parallel Server.

Note: Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

Considerations for Migrating from ConText to Oracle Text

See *Oracle Text Application Developer's Guide* for information about migrating from ConText to Oracle Text.

Migrating to a Different Operating System

The Migration utility *cannot* migrate a database to a computer system that has a different operating system. For example, it cannot migrate a database from Oracle7 on Solaris to Oracle9*i* on Windows NT. However, you normally can use Export/Import to migrate a database to a different operating system.

Note: Starting with release 8.1, a change in word-size is supported during the migration process. A change in word size involves switching between 32-bit and 64-bit architecture within the same operating system. See "Changing Word-Size" on page 1-12 for more information.

Character Set Considerations

In Oracle9*i*, the SQL NCHAR datatypes (NCHAR, NVARCHAR, and NCLOB) will be limited to the Unicode character set encoding (UTF8 and AL16UTF16) only. When you migrate to Oracle9*i*, the value of the National Character Set of the migrated database is set to AL16UTF16.

AL24UTFFSS Character Set Desupported

The AL24UTFFSS Unicode character set has been desupported in Oracle9*i*. AL24UTFFSS was introduced in Oracle7 as the Unicode character set supporting the UTF-8 encoding scheme based on the Unicode 1.1 standard, which is now obsolete. In Oracle9*i*, The Unicode database character sets AL32UTF8 and UTF8, include the Unicode enhancements based on the Unicode 3.1 standard.

The migration path for existing AL24UTFFSS databases is to upgrade your database character set to UTF8 prior to upgrading to Oracle9*i*. As with all migrations to a new database character set, Oracle Corporation recommends you use the Character Set Scanner for data analysis before attempting to migrate your existing database character set to UTF8.

> **See Also:** *Oracle9i Globalization Support Guide* for more information about the Character Set Scanner

Distributed Database Considerations

When migrating from Oracle7 in a distributed database configuration, make sure that no pending transactions are in the `DBA_2PC_PENDING` data dictionary view before migrating the database. Otherwise, when you open the database after migration using the `ALTER DATABASE RESET LOGS` statement and a transaction is pending, you will encounter an error.

If there are any pending transactions, then resolve them before you migrate using the SQL commands `COMMIT FORCE` or `ROLLBACK FORCE`.

Prepare the Oracle7 Source Database for Migration

Complete the following steps before you migrate your Oracle7 database to Oracle9*i*:

1. If your database release number is lower than the release supported by the Migration utility on your operating system, then upgrade or migrate the database to a supported release.

   > **See Also:** "Start with an Oracle7 Database Supported by the Migration Utility" on page 4-4 for more information.

2. If the Procedural Option is not installed, then use your Oracle7 installation media to install it. See your operating system-specific Oracle documentation for instructions.

   If you are not sure whether the Procedural Option is installed, then you can check by starting Server Manager.

   The following is an example of the messages you will see when Server Manager starts:

   ```
   Oracle Server Manager Release 2.3.3.0.0 - Production

   Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.

   Oracle7 Server Release 7.3.4.0.0 - Production
   ```

```
With the distributed, replication, parallel query, Parallel Server
and Spatial Data options
PL/SQL Release 2.3.4.0.0 - Production
```

The messages you see may be slightly different, based on the options you have installed and their release numbers. If you see "PL/SQL" in the messages, as in the last line in the preceding example, then the Procedural Option is installed. Otherwise, it is not installed.

3. Make sure all datafiles and tablespaces are either online or offline normal.

   To determine whether any datafiles require recovery, issue the following SQL statement:

   ```
   SELECT * FROM v$recover_file;
   ```

   You should see a "0 rows selected" message, which indicates that all datafiles are either online or offline normal. If any datafiles are listed, then you must restore the datafiles before you migrate the database. You can use the `V$DATAFILE` dynamic performance view to find the datafile name based on the datafile number. The Oracle9*i* Migration utility will not proceed, and will display an error, if any datafiles require media recovery.

   Tablespaces that are not taken offline cleanly must be dropped or brought online before migration. Otherwise, these tablespaces will not be available under Oracle9*i* after the migration. Typically, tablespaces that are taken offline by using an `ALTER TABLESPACE OFFLINE IMMEDIATE` or `ALTER TABLESPACE OFFLINE TEMPORARY` statement require media recovery.

   After migration, tablespaces that are offline when you open the Oracle9*i* database remain in Oracle7 database file format. The offline tablespaces can be brought online at any time after migration, and the file headers are converted to Oracle9*i* format at that time. In addition, if you want to avoid large restores in the event of a failure, then you can make all tablespaces except `SYSTEM` and `ROLLBACK` offline normal; then, you can restore only the datafiles for `SYSTEM` and `ROLLBACK` if you need to run another migration.

4. Make sure no user or role has the name `OUTLN`, because this schema is created automatically when you install Oracle9*i*. If you have a user or role named `OUTLN`, then you must drop the user or role and recreate it with a different name.

   To check for a user with the name `OUTLN`, issue the following SQL statement:

   ```
   SELECT username FROM dba_users WHERE username = ’OUTLN’;
   ```

If you do not have a user named `OUTLN`, then zero rows are selected.

To check for a role with the name `OUTLN`, issue the following SQL statement:

```
SELECT role FROM dba_roles WHERE role = 'OUTLN';
```

If you do not have a role named `OUTLN`, then zero rows are selected.

5. Make sure no user or role has the name `MIGRATE`, because the Oracle9*i* Migration utility creates this schema and uses it to replace any pre-existing user or role with this name, and finally drops it from the system.

   To check for a user with the name `MIGRATE`, issue the following SQL statement:

   ```
   SELECT username FROM dba_users WHERE username = 'MIGRATE';
   ```

   If you do not have a user named `MIGRATE`, then zero rows are selected.

   To check for a role with the name `MIGRATE`, issue the following SQL statement:

   ```
   SELECT role FROM dba_roles WHERE role = 'MIGRATE';
   ```

   If you do not have a role named `MIGRATE`, then zero rows are selected.

6. Make sure the `SYSTEM` rollback segment does not have an `OPTIMAL` setting. An `OPTIMAL` setting may cause errors during migration.

   To check the `OPTIMAL` setting for the `SYSTEM` rollback segment, issue the following SQL statement:

   ```
   SELECT a.usn, a.name, b.optsize
       FROM v$rollname a, v$rollstat b
       WHERE a.usn = b.usn AND name = 'SYSTEM';
   ```

   Your output should be similar to the following:

   ```
   USN        NAME                           OPTSIZE
   ---------- ------------------------------ ----------
            0 SYSTEM
   1 row selected.
   ```

   If there is a value in the `OPTSIZE` column, then issue the following SQL statement to set optimal to NULL:

   ```
   ALTER ROLLBACK SEGMENT SYSTEM STORAGE (OPTIMAL NULL);
   ```

You can reset OPTIMAL when migration is complete.

> **See Also:** The troubleshooting information in "OPTIMAL Setting for the SYSTEM Rollback Segment" on page A-5.

7. Increase the maximum number of extents for your SYSTEM rollback segment by altering the MAXEXTENTS parameter in the STORAGE clause of the ALTER ROLLBACK SEGMENT statement (optional).

   The following is an example of the ALTER ROLLBACK SEGMENT statement:

   ```
   ALTER ROLLBACK SEGMENT system
      STORAGE (NEXT 500K MAXEXTENTS 121);
   ```

   You may need more space in the SYSTEM rollback segment to complete the migration successfully. If there is not enough space in your SYSTEM rollback segment, then you may encounter errors when you run the Migration utility in the Oracle7 environment.

8. Shutdown the Oracle7 database cleanly using the SHUTDOWN NORMAL or SHUTDOWN IMMEDIATE statement; do *not* use SHUTDOWN ABORT.

   ```
   SHUTDOWN IMMEDIATE
   ```

   If you are using Oracle Parallel Server, then shut down all instances.

Prepare for Migration on Windows Platforms

In addition to the steps described in the previous section, "Prepare the Oracle7 Source Database for Migration", complete the following steps if you are migrating your database on a Windows platform:

1. Make sure you have the required release of SQL*Net installed.

Migrating From	Required SQL*Net Release
Oracle release 7.3.2	SQL*Net release 2.3.2.1.12 or higher **Note:** If your release of SQL*Net is below release 2.3.2.1.4, you must first install release 2.3.2.1.4 before you can upgrade to release 2.3.2.1.12. Contact Oracle Support Services to obtain the patch that includes release 2.3.2.1.4.
Oracle release 7.3.3	SQL*Net release 2.3.3.0.3 or higher

If the required release of SQL*Net is not installed, complete the following steps to install it:

a. Obtain the year 2000-compliant Oracle Installer for release 7.3 from Oracle Corporation.

b. Start the Oracle Installer you obtained in Step a. Respond to the Oracle Installer screens until you reach the Software Asset Manager screen.

c. At the Software Asset Manager screen, click the From button.

d. Navigate to the drive containing the CD-ROM for the current release of Oracle.

e. Navigate to the appropriate directory on the CD-ROM:

If you are installing SQL*Net release 2.3.2.1.12, navigate to the following directory on the CD-ROM:

```
\patches\sqlnet\232112\nt_x86\install
```

If you are installing SQL*Net release 2.3.3.0.3, navigate to the following directory on the CD-ROM:

```
\patches\sqlnet\23303\nt_x86\install
```

f. Open the `nt.prd` file.

g. Complete the installation.

h. Exit the Oracle Installer.

See Also: Your *Oracle9i installation guide for Windows* for more information about required SQL*Net releases.

Note: If you cannot install the required SQL*Net release, contact Oracle Support Services.

---

2. Ensure all Oracle7 services are stopped, including the service for the Oracle7 database instance.

See Also: Your *Administrator's Guide* for Windows for information about stopping services.

Install the Release 9.0.1 Oracle Software

Complete the following steps to install the release 9.0.1 software:

1. Follow the instructions in your operating system-specific Oracle installation documentation to prepare for installation and start the Oracle Universal Installer.

   If you are migrating a system with Oracle Parallel Server installed, then see *Oracle9i Real Application Clusters Installation and Configuration* for additional installation instructions.

2. At the Welcome screen of the Oracle Universal Installer, click Next. The File Locations screen appears.

   If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, then click the Help button to open the online help.

3. At the File Locations screen, complete the following steps:

   a. Do not change the text in the Source field. This is the location of files for installation.

   b. If there is a Destination Name field, enter the name of a new Oracle home in this field.

   c. Enter the complete path of the Oracle home directory where you want to install the new release in the Destination Path field. Click the Browse button to navigate to the directory.

   Note: You must install the new 9.0.1 release in a new Oracle home that is separate from the Oracle7 release.

   d. Click Next.

   The Available Products screen appears.

4. At the Available Products screen, select the Oracle9*i* server. The Oracle9*i* server is either Oracle9*i* Enterprise Edition or Oracle9*i*, depending on your installation medium. Then, click Next.

5. At the Installation Types screen, choose Custom. Do not choose Standard Edition or Enterprise Edition unless you want to install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom.

Note: Normally, you should not install a starter database if you are migrating an existing database.

After you make your selection, click Next.

If you chose Custom, the Available Product Components screen appears. Complete the following steps:

a. Choose the product components you want to install. Then, click Next.

   Make sure you install Oracle Utilities to install the Migration utility.

   Make sure you install all of the options you installed with the Oracle7 database, assuming you do not want to discontinue use of a particular option. For example, if you installed Oracle Replication in Oracle7, then you should install it in Oracle9*i*.

b. If you are installing Oracle9*i* Real Application Clusters, then, at the Cluster Node Selection screen, select the nodes onto which you want the software installed. Then, click Next.

c. Respond to the remaining screens that enable you to specify your custom installation settings, until you reach the Upgrading or Migrating an Existing Database screen.

6. At the Upgrading or Migrating an Existing Database screen, leave the Upgrade or Migrate an Existing Database check box unselected. Then, click Next.

   If you select the Upgrade or Migrate an Existing Database check box, then the Oracle Data Migration Assistant is started automatically after installation. Because you are following the instructions for migrating the database using the Migration utility, you should not start the Oracle Data Migration Assistant.

Note: The Oracle Data Migration Assistant does not support Oracle9*i* Real Application Clusters migrations.

---

7. At the Create Database screen, select the No option, indicating that you do not want to create a database, because you are migrating an existing database. Then, click Next.

   Complete any remaining screens until you reach the Summary screen. Click the Help button if you need help for a certain screen.

8. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

   When installation has completed successfully, click the Exit button to close the Oracle Universal Installer.

After Installing Oracle9*i* on a Windows Platform

After you successfully install Oracle9*i* on a Windows platform, complete the following steps. If your operating system is UNIX, then skip the rest of this section and go to "Review Migration Utility Command-Line Options" on page 4-17.

1. Shut down and restart your computer.
2. Start the Oracle7 service OracleService*SID*, where *SID* is the instance name. For example, if your *SID* is ORCL, then enter the following at an MS-DOS prompt:

   ```
   C:\> NET START OracleServiceORCL
   ```

   Note: The service might already be started. If it is, a message appears on screen.

3. Set `ORACLE_SID` to the SID of the database you are migrating. For example, if the SID of the database you are migrating is ORCL, then enter the following at an MS-DOS prompt:

   ```
   C:\> SET ORACLE_SID=ORCL
   ```

   Note: Make sure there are no spaces around the equal sign (=).

Review Migration Utility Command-Line Options

The next task in the migration process is running the Oracle9*i* Migration utility. Before you begin that task, review the following command-line options for the Migration utility because you may want to use some of them in your migration. In addition, your operating system-specific Oracle documentation may contain more information about Migration utility command-line options.

CHECK_ONLY	When `true`, the Migration utility performs space use calculations without performing a migration. When `false`, the Migration utility performs both space usage calculations and the migration. This command-line option is mutually exclusive with `NO_SPACE_CHECK`.
DBNAME	Specifies the name of the database to migrate (`DB_NAME` in the initialization parameter file).
MULTIPLIER	Specifies the initial size of the Oracle9*i* i_file#_block# index relative to the Oracle7 i_file#_block# index. For example, `MULTIPLIER=30` triples the initial size when the index is created. If no `MULTIPLIER` command-line option is specified, then the Migration utility uses the i_file#_block# value of 15, creating an index for Oracle9*i* that is 1.5 times larger than the Oracle7 i_file#_block# index.
NEW_DBNAME	Specifies a new name for the migrated database. The default name "`DEFAULT`" should not be used; choose a more meaningful name.
NO_SPACE_CHECK	When `true`, the Migration utility does not perform a space usage check before the migration. When `false`, the Migration utility performs a space usage check before migration. This command-line option is mutually exclusive with `CHECK_ONLY`.
PFILE	Specifies the name of the initialization parameter file. If no `PFILE` command-line option is specified, then the Migration utility uses the default initialization parameter file. **Note:** On UNIX, the pathname must be enclosed by double-quotes escaped by a backslash, for example: `mig PFILE=\"/tmp/mig/pfile\"`
SPOOL	Specifies the filename for the spool output. **Note:** On UNIX, the pathname must be enclosed by double-quotes escaped by a backslash, for example: `mig SPOOL=\"/tmp/mig/spool\"`

Migrate the Oracle7 Source Database

Complete the steps in the following sections to migrate an Oracle7 source database to Oracle9*i* using the Migration utility.

Prepare the Oracle7 Environment for Migration on UNIX Operating Systems

You only need to complete the steps described in this section if you are migrating your Oracle database on a UNIX operating system. If your operating system is

Windows, then skip the rest of this section and go to "Perform Migration Steps in the Oracle7 Environment" on page 4-20.

On UNIX operating systems, the `migprep` utility prepares the Oracle7 environment for migration by copying required migration files from the Oracle9*i* Oracle home to the Oracle7 Oracle home. With your environment variables pointing to the new release 9.0.1 Oracle home, run `migprep` in the following way:

```
migprep new_oracle_home old_oracle_home
```

Where *`new_oracle_home`* is the complete path of the new Oracle9*i* Oracle home directory and *`old_oracle_home`* is the complete path of the old Oracle7 Oracle home directory.

For example, if your new Oracle9*i* Oracle home is `/oracle/product/9.0` and your old Oracle7 Oracle home is `/oracle/product/7.3`, then complete the following steps:

1. At a command prompt, change to the *`ORACLE_HOME`*`/bin` directory in your release 9.0.1 installation.
2. Enter the following to run `migprep`:

   ```
   migprep /oracle/product/9.0 /oracle/product/7.3
   ```

3. Change the following environment variables to point to the Oracle7 directories:
   - `ORACLE_HOME`
   - `PATH`
   - `LD_LIBRARY_PATH`
   - `ORA_NLS32`

   Note: For Oracle Parallel Server, perform this step on all nodes.

4. Set the `ORA_NLS33` environment variable to the following directory in your Oracle7 environment:

   ```
   $ORACLE_HOME/migrate/nls/admin/data
   ```

Perform Migration Steps in the Oracle7 Environment

Complete the following migration steps in the Oracle7 environment. These steps apply to both UNIX systems and Windows platforms.

1. Start Server Manager.
2. Connect to the Oracle7 database as a user with SYSDBA privileges.
3. Startup your Oracle7 database:

   ```
   STARTUP
   ```

4. Make sure the NLS_LANG environment variable is set to the character set you are using for your database.

   To check your character set, issue the following SQL statement:

   ```
   SELECT * FROM v$nls_parameters
      WHERE parameter = 'NLS_LANGUAGE'
         OR parameter = 'NLS_TERRITORY'
         OR parameter = 'NLS_CHARACTERSET';
   ```

   You use all three values returned by this query to set NLS_LANG. For example, suppose your output for the query above is the following:

   ```
   PARAMETER                VALUE
   --------------------  --------------------------
   NLS_LANGUAGE            AMERICAN
   NLS_TERRITORY           AMERICA
   NLS_CHARACTERSET        US7ASCII
   ```

   In this case, set NLS_LANG to the following at a command prompt:

   ```
   AMERICAN_AMERICA.US7ASCII
   ```

   See Also: *Oracle9i Globalization Support Guide* for information about setting NLS_LANG.

5. Make sure you have DBA privileges, which are required to run the Oracle9*i* Migration utility.

   To check if you have DBA privileges, query the DBA_ROLE_PRIVS static data dictionary view. For example, if you are connected as user SYSTEM, then enter the following SQL statement:

   ```
   SELECT * FROM dba_role_privs WHERE grantee = 'SYSTEM';
   ```

You have DBA privileges if 'DBA' is listed in the GRANTED_ROLE column for the user. If you do not have DBA privileges, then connect as a user who does.

6. Make sure no other DBA with `RESTRICTED SESSION` privilege connects to the database while the Migration utility is running. Also, "Normal" users should not connect to the database during migration.
7. Determine the files that you will back up in Step 9 by issuing the following SQL statements:

   ```
   SPOOL v7files.log
   SELECT member FROM v$logfile;
   SELECT name FROM v$datafile;
   SELECT value FROM v$parameter WHERE name = ’control_files’;
   SPOOL OFF
   ```

   The `v7files.log` spool file lists all of the files that you must back up in Step 9.
8. Shutdown the Oracle7 database cleanly using the `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE` statement; do *not* use `SHUTDOWN ABORT`. The Oracle7 source database must be shut down cleanly; therefore, no redo information or uncommitted transactions can remain.

   ```
   SHUTDOWN IMMEDIATE
   ```

   If you are using Oracle Parallel Server, then shutdown all instances.

   Note: If you do not shut down the Oracle7 database before migration starts, then the Migration utility will stop and display an error message.

9. Make a complete backup of your Oracle7 database. Make sure you back up the files listed in the `v7files.log` spool file that you generated in Step 7.

   Caution: If you encounter any problems with the migration, then you will need to restore the database from this backup. Therefore, make sure you back up your database now as a precaution.

   See Also: *Oracle7 Server Administrator’s Guide* for information about backing up your Oracle7 database.

Run the Migration Utility

The steps required to run the Migration utility on UNIX are different than the steps required to run the Migration utility on Windows platforms. Complete the steps in the appropriate section:

- Run the Migration Utility on UNIX
- Run the Migration Utility on a Windows Platform

Run the Migration Utility on UNIX

Complete the following steps to run the Migration utility on a UNIX operating system:

1. Make sure your environment variables are still pointing to the Oracle7 directories.

   See Also: Step 3 on page 4-19 in the section "Prepare the Oracle7 Environment for Migration on UNIX Operating Systems" for detailed information about the environment variables to check.

2. Make sure you have enough space in the `SYSTEM` tablespace (optional).

   A common migration problem is running out of space in the `SYSTEM` tablespace during migration. The Migration utility will not complete the migration unless sufficient space is allocated in the `SYSTEM` tablespace. To determine disk space requirements for a successful migration, run the Oracle9*i* Migration utility with the `CHECK_ONLY` command-line option set to `true` by entering the following at a system prompt:

   ```
   mig CHECK_ONLY = true
   ```

   The `CHECK_ONLY` command-line option causes the Migration utility to assess the amount of disk space required for migration, check the amount of space available, and issue an informational message about the disk space requirements. When the `CHECK_ONLY` command-line option is set to `true`, the Migration utility does not build the Oracle9*i* data dictionary or perform any other migration processing.

   If the `CHECK_ONLY` command-line option shows that you need to add more space to the `SYSTEM` tablespace, then you should add the amount specified by the `CHECK_ONLY` option plus an additional 25 megabytes. The additional 25 megabytes approximates the amount of space required by the migration scripts that you will run later in the migration process.

To add space to the `SYSTEM` tablespace, issue a statement similar to the following, substituting the appropriate directory path and name for the new datafile and the amount of space you need to add:

```
 ALTER TABLESPACE system
    ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
    SIZE 50M
AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;
```

If you add space to the `SYSTEM` tablespace, then remember to shut down the database when you are finished.

3. Run the Migration utility by entering the Migration utility command at a system prompt:

```
mig
```

The command is `mig` unless stated otherwise in your operating system-specific Oracle documentation. Enter `mig` alone to run the Migration utility with a default set of options, or enter `mig` followed by one or more selected options.

> **See Also:** "Review Migration Utility Command-Line Options" on page 4-17 for information about command-line options. Oracle Corporation recommends using the `SPOOL` option, because it makes it easier to check your migration results when the migration is complete.

4. When the Migration utility has completed successfully, change the following environment variables to point to the Oracle9*i* executables:

- `ORACLE_HOME`
- `PATH`
- `LD_LIBRARY_PATH`
- `ORA_NLS33`

If `ORACLE_HOME` points to the Oracle7 executables, then an ORA-00223 error is displayed when you run the `ALTER DATABASE CONVERT` statement later in the migration process, stating "conversion data file is invalid or incorrect version".

> **Note:** For Oracle Parallel Server, perform this step on all nodes.

See Also: Your operating system-specific Oracle9*i* installation documents for information about setting other important environment variables on your operating system.

Run the Migration Utility on a Windows Platform

Complete the following steps to run the Migration utility on a Windows platform:

1. In the new Oracle9*i* Oracle home, run the Migration utility by entering the Migration utility command at the MS-DOS prompt with the PFILE option included:

   ```
   C:\> mig PFILE=ORACLE7_HOME\DATABASE\INIT_PARAM_FILE
   ```

   Replace the *ORACLE7_HOME* variable with the complete path to the Oracle7 Oracle home directory. Also, replace the *INIT_PARAM_FILE* variable with the full name of the initialization parameter file for the Oracle7 database.

   For example, if *ORACLE7_HOME* is C:\ORANT and *INIT_PARAM_FILE* is INITORCL.ORA, then enter the following:

   ```
   C:\> mig PFILE=C:\ORANT\DATABASE\INITORCL.ORA
   ```

   You can enter mig with the PFILE option only to run the Migration utility with a default set of options, or you can enter mig followed by more selected options.

   See Also: "Review Migration Utility Command-Line Options" on page 4-17 for information about command-line options. Oracle Corporation recommends using the SPOOL option, because it makes it easier to check your migration results when the migration is complete.

2. If the *Oracle7 Password* appears when you run the Migration utility, then enter the password for the user logged in with SYSDBA privileges on the Oracle7 database. This prompt appears because the DBA_AUTHORIZATION registry parameter is set improperly or is not set at all.

3. After the Migration utility has run successfully, stop the Oracle7 service OracleService*SID*, where *SID* is the instance name. For example, if your *SID* is ORCL, then enter the following at an MS-DOS prompt:

   ```
   C:\> NET STOP OracleServiceORCL
   ```

4. Delete the Oracle7 service at the MS-DOS command prompt using ORADIM7*x*. The following table lists the MS-DOS command to run for each Oracle7 release:

Oracle7 Release...	Enter at the MS-DOS Command Prompt...
7.1	`C:\> ORADIM71 -DELETE -SID` *SID*
7.2	`C:\> ORADIM72 -DELETE -SID` *SID*
7.3	`C:\> ORADIM73 -DELETE -SID` *SID*

For example, if your Oracle7 release is release 7.3.4 and your *SID* is ORCL, then enter the following MS-DOS command:

```
C:\> ORADIM73 -DELETE -SID ORCL
```

5. Restart your computer.

6. Create the Oracle9*i* database service at the MS-DOS command prompt:

```
C:\> ORADIM -NEW -SID SID -INTPWD PASSWORD -MAXUSERS USERS -STARTMODE AUTO
     -PFILE ORACLE_HOME\DATABASE\INITSID.ORA
```

This syntax includes the following variables:

SID	is the same SID name as the SID of the version 7 database you migrated.
PASSWORD	is the password for the new release 9.0.1 database instance. This is the password for the user logged in with SYSDBA privileges. The -INTPWD option is not required. If you do not specify it, then operating system authentication is used, and no password is required.
USERS	is the maximum number of users who can be granted SYSDBA and SYSOPER privileges.
ORACLE_HOME	is the release 9.0.1 Oracle home directory. Ensure that you specify the full pathname with the -PFILE option, including drive letter of the Oracle home directory.

For example, if your *SID* is ORCL, your *PASSWORD* is TWxy579, the maximum number of *USERS* is 10, and the *ORACLE_HOME* directory is `C:\ORA90`, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -INTPWD TWxy579 -MAXUSERS 10 -STARTMODE AUTO
     -PFILE C:\ORA90\DATABASE\INITORCL.ORA
```

Check the Migration Results

Check the results after running the Migration utility. The Migration utility generates informational messages and echoes its progress as it runs the `migrate.bsq` script. If the Migration utility exits with an ORA- error, then check Appendix A, "Troubleshooting Migration Problems", for information about the error and the actions to perform to resolve the problem.

The Migration utility creates a convert file that contains the information of the Oracle7 control file. Later in the migration process, the convert file is used by `ALTER DATABASE CONVERT` to create a new control file in Oracle9*i*.

The name and location of the convert file are operating system-specific. For example, on a UNIX operating system, the default location is *`ORACLE_HOME/`*`dbs` in the Oracle7 environment, and the default filename in this directory is `conv`*`sid`*`.dbf`, where *sid* is your Oracle7 instance ID. On Windows platforms, the default location is *`ORACLE_HOME`*`\rdbms` in the Oracle9*i* environment, and the default filename in this directory is `convert.ora`.

Caution: Do not open the Oracle7 database, which was shut down by the Oracle9*i* Migration utility. To ensure datafile version integrity, the SCNs in the dictionary, the convert file, and file header must all be consistent when the database is converted to Oracle9*i*. If the Oracle7 database is opened after running the Migration utility, then the SCN check will fail when the database is converted to Oracle9*i*, and an ORA-01211 error will be displayed, stating "Oracle7 datafile is not from migration to Oracle8". Therefore, if the Oracle7 database is opened, then you must rerun the Migration utility, starting at Step 8 on page 4-21.

Preserve the Oracle7 Source Database

After you successfully run the Migration utility, perform a cold backup of the Oracle7 database. This backup serves the following purposes:

- If you wish to return to the Oracle7 database after executing the `ALTER DATABASE CONVERT` statement on Oracle9*i*, then you can restore the backup, start the Oracle7 database, and complete the procedure in "Abandoning the Migration" on page 4-35.
- It can be used as the first Oracle9*i* backup for an Oracle9*i* recovery.

- If an error occurs at Oracle9*i* database convert time (`ALTER DATABASE CONVERT` or `ALTER DATABASE OPEN RESETLOGS`), then you can restore this backup, fix the problems, and continue the conversion process. However, if you restore a backup that was performed before you ran the Migration utility, then you must rerun the Migration utility.

  See Also: *Oracle7 Server Administrator's Guide* for information about performing backup and restore operations on your Oracle7 database.

In addition, perform a backup of the entire Oracle7 software distribution, including the Oracle7 home directory. Make sure the backup includes the following:

- All of the subdirectories
- Control files
- Datafiles and online redo log files (in case any datafiles in the Oracle7 database are lost or unreadable), although these files should not contain any outstanding redo information.
- Parameter files
- Convert file
- Scripts that create objects in the Oracle7 database
- Scripts that could restore the original database, if necessary

Perform Migration Steps in the Oracle9*i* Environment

Complete the following migration steps in the Oracle9*i* environment:

1. Either remove or rename the database's control files, or use the `CONTROL_FILES` initialization parameter to specify new control file names. The `CONTROL_FILES` initialization parameter is typically set in the initialization parameter file, but, if you are using Oracle9*i* Real Application Clusters, then it may be set in the `init`*`db_name`*`.ora` file instead.

   You will issue the `ALTER DATABASE CONVERT` statement in Step 10. This statement automatically creates new control files. If you do not use the `CONTROL_FILES` parameter, then this statement uses the control file names of your pre-migration database (derived from the CONVERT file) and returns an error if the control files already exist. Therefore, in this case, you must remove or rename the control files.

However, if you use the `CONTROL_FILES` initialization parameter to specify new control file names, then the `ALTER DATABASE CONVERT` statement creates the new control files with the names you specify, and you do not need to remove the old control files. For a complete list of your existing control files, check the `v7files.log` spool file you created in Step 9 on page 4-21.

Control files are considerably larger in Oracle9*i* than in Oracle7. For example, Oracle7 control files in the hundreds of kilobytes may expand into tens of megabytes in Oracle9*i*. The larger size in Oracle9*i* results from the storage of more information in the control file, such as backup and tablespace records. This size increase could be important if a control file is on a raw device or if its available disk space is restricted.

Note: The `CONTROL_FILES` initialization parameter specifies one or more names of control files, separated by commas. Oracle Corporation recommends using multiple files on different devices or mirroring the file at the operating system level. See the *Oracle9i Database Administrator's Guide* for more information.

Note: For Oracle9*i* Real Application Clusters, perform this step on all nodes.

2. Copy files that are important for migration to a location outside of the Oracle7 Oracle home:

   a. If your operating system is UNIX, then move or copy the convert file from the Oracle7 Oracle home directory to the Oracle9*i* Oracle home directory. On most UNIX operating systems, the convert file, `conv`*`sid`*`.dbf` (where *sid* is the Oracle9*i* database name), should reside in *`ORACLE_HOME/`*`dbs` in both the Oracle7 and the Oracle9*i* environment.

      On Windows platforms, the convert file, `convert.ora`, should reside in *`ORACLE_HOME`*`\rdbms` in the Oracle9*i* environment. It is automatically placed in this directory by the Migration utility, and you do not need to move it.

   b. If you have a password file that resides within the Oracle7 Oracle home, then move or copy the password file to the Oracle9*i* Oracle home directory.

      The name and location of the password file are operating system-specific; for example, on UNIX operating systems, the default password file is

`ORACLE_HOME/dbs/orapwsid`, but on Windows platforms, the default password file is `ORACLE_HOME\database\pwdsid.ora`. In both cases, *sid* is your Oracle instance ID.

c. If your initialization parameter file resides within the Oracle7 Oracle home, then move or copy it to a location outside of the Oracle7 Oracle home. By default Oracle looks for the initialization parameter file in `ORACLE_HOME/dbs` on UNIX platforms and in `ORACLE_HOME\database` on Windows platforms. The initialization parameter file can reside anywhere you wish, but it should not reside in the Oracle7 Oracle home after you migrate to Oracle9*i*.

d. If the initialization parameter file contains an IFILE (include file) entry that resides within the Oracle7 Oracle home, then move or copy the file specified in the IFILE entry to a location outside of the Oracle7 Oracle home. The file specified in the IFILE entry has additional initialization parameters.

e. If you are using Oracle9*i* Real Application Clusters and your `initdb_name.ora` file resides within the Oracle7 Oracle home, then move or copy the `initdb_name.ora` file to a location outside of the Oracle7 Oracle home.

Note: For Oracle9*i* Real Application Clusters, perform this step on all nodes.

3. Adjust the initialization parameter file in the Oracle9*i* environment for use with Oracle9*i*. Specifically, complete the following steps:

a. Set the `COMPATIBLE` initialization parameter in your initialization parameter file to a valid Oracle9*i* setting, such as 8.1.0 or 9.0.0. Make sure the `COMPATIBLE` parameter is not set to any Oracle7 or 8.0 release, because if it is, then you will not be able to start the Oracle9*i* database. See "Setting the COMPATIBLE Initialization Parameter" on page 9-7 for more information.

b. Remove obsolete parameters and adjust changed parameters. Certain Oracle7 initialization parameters are obsolete in Oracle9*i*. Remove all obsolete initialization parameters from any initialization parameter file that will start an Oracle9*i* instance. Obsolete parameters may cause errors in Oracle9*i*. Also, alter any parameter whose syntax has changed in Oracle9*i*;

refer to Appendix B, "Changes to Initialization Parameters", for lists of new, renamed, and obsolete parameters.

Also, if you are using Oracle9*i* Real Application Clusters, then see *Oracle9i Real Application Clusters Installation and Configuration* for more information about obsolete Oracle9*i* Real Application Clusters initialization parameters.

c. If you are updating materialized views automatically by using the `JOB_QUEUE_PROCESSES` initialization parameter, then set this parameter to 0 (zero) in the initialization parameter file. After migrating your database, you can change the setting for this parameter back to its normal setting.

d. Make sure the `OPTIMIZER_MODE` initialization parameter is set to `choose`. After migrating your database, you can change the setting for this parameter back to its normal setting.

e. If you are using Oracle9*i* Real Application Clusters, then set the `PARALLEL_SERVER` initialization parameter to `false`. You can change it back to `true` after migration is complete.

f. If you are using a Distributed Lock Manager (DLM) on a UNIX operating system, then make sure you set the `LM_LOCKS`, `LM_RESS`, and `LM_PROCS` initialization parameters equal to the lock, resource, and process parameters for the DLM used in Oracle7.

g. Make sure your `DB_DOMAIN` initialization parameter is set properly.

See Also: "The DB_DOMAIN Parameter" on page B-11 for more information about setting this initialization parameter.

h. On Windows platforms, change the `BACKGROUND_DUMP_DEST` and `USER_DUMP_DEST` initialization parameters that point to RDBMS71, RDBMS72, or RDBMS73 to point to the following directories instead (optional):

Initialization Parameter	Change Setting To
`BACKGROUND_DUMP_DEST`	*ORACLE_BASE*`\oradata\`*DB_NAME*
`USER_DUMP_DEST`	*ORACLE_BASE*`\oradata\`*DB_NAME*`\archive`

In the settings, substitute the complete ORACLE_BASE path for *ORACLE_BASE* and substitute the database name for *DB_NAME*.

- **i.** Make sure all path names in the initialization parameter file are fully specified. You should not have relative path names in the initialization parameter file.
- **j.** If the initialization parameter file contains an IFILE entry, then change the IFILE entry in the initialization parameter file to point to the new location you copied it to in Step 2. Then, edit the file specified in the IFILE entry in the same way that you edited the initialization parameter file in sub-steps a to i.
- **k.** If you are using Oracle9*i* Real Application Clusters, then modify the `init`*`db_name`*`.ora` file in the same way that you modified the initialization parameter file in steps a to i.

Make sure you save all of the files you modified after making these adjustments.

Note: For Oracle9*i* Real Application Clusters, perform this step on all nodes.

---

4. If the Oracle9*i* DB_NAME is different from the Oracle7 DB_NAME, then complete the following steps. Otherwise, skip to Step 5.

- **a.** On UNIX operating systems, rename the `conv`*`sid`*`.dbf` file to match the Oracle9*i* DB_NAME. For example, if the Oracle7 DB_NAME is DBMS7 and the Oracle9*i* DB_NAME is DBMS9, then rename the convert file from `convDBMS7.dbs` to `convDBMS9.dbs`. This action is not necessary on Windows platforms.
- **b.** Set the `DB_NAME` initialization parameter in the initialization parameter file to the Oracle9*i* database name.

5. Make sure all online data files are accessible and in the correct directories. If you are using a raw disk, then log files also must be accessible.

6. Change to the *`ORACLE_HOME/`*`rdbms/admin` directory. You should be in the Oracle9*i* Oracle home.

7. Start SQL*Plus.

8. Connect to the database instance as a user with `SYSDBA` privileges.

9. Start an Oracle9*i* database instance without mounting the new Oracle9*i* database:

```
SQL> STARTUP RESTRICT NOMOUNT
```

Caution: Starting the database instance in any other mode might corrupt the database.

You may need to use the `PFILE` option to specify the location of your initialization parameter file.

You may see error messages listing obsolete initialization parameters. If so, then make a note of the obsolete initialization parameters and continue with the migration normally. Then, remove the obsolete initialization parameters the next time you shut down the database (Step 17).

10. Create a new Oracle9*i* database control file and convert the file headers of all online tablespaces to Oracle9*i* format by issuing the following statement:

    ```
    SQL> ALTER DATABASE CONVERT;
    ```

    Successful execution of this statement is the "point of no return" to Oracle7 for this database. However, if necessary, you can restore the Oracle7 database from backups.

    If errors occur during this step, then correct the conditions that caused the errors and rerun the Migration utility. Restart at Step 3 on page 4-20. Otherwise restore the backup you performed after you ran the Migration utility.

    See Also: "Problems at the ALTER DATABASE CONVERT Statement" on page A-6 for information about common errors encountered at this step and the actions required to resolve them.

11. Open the Oracle9*i* database with the following statement:

    ```
    SQL> ALTER DATABASE OPEN RESETLOGS;
    ```

    When the Oracle9*i* database is opened, all rollback segments that are online are converted to the new Oracle9*i* format.

    If you encounter errors when you issue this statement, then start the migration process over from the beginning, ensuring the database is not opened in the Oracle7 environment after the Migration utility completes. Start from the beginning of this chapter, Chapter 4, but make sure you completed all of the pre-migration steps described in Chapter 3.

12. Set the system to spool results to a log file for later verification of success:

```
SQL> SPOOL catoutm.log
```

    If you want to see the output of the scripts you will run on your screen, then you can also issue a `SET ECHO ON` statement:

```
SQL> SET ECHO ON
```

13. Run the Oracle9*i* database conversion script `u0703040.sql`:

```
SQL> @u0703040.sql
```

    The `u0703040.sql` script is the database conversion script for all Oracle7 releases supported by the Migration utility on your operating system. The `u0703040.sql` script creates and alters certain system tables and drops the `MIGRATE` user. It also runs the `catalog.sql` and `catproc.sql` scripts, which create the system catalog views and all the necessary packages for using PL/SQL.

    If you encounter any problems when you run this script, or any of the scripts in the remaining steps, then correct the causes of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

    See Also: "Running Scripts" on page 1-11 for information about the types of errors to look for when you run a script.

14. If the Oracle system has Oracle9*i* Real Application Clusters installed, then run `catclust.sql`:

```
SQL> @catclust.sql
```

15. Run `utlrp.sql` (optional):

```
SQL> @utlrp.sql
```

    The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, and so on. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

    Oracle Corporation highly recommends performing this optional step.

16. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 12; the suggested name was `catoutm.log`.

You should look for errors that alert you to insufficient space, and for errors that alert you that a script failed to run. If you see these types of errors, then your migration may not be completely successful. However, you typically can ignore errors about the failure to alter or drop an object that does not exist.

If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

```
SQL> SET ECHO OFF
```

17. Run `SHUTDOWN` on the Oracle9*i* database:

```
SQL> SHUTDOWN IMMEDIATE
```

Caution: Use `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE`. Do not use `SHUTDOWN ABORT`.

Executing this clean shutdown flushes all caches, clears buffers, and performs other DBMS housekeeping activities. These measures are an important final step to ensure the integrity and consistency of the newly migrated Oracle9*i* database.

The `COMPATIBLE` initialization parameter controls the compatibility level of your database. Set the `COMPATIBLE` initialization parameter in your initialization parameter file based on the compatibility level you want for your migrated database.

See Also: "Setting the COMPATIBLE Initialization Parameter" on page 9-7 for information.

If you encountered a message listing obsolete initialization parameters when you started the database in Step 9, then remove the obsolete initialization parameters from the initialization parameter file now.

18. Complete the procedures described in Chapter 8, "After Migrating or Upgrading the Database".

Caution: If you retain the old Oracle7 software, then never start the migrated database with the old Oracle7 software. Only start the database with the executables in the new Oracle9*i* installation. Also, before you remove the old Oracle7 environment, make sure you relocate any datafiles in that environment to the Oracle9*i* environment. See the *Oracle9i Database Administrator's Guide* for information about relocating datafiles.

Troubleshooting Errors During Migration

Errors may be caused by the following actions or omissions:

- Performing a migration step out of order
- Failing to fulfill the prerequisites for migration
- Encountering an occasional conversion irregularity

See Also: Appendix A, "Troubleshooting Migration Problems" and *Oracle9i Database Error Messages* for information about errors during migration and about corrective action for each error.

Abandoning the Migration

If you backed up your Oracle7 database *before* you ran the Migration utility, then the easiest way to abandon a migration is to restore that backup. However, if you do not have a backup, or if you took the backup after running the Migration utility, then you must complete the procedure described in this section to abandon the migration.

You can run the Oracle9*i* Migration utility multiple times and still return to the Oracle7 database. However, running the Migration utility automatically eliminates the Oracle7 database catalog views. Therefore, to return to the Oracle7 database after running the Migration utility, you must run the Oracle7 `catalog.sql` script to restore the Oracle7 database catalog views.

Note: You cannot use the procedure below to abandon the migration if you already executed the `ALTER DATABASE CONVERT` statement. If you executed this statement and want to return to Oracle7, then complete the procedure in Chapter 15, "Downgrading to an Older Release of Oracle".

To abandon the migration, you generally must restore the Oracle7 database by completing the following steps in the Oracle7 environment:

1. Start the Oracle7 database using SQL*Plus.
2. Drop the `MIGRATE` user:

   ```
   SQL> DROP USER MIGRATE CASCADE;
   ```

3. Rerun `catalog.sql` and `catproc.sql`:

   ```
   SQL> @catalog.sql
   SQL> @catproc.sql
   ```

4. If Server Manager is installed, then run `catsvrmg.sql`:

   ```
   SQL> @catsvrmg.sql
   ```

5. If Oracle Parallel Server is installed, then run `catparr.sql`:

   ```
   SQL> @catparr.sql
   ```

6. If Oracle Replication is installed, then run `catrep.sql`:

   ```
   SQL> @catrep.sql
   ```

Note: The Oracle9*i* Migration utility upgrades release 7.1 and release 7.2 databases to release 7.3. If the original Oracle7 production database was release 7.1 or 7.2 and the migration is run but abandoned before the conversion to Oracle9*i*, then the Oracle7 database will be left with a release 7.3 dictionary. However, in such a case, you do not need to downgrade from release 7.3 to release 7.1or 7.2; your release 7.1 or 7.2 software should work with the data dictionary without the need for further action.

5

Migrating from Oracle7 Using the Oracle Data Migration Assistant

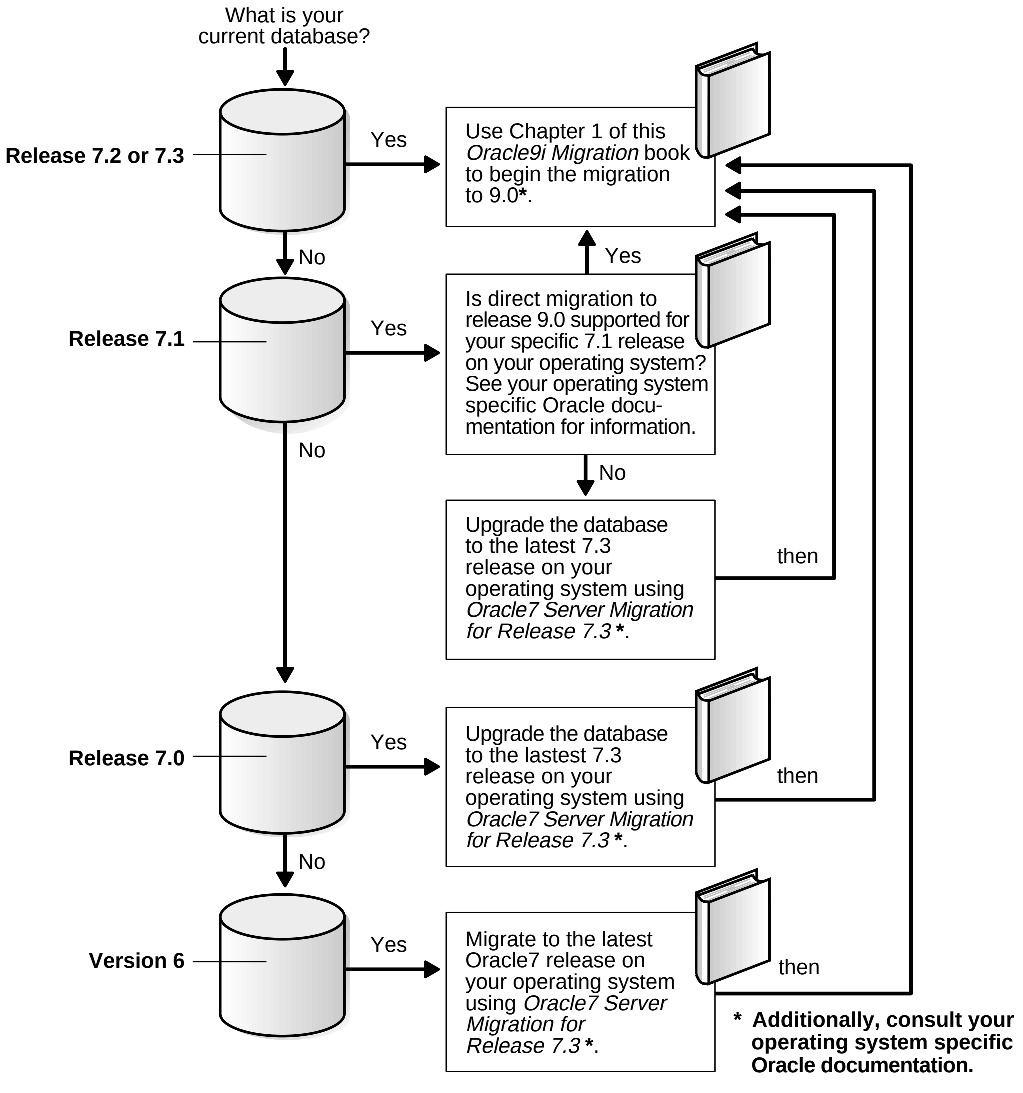
This chapter guides you through the process of migrating an Oracle7 database to Oracle9*i* using the Oracle Data Migration Assistant. This chapter covers the following topics:

See Also: Some aspects of migration are operating system-specific. See your operating system-specific Oracle documentation for additional information about migrating.

Documentation Roadmap for Using the Oracle Data Migration Assistant

Figure 5–1 is a roadmap that specifies the documentation you should use to migrate your database to release 9.0.1 based on your current release of Oracle.

Figure 5–1 Documentation Roadmap for Using the Oracle Data Migration Assistant


What is your current database?
Release 7.2 or 7.3
Yes
Use Chapter 1 of this *Oracle9i Migration* book to begin the migration to 9.0*****.
No
Release 7.1
Yes
Is direct migration to release 9.0 supported for your specific 7.1 release on your operating system? See your operating system specific Oracle documentation for information.
Yes
No
Upgrade the database to the latest 7.3 release on your operating system using *Oracle7 Server Migration for Release 7.3* *****.
then
No
Release 7.0
Yes
Upgrade the database to the lastest 7.3 release on your operating system using *Oracle7 Server Migration for Release 7.3* *****.
then
No
Version 6
Yes
Migrate to the latest Oracle7 release on your operating system using *Oracle7 Server Migration for Release 7.3* *****.
then
*** Additionally, consult your operating system specific Oracle documentation.**

Overview of Migration Using the Oracle Data Migration Assistant

This section contains important considerations for using the Oracle Data Migration Assistant.

Restrictions Related to the Oracle Data Migration Assistant

The following restrictions apply to the Oracle Data Migration Assistant:

- The Oracle Data Migration Assistant does not support the migration of systems with Oracle Parallel Server installed. If you have Oracle Parallel Server installed, then you must use another method to migrate your database, such as the Migration utility or Export/Import.
- The Oracle Data Migration Assistant does not support the migration of systems that use raw devices. If you are using raw devices, then you must use another method to migrate your database, such as the Migration utility or Export/Import.

Note: Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

Start with an Oracle7 Database Supported by the Oracle Data Migration Assistant

In general, the Oracle Data Migration Assistant supports migrations of the last 7.3 release and higher databases on your operating system. The exact maintenance release number of the last 7.3 release varies from operating system to operating system.

For example, on some operating systems, the Oracle Data Migration Assistant can only migrate release 7.3.4 and higher databases, and cannot migrate a release lower than release 7.3.4 (such as release 7.0, 7.1, and 7.2). If your database release number is lower than the release supported by the Oracle Data Migration Assistant on your operating system, then upgrade or migrate the database to the required release.

See Also:

- Your operating system-specific Oracle documentation for information about the earliest release that is supported by the Oracle Data Migration Assistant on your operating system.
- Release 7.3 of *Oracle7 Server Migration* for instructions about migrating or upgrading the database to release 7.3.4. Then, use this manual to migrate to Oracle9*i*.

Downgrading

Downgrading is the process of transforming an existing Oracle database into a previous version or release. The Oracle Data Migration Assistant *cannot* transform an Oracle9*i* database back into Oracle7. In some situations, you can use another facility to downgrade, such as using Export/Import, restoring from backups, and possibly using other functions.

See Also: Chapter 13 and Chapter 15 for information about downgrading.

System Considerations and Requirements

The following sections discuss system considerations and requirements for using the Oracle Data Migration Assistant.

Space Requirements

Oracle9*i* binaries may require as much as three times the disk space required by Oracle7 binaries. This requirement may cause you to run out of disk space during migration. If you are installing Oracle9*i* onto a computer system that already has Oracle7 installed, then ensure that you have enough hard disk space and RAM for both databases. You need to add the system requirements for Oracle9*i* server and Oracle7 server to determine the total system requirements.

The Oracle Data Migration Assistant requires relatively little temporary space. It needs only enough extra room in the SYSTEM tablespace to hold the new Oracle9*i* data dictionary simultaneously with the existing Oracle7 data dictionary.

The space required to hold an Oracle data dictionary depends on how many objects are in the database. Typically, a new Oracle9*i* data dictionary requires double the space that its Oracle7 source data dictionary required. If necessary, add space to the SYSTEM tablespace. The Oracle Data Migration Assistant will not complete the migration unless sufficient space is allocated in the SYSTEM tablespace.

If you need to add more space to the SYSTEM tablespace, then issue a statement similar to the following, substituting the appropriate directory path and name for the new datafile and the amount of space you need to add:

```
ALTER TABLESPACE system
   ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
   SIZE 50M
   AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;
```

Control files are considerably larger in Oracle9*i* than in Oracle7. For example, Oracle7 control files in the hundreds of kilobytes may expand into tens of megabytes in Oracle9*i*. The larger size in Oracle9*i* results from the storage of more information in the control file, such as backup and tablespace records. This size increase could be important if a control file is on a file system where its available disk space is restricted.

Block Size Considerations

The value of DB_BLOCK_SIZE (an initialization parameter in the initialization parameter file) in the Oracle7 database and in the migrated Oracle9*i* database *must* be the same. Oracle9*i* requires a minimum block size of 2048 bytes (2KB). Above this amount, integer multiples of your operating system's physical block size are acceptable. However, multiples of 2KB, especially powers of 2—that is, 2KB, 4KB, 8KB, 16KB—provide for the most robust operation.

Make sure the Oracle9*i* block size setting meets the following criteria:

- Matches the Oracle7 setting.
- Is at least 2048 bytes (2KB). The Oracle Data Migration Assistant displays an error message if the Oracle7 block size is less than 2KB.
- Is an integer-multiple of your operating system's physical block size, preferably a multiple of 2KB.

Considerations for SQL*Net

There are many issues relating to SQL*Net that you must consider when you migrate your database to Oracle9*i*, not the least of which is deciding whether you will migrate to Oracle Net.

See Also: Appendix F, "Migration and Compatibility for Oracle Net Services", for information about these issues and for instructions on migrating from SQL*Net to Oracle Net.

Considerations for Replication Environments

You can migrate an Oracle7 replication environment to Oracle9*i*. Oracle7 sites can coexist and run successfully with version 8 and Oracle9*i* sites within the replication environment. However, take special care to accommodate the various replication features implemented on each system.

See Also: Appendix G, "Migration and Compatibility for Replication Environments", for detailed instructions about migrating systems using replication features.

Considerations for Migrating from ConText to Oracle Text

See *Oracle Text Application Developer's Guide* for information about migrating from ConText to Oracle Text.

Migrating to a Different Operating System

The Oracle Data Migration Assistant *cannot* migrate a database to a computer system that has a different operating system. For example, it cannot migrate a database from Oracle7 on Solaris to Oracle9*i* on Windows NT. However, you can normally use Export/Import to migrate a database to a different operating system.

Note: Starting with release 8.1, a change in word-size is supported during the migration process. A change in word size involves switching between 32-bit and 64-bit architecture within the same operating system. See "Changing Word-Size" on page 1-12 for more information.

Character Set Considerations

In Oracle9*i*, the SQL NCHAR datatypes (`NCHAR`, `NVARCHAR`, and `NCLOB`) will be limited to the Unicode character set encoding (UTF8 and AL16UTF16) only. When you migrate to Oracle9*i*, the value of the National Character Set of the migrated database is set to AL16UTF16.

AL24UTFFSS Character Set Desupported

The AL24UTFFSS Unicode character set has been desupported in Oracle9*i*. AL24UTFFSS was introduced in Oracle7 as the Unicode character set supporting the UTF-8 encoding scheme based on the Unicode 1.1 standard, which is now obsolete. In Oracle9*i*, The Unicode database character sets AL32UTF8 and UTF8, include the Unicode enhancements based on the Unicode 3.1 standard.

The migration path for existing AL24UTFFSS databases is to upgrade your database character set to UTF8 prior to upgrading to Oracle9*i*. As with all migrations to a new database character set, Oracle Corporation recommends you use the Character Set Scanner for data analysis before attempting to migrate your existing database character set to UTF8.

See Also: *Oracle9i Globalization Support Guide* for more information about the Character Set Scanner

Distributed Database Considerations

When migrating from Oracle7 in a distributed database configuration, make sure that no pending transactions are in the `DBA_2PC_PENDING` data dictionary view before migrating the database. Otherwise, when you open the database after migration using the `ALTER DATABASE RESET LOGS` statement and a transaction is pending, you will encounter an error.

If there are any pending transactions, then resolve them before you migrate using the SQL commands `COMMIT FORCE` or `ROLLBACK FORCE`.

Prepare the Oracle7 Source Database for Migration

Complete the following steps before you migrate your Oracle7 database to Oracle9*i*:

1. If your database release number is lower than the release supported by the Oracle Data Migration Assistant on your operating system, then upgrade or migrate the database to a supported release.

   See Also: "Start with an Oracle7 Database Supported by the Oracle Data Migration Assistant" on page 5-4 for more information.

2. Make a complete backup of your Oracle7 database. Determine the files that you must back up by issuing the following SQL statements:

   ```
   SPOOL v7files.log
   SELECT member FROM v$logfile;
   SELECT name FROM v$datafile;
   SELECT value FROM v$parameter WHERE name = 'control_files';
   SPOOL OFF
   ```

   The `v7files.log` spool file lists all of the files that you must back up.

   Caution: If you encounter any problems with the migration, then you will need to restore the database from this backup. Therefore, make sure you back up your database now as a precaution.

   See Also: *Oracle7 Server Administrator's Guide* for information about backing up your Oracle7 database.

3. If the Procedural Option is not installed, then use your Oracle7 installation media to install it. See your operating system-specific Oracle documentation for instructions.

   If you are not sure whether the Procedural Option is installed, then you can check by starting Server Manager.

The following is an example of the messages you will see when Server Manager starts:

```
Oracle Server Manager Release 2.3.3.0.0 - Production

Copyright (c) Oracle Corporation 1994, 1995. All rights reserved.

Oracle7 Server Release 7.3.4.0.0 - Production
With the distributed, replication, and Spatial Data options
PL/SQL Release 2.3.4.0.0 - Production
```

The messages you see may be slightly different, based on the options you have installed and their release numbers. If you see "PL/SQL" in the messages, as in the last line in the preceding example, then the Procedural Option is installed. Otherwise, it is not installed.

4. Make sure all datafiles and tablespaces are either online or offline normal.

   To determine whether any datafiles require recovery, issue the following SQL statement:

   ```
   SELECT * FROM v$recover_file;
   ```

   You should see a "0 rows selected" message, which indicates that all datafiles are either online or offline normal. If any datafiles are listed, then you must restore the datafiles before you migrate the database. You can use the `V$DATAFILE` dynamic performance view to find the datafile name based on the datafile number. You will encounter an error during migration if any datafiles require media recovery.

   Tablespaces that are not taken offline cleanly must be dropped or brought online before migration. Otherwise, these tablespaces will not be available under Oracle9*i* after the migration. Typically, tablespaces that are taken offline by using an `ALTER TABLESPACE OFFLINE IMMEDIATE` or `ALTER TABLESPACE OFFLINE TEMPORARY` statement require media recovery.

   After migration, tablespaces that are offline when you open the Oracle9*i* database remain in Oracle7 database file format. The offline tablespaces can be brought online at any time after migration, and the file headers are converted to Oracle9*i* format at that time. In addition, if you want to avoid large restores in the event of a failure, then you can make all tablespaces except `SYSTEM` and `ROLLBACK` offline normal; then, you can restore only the datafiles for `SYSTEM` and `ROLLBACK` if you need to run another migration.

5. Make sure no user or role has the name OUTLN, because this schema is created automatically when you install Oracle9*i*. If you have a user or role named OUTLN, then you must drop the user or role and recreate it with a different name.

   To check for a user with the name OUTLN, issue the following SQL statement:

   ```
   SELECT USERNAME FROM dba_users WHERE USERNAME = 'OUTLN';
   ```

   If you do not have a user named OUTLN, then zero rows are selected.

   To check for a role with the name OUTLN, issue the following SQL statement:

   ```
   SELECT ROLE FROM dba_roles WHERE ROLE = 'OUTLN';
   ```

   If you do not have a role named OUTLN, then zero rows are selected.

6. Make sure no user or role has the name MIGRATE, because the Oracle Data Migration Assistant creates this schema and uses it to replace any pre-existing user or role with this name, and finally drops it from the system.

   To check for a user named MIGRATE, issue the following SQL statement:

   ```
   SELECT USERNAME FROM dba_users WHERE USERNAME = 'MIGRATE';
   ```

   If you do not have a user named MIGRATE, then zero rows are selected.

   To check for a role named MIGRATE, issue the following SQL statement:

   ```
   SELECT ROLE FROM dba_roles WHERE ROLE = 'MIGRATE';
   ```

   If you do not have a role named MIGRATE, then zero rows are selected.

7. Make sure the SYSTEM rollback segment does not have an OPTIMAL setting. An OPTIMAL setting may cause errors during migration.

   To check the OPTIMAL setting for the SYSTEM rollback segment, issue the following SQL statement:

   ```
   SELECT a.usn, a.name, b.optsize
       FROM v$rollname a, v$rollstat b
       WHERE a.usn = b.usn AND name='SYSTEM';
   ```

   Your output should be similar to the following:

   ```
   USN        NAME                           OPTSIZE
   ---------- ------------------------------ ----------
            0 SYSTEM
   1 row selected.
   ```

If there is a value in the OPTSIZE column, then issue the following SQL statement to set optimal to NULL:

```
ALTER ROLLBACK SEGMENT SYSTEM STORAGE (OPTIMAL NULL);
```

You can reset OPTIMAL when migration is complete.

> **See Also:** The troubleshooting information in "OPTIMAL Setting for the SYSTEM Rollback Segment" on page A-5.

8. Increase the maximum number of extents for your SYSTEM rollback segment by altering the MAXEXTENTS parameter in the STORAGE clause of the ALTER ROLLBACK SEGMENT statement (optional).

   The following is an example of the ALTER ROLLBACK SEGMENT statement:

   ```
   ALTER ROLLBACK SEGMENT system
      STORAGE (NEXT 500K MAXEXTENTS 121);
   ```

   You may need more space in the SYSTEM rollback segment to successfully complete the migration. If there is not enough space in your SYSTEM rollback segment, then you may encounter an error when you run the Oracle Data Migration Assistant.

9. Make sure your SYSTEM tablespace has enough free space to hold the Oracle9*i* data dictionary and the existing Oracle7 data dictionary concurrently.

   To check the free space in your SYSTEM tablespace, issue the following SQL statement:

   ```
   SELECT sum(bytes) FROM dba_free_space
      WHERE tablespace_name='SYSTEM';
   ```

   This statement displays the number of free bytes in the system tablespace.

   > **See Also:** "Space Requirements" on page 5-5 and "Assess System Requirements vs. Resources Available" on page 3-16 for more information.

10. Make sure all online data files are accessible and in the correct directories.

11. Adjust the initialization parameter file for use with Oracle9*i*. The initialization parameter file may still reside in the Oracle7 environment; the Oracle Data Migration Assistant will copy it to the Oracle9*i* environment when it is run.

Specifically, complete the following steps:

a. If you are updating materialized views automatically by using the JOB_QUEUE_PROCESSES initialization parameter, then set this parameter to 0 (zero) in the initialization parameter file. After migrating your database, you can change the setting for this parameter back to its normal setting.

b. Make sure the OPTIMIZER_MODE initialization parameter is set to choose. After migrating your database, you can change the setting for this parameter back to its normal setting.

c. Make sure your DB_DOMAIN initialization parameter is set properly.

See Also: "The DB_DOMAIN Parameter" on page B-11 for more information about setting this initialization parameter.

d. On Windows platforms, change the BACKGROUND_DUMP_DEST and USER_DUMP_DEST initialization parameters that point to RDBMS71, RDBMS72, or RDBMS73 to point to the following directories instead (optional):

Initialization Parameter	**Change Setting To**
BACKGROUND_DUMP_DEST	*ORACLE_BASE*\oradata\DB_NAME
USER_DUMP_DEST	*ORACLE_BASE*\oradata\DB_NAME\archive

In the settings, substitute the complete ORACLE_BASE path for *ORACLE_BASE* and substitute the database name for *DB_NAME*.

e. If the initialization parameter file contains an IFILE (include file) entry, then edit the file specified in the IFILE entry in the same way that you edited the initialization parameter file in sub-steps a to d.

Make sure you save the initialization parameter file and the file specified in the IFILE entry, if one exists, after making these adjustments.

Prepare for Migration on a Windows Platform

In addition to the steps described in the previous section, "Prepare the Oracle7 Source Database for Migration", complete the following steps if you are migrating your database on a Windows platform:

1. Make sure you have the required release of SQL*Net installed.

Migrating From	Required SQL*Net Release
Oracle release 7.3.2	SQL*Net release 2.3.2.1.12 or higher **Note:** If your release of SQL*Net is below release 2.3.2.1.4, you must first install release 2.3.2.1.4 before you can upgrade to release 2.3.2.1.12. Contact Oracle Support Services to obtain the patch that includes release 2.3.2.1.4.
Oracle release 7.3.3	SQL*Net release 2.3.3.0.3 or higher

If the required release of SQL*Net is not installed, complete the following steps to install it:

a. Obtain the year 2000-compliant Oracle Installer for release 7.3 from Oracle Corporation.

b. Start the Oracle Installer you obtained in Step a. Respond to the Oracle Installer screens until you reach the Software Asset Manager screen.

c. At the Software Asset Manager screen, click the From button.

d. Navigate to the drive containing the CD-ROM for the current release of Oracle.

e. Navigate to the appropriate directory on the CD-ROM:

If you are installing SQL*Net release 2.3.2.1.12, navigate to the following directory on the CD-ROM:

```
\patches\sqlnet\232112\nt_x86\install
```

If you are installing SQL*Net release 2.3.3.0.3, navigate to the following directory on the CD-ROM:

```
\patches\sqlnet\23303\nt_x86\install
```

f. Open the `nt.prd` file.

g. Complete the installation.

h. Exit the Oracle Installer.

See Also: Your *Oracle9i installation guide for Windows* for more information about required SQL*Net releases.

Note: If you cannot install the required SQL*Net release, contact Oracle Support Services.

---

2. Ensure all Oracle7 services are stopped, including the service for the Oracle7 database instance.

   See Also: Your *Administrator's Guide* for Windows for information about stopping services.

Install the Release 9.0.1 Oracle Software and Migrate the Database

Complete the following steps to install the release 9.0.1 software and migrate the database:

1. If you are using a UNIX operating system, then make sure you are logged in as a user with write permission to the *ORACLE_HOME* directory and all of its subdirectories.
2. Follow the instructions in your operating system-specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.
3. At the Welcome screen of the Oracle Universal Installer, click Next. The File Locations screen appears.

   If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, then click the Help button to open the online help.
4. At the File Locations screen, complete the following steps:
   a. Do not change the text in the Source field. This is the location of files for installation.
   b. If there is a Destination Name field, enter the name of a new Oracle home in this field.
   c. Enter the complete path of the Oracle home directory where you want to install the new release in the Destination Path field. Click the Browse button to navigate to the directory.

Note: You must install the new 9.0.1 release in a new Oracle home that is separate from the Oracle7 release.

---

   d. Click Next.

   The Available Products screen appears.

5. At the Available Products screen, select the Oracle9*i* server. The Oracle9*i* server is either Oracle9*i* Enterprise Edition or Oracle9*i*, depending on your installation medium. Then, click Next.
6. At the Installation Types screen, choose Custom. Do not choose Standard Edition or Enterprise Edition unless you want to install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom.

   > **Note:** Normally, you should not install a starter database if you are migrating an existing database.

   After you make your selection, click Next.

   If you chose Custom, the Available Product Components screen appears. Complete the following steps:

   a. Choose the product components you want to install. Then, click Next.

      Make sure you install all of the options you installed with the Oracle7 database, assuming you do not want to discontinue use of a particular option. For example, if you installed Oracle Advanced Replication in Oracle7, then you should install it in Oracle9*i*.

   b. Respond to the remaining screens that enable you to specify your custom installation settings, until you reach the Upgrading or Migrating an Existing Database screen.
7. At the Upgrading or Migrating an Existing Database screen, complete the following steps:

   a. Select the Upgrade or Migrate an Existing Database check box.

   b. Choose the Oracle7 database to migrate.

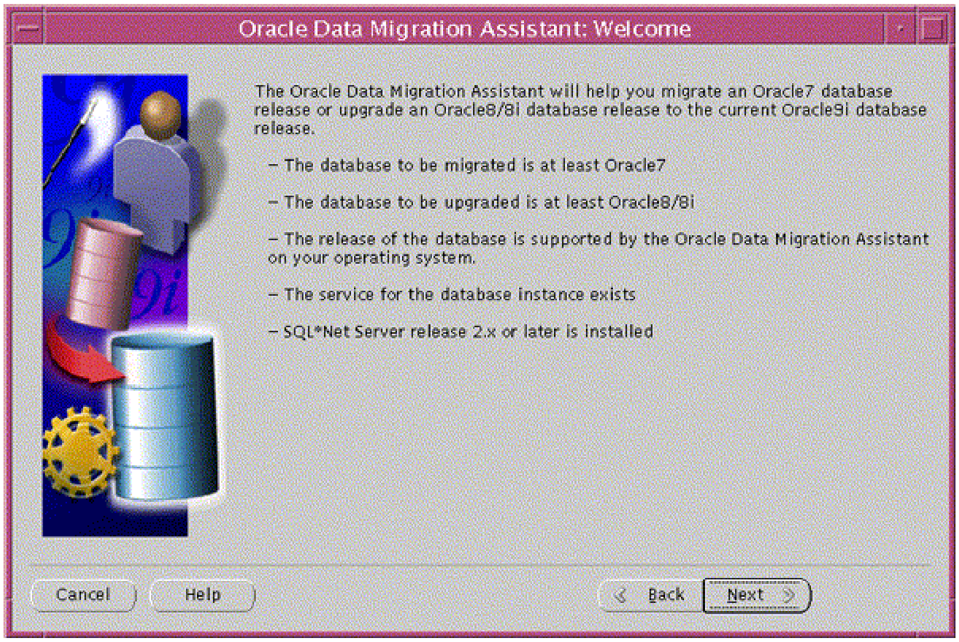
   c. Click Next.
8. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

When installation is complete, one or more assistants may be started. When the Oracle Data Migration Assistant is started, you are ready to proceed with the migration.

9. At the Before You Migrate or Upgrade screen of the Oracle Data Migration Assistant (shown in Figure 5–2), make sure the Oracle7 database that you are migrating meets the conditions specified. Then, click Next.

Note: The Oracle Data Migration Assistant uses a `SHUTDOWN IMMEDIATE` statement to shut down the database. Therefore, no users should be logged into the database when the Oracle Data Migration Assistant starts. Users who are logged in will be disconnected.

Figure 5–2 Before You Migrate or Upgrade Screen



The Oracle Data Migration Assistant will help you migrate an Oracle7 database release or upgrade an Oracle8/8i database release to the current Oracle9i database release.

- The database to be migrated is at least Oracle7
- The database to be upgraded is at least Oracle8/8i
- The release of the database is supported by the Oracle Data Migration Assistant on your operating system.
- The service for the database instance exists
- SQL*Net Server release 2.x or later is installed

If you need help at any screen or want to consult more documentation about the Oracle Data Migration Assistant, then click the Help button to open the online help.

10. At the Select a Database Instance screen, select the database instance of the Oracle7 database you are migrating. Then, click Next.
11. At the Database Password and INIT.ORA File screen, complete the following steps:
    a. Make sure the specified new Oracle home is correct.
    b. Make sure the location of the initialization parameter file specified is the complete path to the initialization parameter file of the Oracle7 database that you are migrating. If the location is incorrect, click the Browse button to navigate to the correct directory.
    c. Make sure the old Oracle home specified is the complete path to the Oracle home of the Oracle7 database you are migrating.
    d. Click Next. The Choose Migration or Upgrade Type screen appears.
12. Choose a migration type. Then, click Next.

    If you chose Custom, then respond to the screens that enable you to specify your custom migration settings until you reach the Backup Your Database screen. If you need help with any of the custom screens, click the Help button.
13. At the Backup Your Database screen, you have two options:
    - Choose "I have already backed up my database" if you completed a backup before running the Oracle Data Migration Assistant.
    - Choose "I would like this tool to back up the database" if you did not complete a backup. If you choose this option, then you can select the backup directory by clicking the Browse button.

    After you have made your choice, click Next.
14. At the Start the Migration or Upgrade screen, make sure all of the specifications are correct. If anything is incorrect, then click Back until you can correct the specification. If everything is correct, then click Next.

    When you click Next, the Status screen appears and the Oracle Data Migration Assistant begins to perform the migration. A status bar shows its progress. When the migration is complete, the Listener.ora Migration Confirmation screen appears.
15. At the Listener.ora Migration Confirmation screen, click the Yes button if you want the assistant to modify your `listener.ora` file automatically, or click the No button if you do not want the assistant to modify the `listener.ora` file.

Certain modifications are required to the `listener.ora` file for your database to work properly with Oracle Enterprise Manager. If you plan to use Oracle Enterprise Manager, then you should click the Yes button to automatically modify the `listener.ora` file. However, if you do not plan to use Oracle Enterprise Manager, then click the No button.

If you click the Yes button, then the Oracle Data Migration Assistant modifies the `listener.ora` file in the following way:

a. The assistant modifies the SID_DESC entry for the migrated database in the Oracle9*i* `listener.ora` in one of the following ways:

A simple case: Suppose the old `listener.ora` has the following SID_DESC entry:

```
...
   (SID_DESC =
      (SID_NAME = ORCL)
   )
...
```

If the database name is SAL, then the domain name is COM, and the Oracle home is `/oracle/product/9.0`, the assistant adds the following entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = sal.com)
          (ORACLE_HOME = /oracle/product/9.0)
          (SID_NAME = SAL)
    )
...
```

A more complicated case: Suppose the old `listener.ora` has the following SID_DESC entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = an_entry)
       (SID_NAME = ORCL)
    )
...
```

If *an_entry* does not match the GLOBAL_DBNAME of the migrated database, and if the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/9.0`, then the assistant adds the following entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = sal.com)
          (ORACLE_HOME = /oracle/product/9.0)
          (SID_NAME = SAL)
    )
...
```

This entry is the same as the entry in the simple case, but the assistant also adds the entry *an_entry* to the `SERVICE_NAMES` parameter. Therefore, the assistant changes the `SERVICE_NAMES` parameter to the following:

```
SERVICE_NAMES = sal.com, an_entry
```

b. The assistant removes the entry of the migrated database from the old `listener.ora` file.

c. The assistant starts up the Oracle9*i* listener.

d. The assistant reloads the `listener.ora` file in both the Oracle7 and the Oracle9*i* environments.

When the Oracle Data Migration Assistant is complete, go to "Finish the Migration" on page 5-21 for information about additional migration steps.

Running the Oracle Data Migration Assistant Independently

If you installed Oracle9*i* without specifying that you are migrating an existing database, then you can run the Oracle Data Migration Assistant independently after the Oracle9*i* installation is complete.

Complete the following steps to run the Oracle Data Migration Assistant independently:

1. Start the Oracle Data Migration Assistant.

   On UNIX, enter the following command at a system prompt:

   ```
   odma
   ```

   On Windows platforms, choose:

   ```
   Start > Programs > Oracle - HOME_NAME > Migration Utilities >
   Oracle Data Migration Assistant
   ```

   When you start the Oracle Data Migration Assistant, its Before You Migrate or Upgrade screen appears (see Figure 5–2 on page 5-17).

2. Respond to the instructions in each Oracle Data Migration Assistant window, and click Next when you are ready to continue to the next window. When you reach the last window, click Next to begin the migration of the database. For more information about using the Oracle Data Migration Assistant, click the Help button in any window.

   See Also: Step 9 to Step 15 in "Install the Release 9.0.1 Oracle Software and Migrate the Database" on page 5-15 for more information about the windows that appear in the Oracle Data Migration Assistant.

Finish the Migration

Complete the following steps after you have successfully run the Oracle Data Migration Assistant:

1. Change to the `*ORACLE_HOME*/rdbms/admin` directory in the Oracle9*i* Oracle home.
2. Start SQL*Plus.
3. Connect to the database instance as a user with SYSDBA privileges.

   Note: You must connect as the same user who owns the ORACLE_HOME and the database to be migrated.

4. Run `STARTUP RESTRICT`:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the `PFILE` option to specify the location of your initialization parameter file.

   Note: The database may already be started. If so, then you do not need to restart it.

5. Set the system to spool results to a log file for later verification of success:

   ```
   SQL> SPOOL catoutma.log
   ```

If you want to see the output of the scripts you will run on your screen, then you can also issue a `SET ECHO ON` statement:

```
SQL> SET ECHO ON
```

6. Run `utlrp.sql` if you did not run it using the Oracle Data Migration Assistant (optional):

   ```
   SQL> @utlrp.sql
   ```

   The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, and so on. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

   Oracle Corporation highly recommends performing this optional step.

   If you encounter any problems when you run these scripts, or any of the scripts in the remaining steps, then correct the causes of the problems and rerun the scripts. You can rerun any of the scripts described in this chapter as many times as necessary.

7. Turn off the spooling of script results to the log file:

   ```
   SQL> SPOOL OFF
   ```

   Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 5; the suggested name was `catoutma.log`.

   You should look for errors that alert you to insufficient space, and for errors that alert you that a script failed to run. If you see these types of errors, then your migration may not be completely successful. However, you typically can ignore errors about the failure to alter or drop an object that does not exist.

   If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

   ```
   SQL> SET ECHO OFF
   ```

8. Run `SHUTDOWN` on the Oracle9*i* database:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

Caution: Use `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE`. Do not use `SHUTDOWN ABORT`.

Executing this clean shutdown flushes all caches, clears buffers, and performs other DBMS housekeeping activities. These measures are an important final step to ensure the integrity and consistency of the newly migrated Oracle9*i* database.

9. Adjust the initialization parameter file for Oracle9*i*.

   Alter any parameter whose syntax has changed in Oracle9*i*; refer to Appendix B, "Changes to Initialization Parameters", for lists of new, renamed, and obsolete parameters.

   Also, learn about the new parameters listed in Appendix B, "Changes to Initialization Parameters", and decide which ones you want to use for your migrated database.

   In addition, the Oracle Data Migration Assistant sets the `COMPATIBLE` initialization parameter to 8.0.5. See Chapter 9, "Compatibility and Interoperability", for information about resetting the `COMPATIBLE` initialization parameter.

   See Also: *Oracle9i Database Reference* for detailed information about initialization parameters.

10. If your operating system is Windows, then restart your computer.
11. Complete the procedures described in Chapter 8, "After Migrating or Upgrading the Database".

Caution: If you retain the old Oracle7 software, then never start the migrated database with the old Oracle7 software. Only start the database with the executables in the new Oracle9*i* installation directory. Also, before you remove the old Oracle7 environment, make sure you relocate any datafiles in that environment to the Oracle9*i* environment. See the *Oracle9i Database Administrator's Guide* for information about relocating datafiles.

Troubleshooting Errors During Migration

Errors may be caused by the following actions or omissions:

- Performing a migration step out of order
- Failing to fulfill the prerequisites for migration

- Encountering an occasional conversion irregularity

See Also: Appendix A, "Troubleshooting Migration Problems" and *Oracle9i Database Error Messages* for information about errors during migration and about corrective action for each error.

Abandoning the Migration

The easiest way to abandon a migration is to restore the backup of your Oracle7 database that you took before you ran the Oracle Data Migration Assistant.

6 Migrating Using Export/Import

This chapter guides you through the process of migrating an Oracle7 database to Oracle9*i* using Export/Import. This chapter covers the following topics:

- Basics of Export/Import
- Migrate the Source Database Using Export/Import

See Also: *Oracle9i Database Utilities* for detailed information about using the Export and Import utilities.

Basics of Export/Import

To migrate a database using Export/Import, complete the following three basic steps:

1. Export the data from the database you are migrating (the source database). The export physically copies the data to the export dump file.
2. Create the Oracle9*i* database into which you will import the exported data (the target database).
3. Import the exported data into the new Oracle9*i* target database.

See Also: "Choose a Migration Method" on page 3-3 and *Oracle9i Database Utilities* for information that can help you to evaluate the choice of Export/Import for migration.

Export Utility Requirements

To migrate or upgrade a database, use the Export utility shipped with the release of the **source** database. After the export, the Import utility can copy the data from the export dump file into the target database. The target database must be created and operational before the Import utility can migrate the exported data into the target database.

For example, if you are migrating to release 9.0.1 from release 7.3, then use the Export utility for release 7.3.

Note: If the source Oracle database is earlier than version 6, then first migrate the source database to at least version 6 before proceeding with the export.

Import Utility Requirements

To migrate, upgrade, or downgrade a database, use the Import utility shipped with the release of the **target** database. For example, if you are migrating to release 9.0.1 from release 7.3, then use the Import utility for release 9.0.1.

Additional Options

Refer to the following sources if you have additional options installed:

- The Trusted Oracle documentation for information about migrating the features of the Trusted Oracle database if you are exporting from, or importing to, a Trusted Oracle database.
- Appendix G, "Migration and Compatibility for Replication Environments" if you are migrating a database system that has Oracle Replication installed.

Migrate the Source Database Using Export/Import

To migrate an Oracle7 or version 6 database using the Export/Import utilities, complete the following steps:

1. Export the source database using the Export utility shipped with the source database. See the source database's server utilities documents for information about using the Export utility on the source database. Both Oracle7 and version 6 database exports can be imported into Oracle9*i*.

   To ensure a consistent export, make sure the source database is not available for updates during and after the export. If the source database will be available to users for updates after the export, then, prior to making the source database available, put procedures in place to copy the changes made in the source database to the Oracle9*i* target database after the import is complete.

2. Install the Oracle9*i* software. Installation is operating system-specific. Installation steps for Oracle9*i* are covered in your operating system-specific Oracle9*i* documentation.
3. If the new Oracle9*i* database will have the same name as the existing source database, then shut down the existing database before creating the new Oracle9*i* database.
4. Create the Oracle9*i* target database.

   See Also: *Oracle9i Database Administrator's Guide* for information about creating an Oracle9*i* database.

5. Start SQL*Plus in the Oracle9*i* environment.
6. Connect to the database instance as a user with SYSDBA privileges.
7. Start an Oracle9*i* database instance using STARTUP.

8. Pre-create tablespaces, users, and tables in the target database to improve space usage by changing storage parameters. When you pre-create tables using SQL*Plus, either run the database in the original database compatibility mode or make allowances for the specific data definition conversions that occur during import.

   Note: If the new Oracle9*i* database will be created on the same computer as the source database, and you do not want to overwrite the source database datafiles, then you must pre-create the tablespaces and specify `IGNORE=Y` and `DESTROY=N` when you import.

   ---

9. Use the Oracle9*i* Import utility to import the objects exported from the source database. Include the `LOG` parameter to save the informational and error messages from the import session to a file.

   See Also: *Oracle9i Database Utilities* for a complete description of the Import utility.

10. After the migration, check the import log file for information about which imports of which objects completed successfully and, if there were failures, which failed.

    See Also: *Oracle9i Database Utilities* and the Oracle9*i* server `README.doc` file for error handling information.

11. Use further Import scenarios (see *Oracle9i Database Utilities*) or SQL scripts that create the source objects to clean up incomplete imports (or possibly to start an entirely new import).
12. If changes are made to the source database after the export, then make sure those changes are propagated to the Oracle9*i* database prior to making it available to users. See Step 1 on page 6-3 for more information.
13. Complete the procedures described in Chapter 8, "After Migrating or Upgrading the Database".

7 Upgrading from an Older Release of Oracle to the New Oracle9*i* Release

This chapter contains information about upgrading your current release of Oracle to the new Oracle9*i* release. The information in this chapter only applies to release 8.0 and higher installations of Oracle. If your current release is version 6 or version 7 and you want to migrate to Oracle9*i*, then follow the instructions at the beginning of this book, starting with Chapter 2, "Overview of Migration".

This chapter covers the following topics:

- Upgrade Paths
- Upgrading the Database to the New Oracle9i Release
- Upgrading Specific Components
- Changing the Word-Size of Your Current Release

> **See Also:** Some aspects of upgrading are operating system-specific. See your operating system-specific Oracle documentation for additional instructions about upgrading on your operating system.

Upgrade Paths

The path that you must take to upgrade your database to the new release depends on the release you are currently using. Table 7–1 contains the upgrade path required for each old release of Oracle. Use the upgrade path and the specified documentation to upgrade the release you are currently running.

Table 7–1 Upgrade Paths

Old Release	Upgrade Path
8.0.3 8.0.4 8.0.5	Direct upgrade is *not* supported. Complete the following steps to upgrade to the new release: **1.** Upgrade to release 8.0.6 using the instructions in the release 8.0.6 `READMEMIG.doc` file. **2.** Upgrade the release 8.0.6 database to the new release using the instructions in "Upgrading the Database to the New Oracle9i Release" on page 7-2.
8.0.6 8.1.5 8.1.6 8.1.7	Direct upgrade is supported. Upgrade to the new release using the instructions in "Upgrading the Database to the New Oracle9i Release" on page 7-2.

Upgrading the Database to the New Oracle9*i* Release

This section guides you through the process of upgrading your database to the new Oracle9*i* release.

Character Set Considerations

In Oracle9*i*, the SQL NCHAR datatypes (`NCHAR`, `NVARCHAR2`, and `NCLOB`) will be limited to the Unicode character set encoding (UTF8 and AL16UTF16) only. Any other version 8 character sets that were available under the NCHAR data type, including Asian character sets (for example, such as JA16SJISFIXED), will no longer be supported.

Before migrating your SQL NCHAR data to the new Unicode NCHAR, Oracle Corporation recommends that you analyze your SQL NCHAR data, using the Character Set Scanner for the identification of possible invalid character set conversion or data truncation.

See Also: *Oracle9i Globalization Support Guide* for more information about the Character Set Scanner

When you upgrade to Oracle9*i*, the value of the National Character Set of the upgraded database is set based on the value of the National Character Set of the version 8 database being upgraded.

If the old National Character Set is UTF8, then the new National Character Set will be UTF8. Otherwise, the National Character Set is changed to AL16UTF16.

During the upgrade, the existing NCHAR columns in the data dictionary are changed to use the new Oracle9*i* format and, if the National Character Set has been changed to AL16UTF16, the dictionary NCHAR columns will be converted to the AL16UTF16 character set.

Note: NCHAR columns in user tables are not changed during the upgrade. To change NCHAR columns in user tables, see "Upgrade User NCHAR Columns" on page 8-3.

AL24UTFFSS Character Set Desupported

The AL24UTFFSS Unicode character set has been desupported in Oracle9*i*. AL24UTFFSS was introduced in Oracle7 as the Unicode character set supporting the UTF-8 encoding scheme based on the Unicode 1.1 standard, which is now obsolete. In Oracle9*i*, The Unicode database character sets AL32UTF8 and UTF8, include the Unicode enhancements based on the Unicode 3.1 standard.

The migration path for existing AL24UTFFSS databases is to upgrade your database character set to UTF8 prior to upgrading to Oracle9*i*. As with all migrations to a new database character set, Oracle Corporation recommends you use the Character Set Scanner for data analysis before attempting to migrate your existing database character set to UTF8.

See Also: *Oracle9i Globalization Support Guide* for more information about the Character Set Scanner

Considerations for Replication Environments

If you plan to use CHAR column length semantics in Oracle9*i*, or if your replication database contains tables with NCHAR or NVARCHAR2 columns, then this section contains considerations for upgrading a replication environment to Oracle9*i*.

CHAR Column Length Semantics

If you plan to use CHAR column length semantics in a replication database after you upgrade it to Oracle9*i*, then all of the databases participating with that database in the replication environment must also use CHAR column length semantics. In this case, Oracle Corporation recommends that you upgrade all of the databases participating in the replication environment at the same time. This applies to both master sites and materialized view sites in your replication environment.

If you cannot upgrade all of the databases in your replication environment at the same time, then you can only use CHAR column length semantics in your Oracle9*i* databases if all of the databases prior to Oracle9*i* are using a single-byte character set. Otherwise, do not switch to CHAR column length semantics in the Oracle9*i* database until all of the other databases in the replication environment are upgraded to Oracle9*i*.

NCHAR or NVARCHAR2 Columns

If your replication database contains tables with NCHAR or NVARCHAR2 columns, then Oracle Corporation recommends that you upgrade all of the databases participating in the replication environment at the same time. This applies to both master sites and materialized view sites in your replication environment. In Oracle9*i*, all columns specified as NCHAR or NVARCHAR2 datatype are stored in Unicode format.

If you cannot upgrade all of the databases in your replication environment at the same time, then interoperability is only supported if all of the databases prior to Oracle9*i* are using a fixed width national character set. If any of the databases prior to Oracle9*i* are using a variable width character set, then you must convert these databases to fixed width character sets before you upgrade any of the other databases in the replication environment to Oracle9*i*.

See Also:

- *Oracle9i Replication* for more information about replication support for column length semantics and Unicode
- *Oracle9i Globalization Support Guide* for general information about column length semantics and Unicode
- *Oracle8i National Language Support Guide* for information about converting character sets in release 8.1

Upgrading Oracle Parallel Server

If you are upgrading a system that has Oracle Parallel Server installed, then most of the actions described in this chapter should be performed on only one node of the system. So, perform the actions described in this chapter on only one node unless instructed otherwise in a particular step.

Note: Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

Prepare to Upgrade

Complete the following steps to begin the upgrade process:

1. Review upgrade issues relating to SQL*Net, Net8, and Oracle Net.

   See Also: Appendix F, "Migration and Compatibility for Oracle Net Services", for information.

2. Make sure your `DB_DOMAIN` initialization parameter is set properly.

   See Also: "The DB_DOMAIN Parameter" on page B-11 for more information about setting this initialization parameter.

3. Log in to the system as the owner of the Oracle home directory of the release from which you are upgrading.
4. Start SQL*Plus.
5. Connect to the database instance as a user with `SYSDBA` privileges.
6. If you are upgrading from release 8.0.6, then make sure no user or role has the name `OUTLN`, because this schema is created automatically when you install Oracle9*i*. If you have a user or role named `OUTLN`, then you must drop the user or role and recreate it with a different name.

   Note: If you are upgrading from an 8.1 release, then you do not need to perform this check because the `OUTLN` user should have been created when you installed the 8.1 release. Therefore, if you are upgrading from an 8.1 release, then go to Step 7. Do *not* drop the `OUTLN` user if you are upgrading from an 8.1 release.

   To check for a user with the name `OUTLN`, enter the following SQL statement:

   ```
   SELECT username FROM dba_users WHERE username = 'OUTLN';
   ```

If you do not have a user named OUTLN, then zero rows are selected.

To check for a role with the name OUTLN, enter the following SQL statement:

```
SELECT role FROM dba_roles WHERE role = 'OUTLN';
```

If you do not have a role named OUTLN, then zero rows are selected.

7. Add space to your SYSTEM tablespace and to the tablespaces where you store rollback segments, if necessary.

   Upgrading to a new release requires more space in your SYSTEM tablespace and in the tablespaces where you store rollback segments. If you have enough space on your system, then consider adding more space to these tablespaces. In general, you need at least 50 MB of free space in your SYSTEM tablespace to upgrade. If you run out of space during the upgrade, then you will need to perform the upgrade again.

   The following SQL statement illustrates how to add more space to a tablespace:

```
ALTER TABLESPACE system
   ADD DATAFILE '/home/user1/mountpoint/oradata/db1/system02.dbf'
   SIZE 50M
   AUTOEXTEND ON NEXT 10M MAXSIZE UNLIMITED;

ALTER ROLLBACK SEGMENT rb
   STORAGE (MAXEXTENTS UNLIMITED);
```

8. Determine the files that you will back up in Step 11 by issuing the following SQL statements:

```
SPOOL v8files.log
SELECT member FROM v$logfile;
SELECT name FROM v$datafile;
SELECT value FROM v$parameter WHERE name = 'control_files';
SPOOL OFF
```

   The v8files.log spool file lists all of the files that you must back up in Step 11.

9. Run SHUTDOWN IMMEDIATE on the database:

```
SQL> SHUTDOWN IMMEDIATE
```

   If you are using Oracle Parallel Server, then shutdown all instances.

10. If your operating system is Windows, then ensure all Oracle services are stopped.

See Also: Your *Administrator's Guide* for Windows for information about stopping services.

11. Perform a full offline backup of the database. Make sure you back up the files listed in the `v8files.log` spool file that you generated in Step 8.

Caution: If you encounter any problems with the upgrade, then you will need to restore the database from this backup. Therefore, make sure you back up your database now as a precaution.

See Also:

- *Oracle8i Backup and Recovery Guide* for more information about backing up a release 8.1 database
- *Oracle8 Backup and Recovery Guide* for more information about backing up a release 8.0 database

12. Exit SQL*Plus.

Upgrade the Database

Choose an upgrade method and then follow the instructions for upgrading using the method you have chosen.

See Also: Some aspects of upgrading are operating system-specific. See your operating system-specific Oracle documentation for additional information about migrating.

Choose an Upgrade Method

There are two ways to upgrade your database to release 9.0.1. You can either use the Oracle Data Migration Assistant to complete the upgrade, or you can perform the upgrade manually.

The Oracle Data Migration Assistant provides a completely automated upgrade of your database. You use a graphical user interface (GUI), which guides you through each step of the process. In addition, the Oracle Data Migration Assistant includes extensive online help. The Oracle Data Migration Assistant runs the appropriate upgrade script for your current release, deletes any obsolete initialization parameters from your initialization parameter file, and optionally configures your `listener.ora` file. See Appendix B, "Changes to Initialization Parameters", for lists of obsolete initialization parameters.

On the other hand, you lose some flexibility and control by using the Oracle Data Migration Assistant. If you want complete control over the upgrade process, especially with regard to setting initialization parameters, then you may want to perform the upgrade manually.

Caution: The Oracle Data Migration Assistant cannot upgrade systems with Oracle Parallel Server installed. If you have Oracle Parallel Server installed, then you must upgrade the database manually.

Decide which method you want to use to upgrade your database, and then complete the steps in one of the following sections accordingly:

- Upgrade the Database Using the Oracle Data Migration Assistant on page 7-8.
- Upgrade the Database Manually on page 7-15.

Upgrade the Database Using the Oracle Data Migration Assistant

Complete the following steps to upgrade the database using the Oracle Data Migration Assistant:

1. Follow the instructions in your operating system-specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.

   If you need help at any screen or want to consult more documentation about the Oracle Universal Installer, then click the Help button to open the online help.

2. At the Welcome screen of the Oracle Universal Installer, click Next. The File Locations screen appears.

3. At the File Locations screen, complete the following steps:

   a. Do not change the text in the Source field. This is the location of files for installation.

   b. If there is a Destination Name field, enter the name of a new Oracle home in this field.

c. Enter the complete path of the Oracle home directory where you want to install the new release in the Destination Path field. Click the Browse button to navigate to the directory.

> **Note:** You must install the new 9.0.1 release in a new Oracle home that is separate from the 8.0 or 8.1 release.

d. Click Next.

The Available Products screen appears.

4. At the Available Products screen, select the Oracle9*i* server. The Oracle9*i* server is either Oracle9*i* Enterprise Edition or Oracle9*i*, depending on your installation medium. Then, click Next.
5. At the Installation Types screen, choose Custom. Do not choose Standard Edition or Enterprise Edition unless you want to install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom.

> **Note:** Normally, you should not install a starter database if you are upgrading an existing database.

After you make your selection, click Next.

If you chose Custom, the Available Product Components screen appears. Complete the following steps:

a. Choose the product components you want to install. Then, click Next.

Make sure you install all of the options you installed with the previous database, assuming you do not want to discontinue use of a particular option. For example, if you installed Oracle Parallel Server in the previous database, then you should install Oracle9*i* Real Application Clusters in the new Oracle9*i* database.

b. Respond to the remaining screens that enable you to specify your custom installation settings, until you reach the Upgrading or Migrating an Existing Database screen.

6. At the Upgrading or Migrating an Existing Database screen, complete the following steps:

   a. Select the Upgrade or Migrate an Existing Database check box.
   b. Choose the database to upgrade.
   c. Click Next.

7. At the Create Database screen, select the No option, indicating that you do not want to create a database because you are upgrading an existing database. Then, click Next.

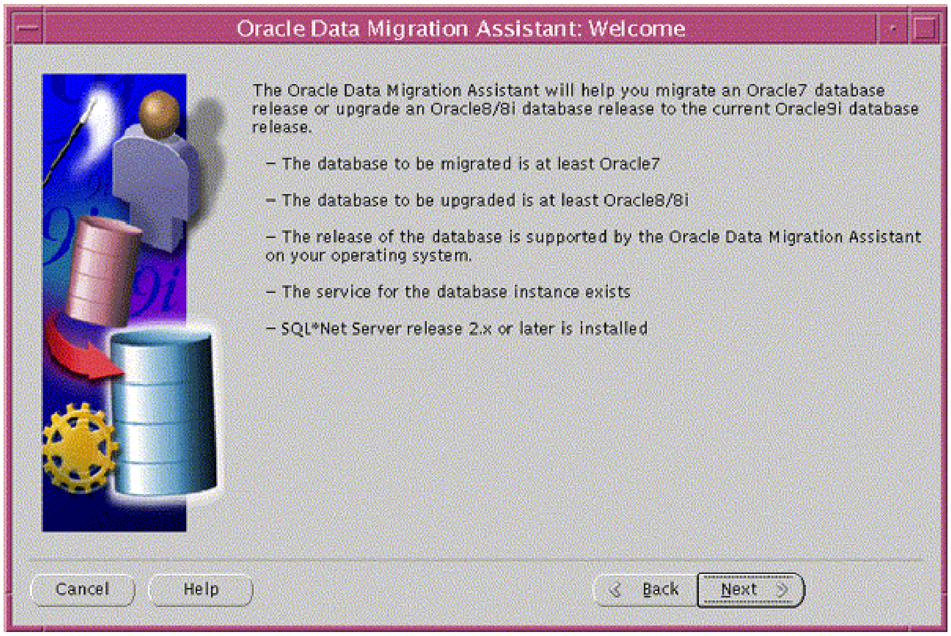
8. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

   When installation is complete, one or more assistants may be started. When the Oracle Data Migration Assistant is started, you are ready to proceed with the upgrade.

9. At the Before You Migrate or Upgrade screen of the Oracle Data Migration Assistant (shown in Figure 7–1), make sure the database that you are upgrading meets the conditions specified. Then, click Next.

   If you need help at any screen or want to consult more documentation about the Oracle Data Migration Assistant, then click the Help button to open the online help.

Figure 7–1 Before You Migrate or Upgrade Screen



The Oracle Data Migration Assistant will help you migrate an Oracle7 database release or upgrade an Oracle8/8i database release to the current Oracle9i database release.

- The database to be migrated is at least Oracle7
- The database to be upgraded is at least Oracle8/8i
- The release of the database is supported by the Oracle Data Migration Assistant on your operating system.
- The service for the database instance exists
- SQL*Net Server release 2.x or later is installed

10. At the Select a Database Instance screen, select the database instance of the database you are upgrading. Then, click Next.

Note: The database you choose must be release 8.0 or higher. If the database is an Oracle7 or lower database, then you must complete a migration of the database, not an upgrade. If the database is an Oracle7 or lower database, then exit the Oracle Data Migration Assistant, and see Chapter 2 to start the migration process.

---

11. At the Database Password and INIT.ORA File screen, complete the following steps:

    a. Make sure the specified new Oracle home is correct.

    b. Make sure the location of the initialization parameter file specified is the complete path to the initialization parameter file of the database that you are upgrading. If the location is incorrect, click the Browse button to navigate to the correct directory.

c. Make sure the old Oracle home specified is the complete path to the Oracle home of the database you are upgrading.

d. Click Next. The Choose Migration or Upgrade Type screen appears.

12. Choose an upgrade type. Then, click Next.

    If you chose Custom, then respond to the screens that enable you to specify your custom migration settings until you reach the Backup Your Database screen. If you need help with any of the custom screens, click the Help button.

13. At the Backup Your Database screen, you have two options:

    - Choose "I have already backed up my database" if you completed a backup before running the Oracle Data Migration Assistant.
    - Choose "I would like this tool to back up the database" if you did not complete a backup. If you choose this option, then you can select the backup directory by clicking the Browse button.

    After you have made your choice, click Next.

14. At the Start the Migration or Upgrade screen, make sure all of the specifications are correct. If anything is incorrect, then click Back until you can correct the specification. If everything is correct, then click Next.

    The Oracle Data Migration Assistant performs the upgrade. During the upgrade, the assistant sets the `COMPATIBLE` initialization parameter to 8.1.0. See Chapter 9, "Compatibility and Interoperability" for information about setting the `COMPATIBLE` initialization parameter after the upgrade.

    You may encounter the following error message:

    ```
    An error occurred while upgrading your Oracle database. Check the log files to
    determine if the upgrade was successful.
    ```

    If you encounter this message, then complete the following steps:

    a. Click the OK button.

    b. Click the View Summary in the subsequent box that appears. If the errors are similar to the following, then you can ignore the messages:

    ```
    ORA-00604: error occurred at recursive SQL level 1
    ORA-00001: unique constraint (SYSTEM.AQ$_QUEUES_CHECK) violated
    ORA-06512: at "SYS.DBMS_AQADM", line 2023
    ```

```
ORA-06512: at line 2
```

If other errors appear, then you must address them accordingly. For example, if you receive the error message "**error accessing package DBMS_ APPLICATION_INFO**", it means that the migration utility does not recognize that your old database is not release 9.0.1. The solution to this is to run the appropriate upgrade script, located in the *`ORACLE_HOME`*`/rdbms/admin` directory. For example, if your old database is release 8.1.5, run `u0801050.sql`.

Caution: If you retain the old Oracle software, then never start the upgraded database with the old software. Only start the database with the executables in the new release 9.0.1 installation directory.

15. At the Listener.ora Migration Confirmation screen, click the Yes button if you want the assistant to modify your `listener.ora` file automatically, or click the No button if you do not want the assistant to modify the `listener.ora` file.

    Certain modifications are required to the `listener.ora` file for your database to work properly with Oracle Enterprise Manager. If you plan to use Oracle Enterprise Manager, then you should click the Yes button to automatically modify the `listener.ora` file. However, if you do not plan to use Oracle Enterprise Manager, then click the No button.

    If you click the Yes button, then the Oracle Data Migration Assistant modifies the `listener.ora` file in the following way:

    a. The assistant shuts down the old listener and the new Oracle9*i* listener.

    b. The assistant modifies the SID_DESC entry for the migrated database in the Oracle9*i* `listener.ora` file in one of the following ways:

       A simple case: Suppose the old `listener.ora` has the following SID_ DESC entry:

       ```
       ...
          (SID_DESC =
             (SID_NAME = ORCL)
          )
       ...
       ```

If the database name is SAL, the domain name is COM, and the Oracle home is `/oracle/product/9.0`, then the assistant adds the following entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = sal.com)
          (ORACLE_HOME = /oracle/product/9.0)
          (SID_NAME = SAL)
    )
...
```

A more complicated case: Suppose the old `listener.ora` has the following SID_DESC entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = an_entry)
       (SID_NAME = ORCL)
    )
...
```

If *an_entry* does not match the GLOBAL_DBNAME of the migrated database, and if the database name is SAL, the domain name is COM, and the Oracle home is `\oracle\product\9.0` on the D drive, then the assistant adds the following entry:

```
...
   (SID_DESC =
       (GLOBAL_DBNAME = sal.com)
          (ORACLE_HOME = d:\oracle\product\9.0)
          (SID_NAME = SAL)
    )
...
```

This entry is the same as the entry in the simple case, but the assistant also adds the entry *an_entry* to the SERVICE_NAMES parameter in the `listener.ora` file. Therefore, the assistant changes the SERVICE_NAMES parameter to the following:

```
SERVICE_NAMES = sal.com, an_entry
```

c. On Windows platforms, the assistant removes the entry of the migrated database from the old `listener.ora` file. The assistant does not perform this action on UNIX operating systems.

d. The assistant starts up the Oracle9*i* listener.

16. Complete the procedures described in "Upgrading Specific Components" on page 7-24 and in Chapter 8, "After Migrating or Upgrading the Database".

> **Caution:** If you retain the old Oracle software, then never start the upgraded database with the old Oracle software. Only start the database with the executables in the new Oracle9*i* installation. Also, before you remove the old Oracle environment, make sure you relocate any datafiles in that environment to the new Oracle9*i* environment. See the *Oracle9i Database Administrator's Guide* for information about relocating datafiles.

Running the Oracle Data Migration Assistant Independently If you installed Oracle9*i* without specifying that you are migrating or upgrading an existing database, then you can run the Oracle Data Migration Assistant independently after the Oracle9*i* installation is complete.

Complete the following steps to run the Oracle Data Migration Assistant independently:

1. Start the Oracle Data Migration Assistant.

   On UNIX, enter the following command at a system prompt:

   ```
   odma
   ```

   On Windows platforms, choose:

   ```
   Start > Programs > Oracle - HOME_NAME > Migration Utilities >
   Oracle Data Migration Assistant
   ```

   When you start the Oracle Data Migration Assistant, its Before You Migrate or Upgrade screen appears (see Figure 7–1 on page 7-11).

2. Respond to questions in each Oracle Data Migration Assistant window, and click Next when you are ready to continue to the next window.

   > **See Also:** Step 9 to Step 16 in "Upgrade the Database Using the Oracle Data Migration Assistant" on page 7-8 for more information.

Upgrade the Database Manually

Complete the following steps to upgrade the database manually using SQL scripts:

1. Follow the instructions in your operating system-specific Oracle documentation to prepare for installation and start the Oracle Universal Installer.

   If you are upgrading a system with Oracle Parallel Server installed, then see *Oracle9i Real Application Clusters Installation and Configuration* for additional installation instructions.

2. At the Welcome screen of the Oracle Universal Installer, click Next. The File Locations screen appears.

3. At the File Locations screen, complete the following steps:

   a. Do not change the text in the Source field. This is the location of files for installation.

   b. If there is a Destination Name field, enter the name of a new Oracle home in this field.

   c. Enter the complete path of the Oracle home directory where you want to install the new release in the Destination Path field. Click the Browse button to navigate to the directory.

   Note: You must install the new 9.0.1 release in a new Oracle home that is separate from the 8.0 or 8.1 release.

   d. Click Next.

   The Available Products screen appears.

4. At the Available Products screen, select the Oracle9*i* server. The Oracle9*i* server is either Oracle9*i* Enterprise Edition or Oracle9*i*, depending on your installation medium. Then, click Next.

5. At the Installation Types screen, choose Custom. Do not choose Standard Edition or Enterprise Edition unless you want to install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom.

   Note: Normally, you should not install a starter database if you are upgrading an existing database.

   After you make your selection, click Next.

If you chose Custom, the Available Product Components screen appears. Complete the following steps:

a. Choose the product components you want to install. Then, click Next.

   Make sure you install all of the options you installed with the previous database, assuming you do not want to discontinue use of a particular option. For example, if you installed Oracle Parallel Server in the previous database, then you should install Oracle9*i* Real Application Clusters in the new Oracle9*i* database.

b. If you are installing Oracle9*i* Real Application Clusters, then, at the Cluster Node Selection screen, select the nodes onto which you want the software installed. Then, click Next.

c. Respond to the remaining screens that enable you to specify your custom installation settings, until you reach the Upgrading or Migrating an Existing Database screen.

6. At the Upgrading or Migrating an Existing Database screen, leave the Upgrade or Migrate an Existing Database check box unselected. Then, click Next.

   If you select the Upgrade or Migrate an Existing Database check box, then the Oracle Data Migration Assistant is started automatically after installation. Because you are following the instructions for upgrading the database manually, you should not start the Oracle Data Migration Assistant.

   Note: The Oracle Data Migration Assistant does not support Oracle Parallel Server migrations.

7. At the Summary screen, make sure all of the settings and choices are correct for your installation. Then, click Install. The Oracle Universal Installer performs the installation, which may take some time.

   When installation has completed successfully, click the Exit button to close the Oracle Universal Installer.

8. If your operating system is Windows, then complete the following steps:

   a. Shut down and restart your computer.

   b. Stop the Oracle service OracleService*SID* of the database you are upgrading, where *SID* is the instance name. For example, if your *SID* is ORCL, then enter the following at an MS-DOS prompt:

```
C:\> NET STOP OracleServiceORCL
```

c. Delete the Oracle service at the MS-DOS command prompt using ORADIM. The following table lists the MS-DOS command to run for each Oracle9*i* release:

Oracle Release...	Enter at the MS-DOS Command Prompt...
8.0	`C:\> ORADIM80 -DELETE -SID` *`SID`*
8.1	`C:\> ORADIM -DELETE -SID` *`SID`*
9.0	`C:\> ORADIM -DELETE -SID` *`SID`*

For example, if your Oracle release is release 8.0.6 and your *SID* is ORCL, then enter the following MS-DOS command:

```
C:\> ORADIM80 -DELETE -SID ORCL
```

If your Oracle release is release 8.1.7 and your *SID* is ORCL, then enter the following MS-DOS command:

```
C:\> ORADIM -DELETE -SID ORCL
```

d. Create the Oracle9*i* database service at the MS-DOS command prompt:

```
C:\> ORADIM -NEW -SID SID -INTPWD PASSWORD -MAXUSERS USERS
     -STARTMODE AUTO -PFILE ORACLE_HOME\DATABASE\INITSID.ORA
```

This syntax includes the following variables:

SID	is the same SID name as the SID of the database you are upgrading.
PASSWORD	is the password for the new release 9.0.1 database instance. This is the password for the user connected AS SYSDBA. The -INTPWD option is not required. If you do not specify it, then operating system authentication is used, and no password is required.
USERS	is the maximum number of users who can be granted SYSDBA and SYSOPER privileges.
ORACLE_HOME	is the release 9.0.1 Oracle home directory. Ensure that you specify the full pathname with the -PFILE option, including drive letter of the Oracle home directory.

For example, if your *SID* is ORCL, your *PASSWORD* is TWxy579, the maximum number of *USERS* is 10, and the *ORACLE_HOME* directory is `C:\ORA900`, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -INTPWD TWxy579 -MAXUSERS 10
     -STARTMODE AUTO -PFILE C:\ORA900\DATABASE\INITORCL.ORA
```

9. Copy configuration files to a location outside of the old Oracle home:

a. If your initialization parameter file resides within the old environment's Oracle home, then copy it to a location outside of the old environment's Oracle home. By default Oracle looks for the initialization parameter file in *`ORACLE_HOME`*`/dbs` on UNIX platforms and in *`ORACLE_HOME`*`\database` on Windows platforms. The initialization parameter file can reside anywhere you wish, but it should not reside in the old environment's Oracle home after you upgrade to the new release.

b. If your initialization parameter file has an `IFILE` (include file) entry and the file specified in the `IFILE` entry resides within the old environment's Oracle home, then copy the file specified by the `IFILE` entry to a location outside of the old environment's Oracle home. The file specified in the `IFILE` entry has additional initialization parameters.

c. If you have a password file that resides within the old Oracle home, then move or copy the password file to the Oracle9*i* Oracle home. The name and location of the password file are operating system-specific. On UNIX operating systems, the default password file is *`ORACLE_HOME`*`/dbs/orapw`*`sid`*. On Windows operating systems, the default password file is *`ORACLE_HOME`*`\database\pwd`*`sid`*`.ora`. On both UNIX and Windows operating systems, *sid* is your Oracle instance ID.

Note: For Oracle Parallel Server, perform this step on all nodes. Also, if your `init`*`db_name`*`.ora` file resides within the old environment's Oracle home, then move or copy the `init`*`db_name`*`.ora` file to a location outside of the old environment's Oracle home.

---

10. Adjust the initialization parameter file for use with the new release. Specifically, complete the following steps:

a. Remove obsolete parameters and adjust changed parameters. Certain initialization parameters are obsolete in the new 9.0.1 release. Remove all obsolete parameters from any initialization parameter file that will start a new release 9.0.1 instance. Obsolete parameters may cause errors in the new 9.0.1 release. Also, alter any parameter whose syntax has changed in the new 9.0.1 release; refer to Appendix B, "Changes to Initialization Parameters", for lists of new, renamed, and obsolete parameters.

Also, if you are using Oracle Parallel Server, then see *Oracle9i Real Application Clusters Installation and Configuration* for more information about obsolete Oracle Parallel Server initialization parameters.

b. If you are updating materialized views automatically by using the `JOB_QUEUE_PROCESSES` initialization parameter, then set this parameter to 0 (zero) in the initialization parameter file. Also, if you are using Advanced Queuing and have propagation schedules, then set both the `JOB_QUEUE_PROCESSES` and `AQ_TM_PROCESSES` initialization parameters to 0 (zero).

c. Make sure the `OPTIMIZER_MODE` initialization parameter is set to `choose`.

d. If you are using a password file, then set `REMOTE_LOGIN_PASSWORDFILE` to `NONE` in the initialization parameter file. After upgrading your database, you can change the settings for these parameters back to their normal settings.

e. On Windows platforms, change the `BACKGROUND_DUMP_DEST` and `USER_DUMP_DEST` initialization parameters that point to RDBMS80 or any other environment variable to point to the following directories instead:

Initialization Parameter	Change Setting To
BACKGROUND_DUMP_DEST	*ORACLE_BASE*\oradata\DB_NAME
USER_DUMP_DEST	*ORACLE_BASE*\oradata\DB_NAME\archive

In the settings, substitute the complete ORACLE_BASE path for *ORACLE_BASE* and substitute the database name for *DB_NAME*.

f. If the initialization parameter file contains an `IFILE` entry, then change the `IFILE` entry in the initialization parameter file to point to the new location you copied it to in Step 9. b. Then, edit the file specified in the `IFILE` entry in the same way that you edited the initialization parameter file in sub-steps a to e.

g. Make sure the `COMPATIBLE` initialization parameter is properly set for Oracle9*i*. If `COMPATIBLE` is set below 8.1.0, then you will encounter the following error when you attempt to start up your release 9.0.1 database later in step 16:

```
ORA-00401: the value for parameter compatible is not supported by this release
```

Either leave COMPATIBLE unset in your initialization parameter file or set COMPATIBLE to 8.1.x.

h. If you are using Oracle Parallel Server, then modify the init*db_name*.ora file in the same way that you modified the initialization parameter file.

Make sure you save all of the files you modified after making these adjustments.

Note: For Oracle Parallel Server, perform this step on all nodes. Also, set the CLUSTER_DATABASE initialization parameter to false. You can change it back to true after the upgrade operation is complete.

---

11. If your operating system is UNIX, then make sure that the following environment variables point to the new release 9.0.1 directories:

- ORACLE_HOME
- PATH
- ORA_NLS33
- LD_LIBRARY_PATH

Note: For Oracle Parallel Server, perform this step on all nodes.

See Also: Your operating system-specific Oracle9*i* installation documents for information about setting other important environment variables on your operating system.

12. Log in to the system as the owner of the Oracle home directory of the new release.

13. At a system prompt, change to the *ORACLE_HOME/*rdbms/admin directory.

14. Start SQL*Plus.

15. Connect to the database instance as a user with SYSDBA privileges.

16. Start up the instance in RESTRICT mode:

```
SQL> STARTUP RESTRICT
```

You may need to use the `PFILE` option to specify the location of your initialization parameter file.

You may see error messages listing obsolete initialization parameters. If so, then make a note of the obsolete initialization parameters and continue with the upgrade normally. Then, remove the obsolete initialization parameters the next time you shut down the database (Step 20).

17. Set the system to spool results to a log file for later verification of success:

```
SQL> SPOOL catoutu.log
```

If you want to see the complete detailed output of the script you will run, then you can also issue a `SET ECHO ON` statement:

```
SQL> SET ECHO ON
```

18. Run `u`*`old_release`*`.sql`, where *old_release* refers to the release you had installed prior to upgrading. See Table 7–2 to choose the correct script. Each script provides a direct upgrade from the release specified in the "Old Release" column. The "Old Release" is the release from which you are upgrading.

To run a script, enter the following:

```
SQL> @uold_release.sql
```

Table 7–2 Upgrade Scripts

Old Release	**Run Script**
8.0.6	`u0800060.sql`
8.1.5	`u0801050.sql`
8.1.6	`u0801060.sql`
8.1.7	`u0801070.sql`

Note: If the old release you had installed prior to upgrading is not listed in Table 7–2, then see the readme files in the new installation for the correct upgrade script to run.

Make sure you follow these guidelines when you run the script:

- You must use the version of the script supplied with the new release 9.0.1 installation.
- You must run the script in the new release 9.0.1 environment.
- You only need to run ONE script, even if your upgrade spans several releases. For example, if your old release was 8.1.5, then you need to run only `u0801050.sql`.

The script you run creates and alters certain dictionary tables. It also runs the `catalog.sql` and `catproc.sql` scripts that come with the release to which you are upgrading, which create the system catalog views and all the necessary packages for using PL/SQL.

If you encounter any problems when you run the script, or any of the scripts in the remaining steps, then correct the causes of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

> **See Also:** "Running Scripts" on page 1-11 for information about the types of errors to look for when you run a script.

You may encounter a series of messages similar to the following during the upgrade:

```
ORA-00604: error occurred at recursive SQL level 1
ORA-00001: unique constraint (SYSTEM.AQ$_QUEUES_CHECK) violated
ORA-06512: at "SYS.DBMS_AQADM", line 2023
ORA-06512: at line 2
```

You can ignore these messages.

19. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 17; the suggested name was `catoutu.log`. Correct any problems you find in this file and rerun the appropriate upgrade script if necessary.

If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

```
SQL> SET ECHO OFF
```

20. Run `SHUTDOWN` on the Oracle9*i* database:

```
SQL> SHUTDOWN IMMEDIATE
```

Caution: Use `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE`. Do not use `SHUTDOWN ABORT`.

Executing this clean shutdown flushes all caches, clears buffers, and performs other DBMS housekeeping activities. These measures are an important final step to ensure the integrity and consistency of the newly upgraded Oracle9*i* database.

Also, if you encountered a message listing obsolete initialization parameters when you started the database in Step 16, then remove the obsolete initialization parameters from the initialization parameter file now.

Your database is now upgraded to the new 9.0.1 release.

Caution: If you retain the old Oracle software, then never start the upgraded database with the old software. Only start the database with the executables in the new release 9.0.1 installation directory. Also, before you remove the old Oracle environment, make sure you relocate any datafiles in that environment to the new Oracle9*i* environment. See the *Oracle9i Database Administrator's Guide* for information about relocating datafiles.

21. Complete the procedures described in "Upgrading Specific Components" on page 7-24 and in Chapter 8, "After Migrating or Upgrading the Database".

Upgrading Specific Components

Some components of the Oracle database server require an upgrade operation separate from the general database upgrade operation. Complete the actions in the following sections to upgrade specific components:

- Upgrading JServer
- Upgrading XDK for Java
- Upgrading Session Namespace, CORBA, and OSE
- Upgrading JSP

- Upgrading Oracle Spatial
- Upgrading interMedia
- Upgrading Oracle Text
- Upgrading Oracle Label Security
- Upgrading Oracle9i Real Application Clusters
- Upgrading Materialized Views
- Upgrading the Advanced Queuing Option

See Also: Some of the following upgrade procedures involve Export/Import. See *Oracle9i Database Utilities* for Export/Import instructions.

Note: You should perform the actions described in these sections only after you have upgraded the database by following the instructions in "Upgrading the Database to the New Oracle9i Release" on page 7-2.

Upgrading JServer

If the Oracle system has the JServer component installed, then complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the new release.
2. At a system prompt, change to the *ORACLE_HOME*/`javavm/install` directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with `SYSDBA` privileges.
5. If the instance is running, shut it down using `SHUTDOWN IMMEDIATE`:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

Note: For Oracle9*i* Real Application Clusters, set the CLUSTER_DATABASE initialization parameter to false. You can change it back to true after the upgrade operation is complete.

---

6. Start up the instance in RESTRICT mode:

```
SQL> STARTUP RESTRICT
```

   You may need to use the PFILE option to specify the location of your initialization parameter file.

7. Set the system to spool results to a log file for later verification of success:

```
SQL> SPOOL catoutjava.log
```

   If you want to see the complete detailed output of the script you will run, then you can also issue a SET ECHO ON statement:

```
SQL> SET ECHO ON
```

8. Run the appropriate upgrade script depending on the release from which you are upgrading.

   If you are upgrading from release 8.1.5, run jvmu815.sql:

```
SQL> @jvmu815.sql
```

   If you are upgrading from release 8.1.6, run jvmu816.sql:

```
SQL> @jvmu816.sql
```

   If you are upgrading from release 8.1.7, run jvmu817.sql:

```
SQL> @jvmu817.sql
```

   After you run any one of these scripts, user classes that were present before the upgrade are left in place but are typically invalid. These classes are implicitly revalidated when they are used. You can explicitly revalidate any class by issuing an ALTER ANY CLASS statement. For example, to revalidate a class named MyClass, issue the following statement:

```
ALTER JAVA CLASS MyClass RESOLVE;
```

9. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 7; the suggested name was `catoutjava.log`. Correct any problems you find in this file and rerun the appropriate upgrade script if necessary.

If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

```
SQL> SET ECHO OFF
```

10. Run `ALTER SYSTEM DISABLE RESTRICTED SESSION`:

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

11. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

12. Exit SQL*Plus.

The JServer component is upgraded to the new release.

Upgrading XDK for Java

If the Oracle system has the XDK for Java component installed, then complete the following steps:

1. Make sure you have successfully upgraded JServer.

   See Also: "Upgrading JServer" on page 7-25 for more information.

2. Log in to the system as the owner of the Oracle home directory of the new release.
3. At a system prompt, change to the *`ORACLE_HOME`*`/xdk/admin` directory.
4. Start SQL*Plus.
5. Connect to the database instance as a user with `SYSDBA` privileges.
6. If the instance is running, shut it down using `SHUTDOWN IMMEDIATE`:

```
SQL> SHUTDOWN IMMEDIATE
```

Note: For Oracle9*i* Real Application Clusters, set the `CLUSTER_DATABASE` initialization parameter to `false`. You can change it back to `true` after the upgrade operation is complete.

7. Start up the instance in `RESTRICT` mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the `PFILE` option to specify the location of your initialization parameter file.

8. Run the appropriate upgrade script depending on the release from which you are upgrading.

   If you are upgrading from release 8.1.5, run `xmlu815.sql`:

   ```
   SQL> @xmlu815.sql
   ```

   If you are upgrading from release 8.1.6, run `xmlu816.sql`:

   ```
   SQL> @xmlu816.sql
   ```

   If you are upgrading from release 8.1.7, run `xmlu817.sql`:

   ```
   SQL> @xmlu817.sql
   ```

9. Shut down the instance:

   ```
   SQL> SHUTDOWN
   ```

10. Exit SQL*Plus.

The XDK for Java component is upgraded to the new release.

Upgrading Session Namespace, CORBA, and OSE

1. Make sure you have successfully upgraded JServer and XDK for Java.
2. Log in to the system as the owner of the Oracle home directory of the new release.
3. At a system prompt, change to the *`ORACLE_HOME/`*`javavm/install` directory.
4. Start SQL*Plus.

5. Connect to the database instance as a user with SYSDBA privileges.
6. If the instance is running, shut it down using SHUTDOWN IMMEDIATE:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

   Note: For Oracle9*i* Real Application Clusters, set the CLUSTER_DATABASE initialization parameter to false. You can change it back to true after the upgrade operation is complete.

   ---

7. Start up the instance in RESTRICT mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.
8. Make sure you have at least 100 MB of free rollback segment space.

   See Also: *Oracle9i Database Administrator's Guide* for information about managing rollback segments.

9. Start the listener.
10. Run the appropriate upgrade script depending on the release from which you are upgrading.

    If you are upgrading from release 8.1.5, run jisu815.sql:

    ```
    SQL> @jisu815.sql
    ```

    If you are upgrading from release 8.1.6, run jisu816.sql:

    ```
    SQL> @jisu816.sql
    ```

    If you are upgrading from release 8.1.7, run jisu817.sql:

    ```
    SQL> @jisu817.sql
    ```

11. Shut down the instance:

    ```
    SQL> SHUTDOWN
    ```

12. Exit SQL*Plus.

Note: Upgrade of Enterprise JavaBeans is not supported. If you deployed Enterprise JavaBeans in a past release, then you need to redeploy it for release 9.0.1. See the *Oracle9i Enterprise JavaBeans Developer's Guide and Reference* for information.

Upgrading JSP

If the Oracle system has JSP installed, then complete the following steps:

1. Make sure you have successfully upgraded JServer, XDK for Java, and Session Namespace, CORBA, and OSE.
2. Log in to the system as the owner of the Oracle home directory of the new release.
3. At a system prompt, change to the *`ORACLE_HOME/`*`javavm/install` directory.
4. Start SQL*Plus.
5. Connect to the database instance as a user with `SYSDBA` privileges.
6. If the instance is running, shut it down using `SHUTDOWN IMMEDIATE`:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

   Note: For Oracle9*i* Real Application Clusters, set the `CLUSTER_DATABASE` initialization parameter to `false`. You can change it back to `true` after the upgrade operation is complete.

7. Start up the instance in `RESTRICT` mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the `PFILE` option to specify the location of your initialization parameter file.

8. Make sure you have at least 100 MB of free rollback segment space.

   See Also: *Oracle9i Database Administrator's Guide* for information about managing rollback segments.

9. Run the appropriate upgrade script depending on the release from which you are upgrading.

   If you are upgrading from release 8.1.5, run `jspu815.sql`:

   ```
   SQL> @jspu815.sql
   ```

   If you are upgrading from release 8.1.6, run `jspu816.sql`:

   ```
   SQL> @jspu816.sql
   ```

   If you are upgrading from release 8.1.7, run `jspu817.sql`:

   ```
   SQL> @jspu817.sql
   ```

10. Shut down the instance:

    ```
    SQL> SHUTDOWN
    ```

11. Exit SQL*Plus.

Upgrading Oracle Spatial

If the Oracle system has Oracle Spatial installed, then see the *Oracle Spatial User's Guide and Reference* for instructions about upgrading Oracle Spatial to release 9.0.1.

Upgrading *inter*Media

Manual upgrade instructions for *inter*Media can be found in *ORACLE_HOME*`/ord/im/admin/README.txt` on UNIX platforms and in *ORACLE_HOME*`\ord\im\admin\README.txt` on Windows platforms.

Manual upgrade instructions for customers with existing Visual Information Retrieval applications who wish to upgrade to Oracle9*i* Visual Information Retrieval-compatible API can be found in *ORACLE_HOME*`/ord/vir/admin/README.txt` on UNIX platforms and in *ORACLE_HOME*`\ord\vir\admin\README.txt` on Windows platforms.

Upgrading Oracle Text

If the Oracle system has Oracle Text installed, then complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the new release.
2. At a system prompt, change to the *ORACLE_HOME*`/ctx/admin` directory.

3. Start SQL*Plus.
4. Connect to the database instance as a user with SYSDBA privileges.
5. If the instance is running, shut it down using SHUTDOWN IMMEDIATE:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start up the instance in RESTRICT mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.
7. Set the system to spool results to a log file for later verification of success:

   ```
   SQL> SPOOL catouttext.log
   ```

   If you want to see the complete detailed output of the script you will run, then you can also issue a SET ECHO ON statement:

   ```
   SQL> SET ECHO ON
   ```

8. If you are upgrading from release 8.1.5, then complete the following steps. Skip to Step 9 if you are upgrading from release 8.1.6. Skip to step 10 if you are upgrading from release 8.1.7.
   a. Run s0801060.sql:

      ```
      SQL> @s0801060.sql
      ```

   b. Connect to the database instance as user CTXSYS.
   c. Run u0801060.sql:

      ```
      SQL> @u0801060.sql
      ```

   d. Connect to the database instance as a user with SYSDBA privileges.
9. If you are upgrading from release 8.1.6, then complete the following steps. Skip to Step 10 if you are upgrading from release 8.1.7.
   a. Run s0801070.sql:

      ```
      SQL> @s0801070.sql
      ```

      This script grants new, required database privileges to user CTXSYS.
   b. Connect to the database instance as user CTXSYS.

   c. Run `u0801070.sql`:

      ```
      SQL> @u0801070.sql
      ```

      This script upgrades the `CTXSYS` schema to release 8.1.7.

   d. Connect to the database instance as a user with `SYSDBA` privileges.

10. If you are upgrading from release 8.1.7, then complete the following steps.

    a. Run `s0900010.sql`:

       ```
       SQL> @s0900010.sql
       ```

       This script grants new, required database privileges to user `CTXSYS`.

    b. Connect to the database instance as user `CTXSYS`.

    c. Run `u0900010.sql`:

       ```
       SQL> @u0900010.sql
       ```

       This script upgrades the `CTXSYS` schema to release 9.0.1.

    d. Connect to the database instance as a user with `SYSDBA` privileges.

11. Check for any invalid `CTXSYS` objects and alter compile as needed.

12. Turn off the spooling of script results to the log file:

    ```
    SQL> SPOOL OFF
    ```

    Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 7; the suggested name was `catouttext.log`. Correct any problems you find in this file and rerun the appropriate upgrade scripts if necessary.

    If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

    ```
    SQL> SET ECHO OFF
    ```

13. Run `ALTER SYSTEM DISABLE RESTRICTED SESSION`:

    ```
    SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
    ```

14. Shut down the instance:

    ```
    SQL> SHUTDOWN IMMEDIATE
    ```

15. Exit SQL*Plus.

Oracle Text is upgraded to the new release.

Upgrading Oracle Label Security

If the Oracle system has Oracle Label Security installed, and you did not choose to upgrade Oracle Label Security using the Oracle Data Migration Assistant, then complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the new release.
2. At a system prompt, change to the *ORACLE_HOME*/rdbms/admin directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with SYSDBA privileges.
5. If the instance is running, shut it down using SHUTDOWN IMMEDIATE:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start up the instance in RESTRICT mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.
7. Set the system to spool results to a log file for later verification of success:

   ```
   SQL> SPOOL catoutols.log
   ```

   If you want to see the complete detailed output of the script you will run, then you can also issue a SET ECHO ON statement:

   ```
   SQL> SET ECHO ON
   ```

8. Run the appropriate upgrade script depending on the release from which you are upgrading.

   If you are upgrading from release 8.1.7, run olsu817.sql:

   ```
   SQL> @olsu817.sql
   ```

9. Turn off the spooling of script results to the log file:

   ```
   SQL> SPOOL OFF
   ```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 7; the suggested name was `catoutols.log`. Correct any problems you find in this file and rerun the appropriate upgrade script if necessary.

If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

```
SQL> SET ECHO OFF
```

10. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

11. Exit SQL*Plus.

Oracle Label Security is upgraded to the new release.

Upgrading Oracle9*i* Real Application Clusters

If the Oracle system has Oracle9*i* Real Application Clusters installed, then complete the following steps:

1. Shut down all instances using `SHUTDOWN IMMEDIATE`:

```
SQL> SHUTDOWN IMMEDIATE
```

> **Note:** For Oracle9*i* Real Application Clusters, set the `CLUSTER_DATABASE` initialization parameter to `false`. You can change it back to `true` after the upgrade operation is complete.

2. Log in to the system as the owner of the Oracle home directory of the new release.
3. At a system prompt, change to the *`ORACLE_HOME`*`/rdbms/admin` directory.
4. Start SQL*Plus.
5. Connect to the database instance as a user with `SYSDBA` privileges.
6. Run `STARTUP RESTRICT`:

```
SQL> STARTUP RESTRICT
```

You may need to use the `PFILE` option to specify the location of your initialization parameter file.

7. Set the system to spool results to a log file for later verification of success:

   ```
   SQL> SPOOL catoutclust.log
   ```

   If you want to see the output of the script you will run on your screen, then you can also issue a `SET ECHO ON` statement:

   ```
   SQL> SET ECHO ON
   ```

8. Run `catclust.sql`:

   ```
   SQL> @catclust.sql
   ```

9. Turn off the spooling of script results to the log file:

   ```
   SQL> SPOOL OFF
   ```

   Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 7; the suggested name was `catoutclust.log`. Correct any problems you find in this file.

   If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

   ```
   SQL> SET ECHO OFF
   ```

10. Run `ALTER SYSTEM DISABLE RESTRICTED SESSION`:

    ```
    SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
    ```

Oracle9*i* Real Application Clusters is upgraded to the new release.

Upgrading Materialized Views

Note: The word "snapshot" is synonymous with the word "materialized view".

Materialized views upgraded from release 8.0 or imported from a release 8.0 database cannot use the new summary management features available in release 8.1 and higher. If you want to use these new features, then complete the following steps for each materialized view and for each materialized view imported from release 8.0:

1. GRANT QUERY REWRITE privileges to the owner of the materialized view. Only local materialized views are available for query rewrite.

If the materialized view references any schema objects outside its owner's schema, then you must issue a `GRANT GLOBAL QUERY REWRITE` statement.

2. Issue the `ALTER MATERIALIZED VIEW ... ENABLE QUERY REWRITE` statement on the materialized views you want to upgrade.

   For example, on a materialized view named `SSORDERS`, issue the following statement:

   ```
   ALTER MATERIALIZED VIEW ssorders ENABLE QUERY REWRITE;
   ```

In addition, if you do not `ENABLE QUERY REWRITE` on a materialized view, then the `ATOMIC=FALSE` option of the `DBMS_MVIEW.REFRESH` procedure may not work unless you issue an `ALTER MATERIALIZED VIEW ... COMPILE` statement on the materialized view. For example, for a materialized view named `SSCUST`, issue the following statement:

```
ALTER MATERIALIZED VIEW sscust COMPILE;
```

You do not need to issue this statement if you have issued any other `ALTER MATERIALIZED VIEW` statement on the materialized view, such as the `ALTER MATERIALIZED VIEW ... ENABLE QUERY REWRITE` statement.

Upgrading the Advanced Queuing Option

The following sections describe the actions required to upgrade the Advanced Queuing (AQ) option.

Upgrade Your Queue Tables

The following release 8.1 and higher AQ enhancements are available only if you upgrade your existing queue tables:

- Addition of the original message ID column for propagated messages
- Addition of a sender's ID column
- Queue and system level privileges
- Rule based subscriptions
- Separate storage of history management information, which was stored in a varray in release 8.0

To upgrade an existing queue table, run the `DBMS_AQADM.MIGRATE_QUEUE_TABLE` procedure, specifying 8.1 for the option. For example, for a queue table named `tb_queue` owned by user `scott`, run the following procedure:

```
EXECUTE dbms_aqadm.migrate_queue_table (
    queue_table => 'scott.tb_queue',
    compatible => '8.1');
```

To create a new queue table that is compatible with release 8.1 and higher, connect as the owner of the queue table and run the `DBMS_AQADM.CREATE_QUEUE_TABLE` procedure, specifying 8.1 for the `COMPATIBLE` option, as in the following example:

```
EXECUTE dbms_aqadm.create_queue_table(
    queue_table => 'scott.tkaqqtpeqt',
    queue_payload_type =>'message',
    sort_list => 'priority,enq_time',
    multiple_consumers => true,
    comment => 'Creating queue with priority and enq_time sort order',
    compatible => '8.1');
```

Note: The `COMPATIBLE` initialization parameter must be set to 8.1.0 or higher to upgrade your queue tables and to create new release 8.1 compatible queue tables.

Upgrading the Recovery Catalog

Your recovery catalog schema for the upgraded database may reside in a database that is separate from the database you upgraded. If you upgraded the Recovery Manager executable to release 8.1, then you must upgrade the recovery catalog to release 8.1 as well.

Also, if you have multiple databases of different releases managed by a single recovery catalog, then you need to consider compatibility issues between a particular Recovery Manager release and the recovery catalog release. For example, release 8.1.3 and 8.1.4 of Recovery Manager cannot access a release 8.1.5 or higher recovery catalog. Therefore, in this case, you must upgrade all of the databases managed by the recovery catalog to release 8.1.5 or higher. For more information about recovery catalog compatibility with Recovery Manager, see "Recovery Manager" on page 9-44.

Complete the following steps to upgrade the recovery catalog:

1. Log in to Recovery Manager and connect to the recovery catalog.

   For example, if RCAT/RCAT is the user name and password for the recovery catalog owner, and RECDB is the network service name, then enter the following:

```
rman rcvcat rcat/rcat@recdb
```

The first time you connect to an older recovery catalog with the 8.1 release of Recovery Manager, you will see message RMAN-06186, indicating that the recovery catalog must be upgraded.

2. Use the UPGRADE CATALOG command to upgrade the recovery catalog to the most current release. Recovery Manager prompts you to enter the command twice to confirm the catalog upgrade. If any errors are encountered while upgrading, then they are displayed in the Recovery Manager log.

   Here is the log from a session that upgrades the recovery catalog from release 8.0.4:

```
Recovery Manager: Release 8.1.5.0.0

RMAN-06008: connected to recovery catalog database
RMAN-06186: PL/SQL package rcat.DBMS_RCVCAT version 08.00.04 in RCVCAT
database is too old

RMAN> upgrade catalog

RMAN-06435: recovery catalog owner is rcat
RMAN-06442: enter UPGRADE CATALOG command again to confirm catalog upgrade

RMAN> upgrade catalog

RMAN-06408: recovery catalog upgraded to version 08.01.05
```

Upgrading Statistics Tables Created by the DBMS_STATS Package

If you created statistics tables using the DBMS_STATS.CREATE_STAT_TABLE procedure, then upgrade these tables by executing the following procedure:

```
EXECUTE DBMS_STATS.UPGRADE_STAT_TABLE('scott', 'stat_table');
```

where SCOTT is the owner of the statistics table and STAT_TABLE is the name of the statistics table. Execute this procedure for each statistics table.

Recompiling Invalid PL/SQL Modules

The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an INVALID state, such as packages, procedures, types, and so on. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

To run the `utlrp.sql` script, complete the following steps:

1. At a system prompt, change to the `ORACLE_HOME/rdbms/admin` directory.
2. Start SQL*Plus.
3. Connect to the database instance AS SYSDBA.
4. Run `utlrp.sql`:

   ```
   SQL> @utlrp.sql
   ```

Oracle Corporation highly recommends running `utlrp.sql`.

Changing the Word-Size of Your Current Release

The instructions in this section guide you through changing the word-size of your current release (switching from 32-bit software to 64-bit software or vice versa).

> **See Also:** "Changing Word-Size" on page 1-12 for more information about changing word-size.

Complete the following steps to change the word-size of your current release:

1. Start SQL*Plus.
2. Connect to the database instance AS SYSDBA.
3. Run `SHUTDOWN IMMEDIATE` on the database:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

   > **Note:** For Oracle9*i* Real Application Clusters, issue this statement for all instances. Also, set the `PARALLEL_SERVER` initialization parameter to `false`. You can change it back to `true` after the upgrade operation is complete.

4. Perform a full offline backup of the database.

   > **See Also:** *Oracle9i User-Managed Backup and Recovery Guide* for more information.

5. If you are using the same Oracle home for your current release and the release to which you are switching, then deinstall your current release using the Oracle Installer. You do not need to deinstall your current release if you are using separate Oracle home directories.
6. If you currently have a 32-bit installation, then install the 64-bit version of the same release. Or, if you currently have a 64-bit installation, then install the 32-bit version of the same release.

> **Note:** Installation and deinstallation are operating system-specific. For installation and deinstallation instructions, see your Oracle9*i* operating system-specific installation documentation and the Oracle9*i* README for your operating system.

7. Copy configuration files to a location outside of the old Oracle home:
   a. If your initialization parameter file resides within the old environment's Oracle home, then copy it to a location outside of the old environment's Oracle home. The initialization parameter file can reside anywhere you wish, but it should not reside in the old environment's Oracle home after you switch to the new release.
   b. If your initialization parameter file has an IFILE (include file) entry and the file specified in the IFILE entry resides within the old environment's Oracle home, then copy the file specified by the IFILE entry to a location outside of the old environment's Oracle home. The file specified in the IFILE entry has additional initialization parameters. After you copy this file, edit the IFILE entry in the initialization parameter file to point to its new location.
   c. If you have a password file that resides within the old Oracle home, then move or copy the password file to the Oracle9*i* Oracle home. The name and location of the password file are operating system-specific; for example, on UNIX operating systems, the default password file is `ORACLE_HOME/dbs/orapwsid`, but on Windows platforms, the default password file is `ORACLE_HOME\database\pwdsid.ora`. In both cases, *sid* is your Oracle instance ID.

Note: For Oracle9*i* Real Application Clusters, perform this step on all nodes. Also, if your `init`*`db_name`*`.ora` file resides within the old environment's Oracle home, then move or copy the `init`*`db_name`*`.ora` file to a location outside of the old environment's Oracle home.

8. At a system prompt, change to the *`ORACLE_HOME/`*`rdbms/admin` directory.
9. Start SQL*Plus.
10. Connect to the database instance AS SYSDBA.
11. Run `STARTUP RESTRICT`:

    ```
    SQL> STARTUP RESTRICT
    ```

    You may need to use the `PFILE` option to specify the location of your initialization parameter file.
12. Set the system to spool results to a log file for later verification of success:

    ```
    SQL> SPOOL catoutw.log
    ```

    If you want to see the output of the script you will run on your screen, then you can also issue a `SET ECHO ON` statement:

    ```
    SQL> SET ECHO ON
    ```

13. Run `utlirp.sql`:

    ```
    SQL> @utlirp.sql
    ```

    The `utlirp.sql` script recompiles existing PL/SQL modules in the format required by the new database. This script first alters certain dictionary tables. Then, it reloads package STANDARD and DBMS_STANDARD, which are necessary for using PL/SQL. Finally, it triggers a recompile of all PL/SQL modules, such as packages, procedures, types, and so on.
14. Turn off the spooling of script results to the log file:

    ```
    SQL> SPOOL OFF
    ```

    Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 12; the suggested name was `catoutw.log`. Correct any problems you find in this file.

If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

```
SQL> SET ECHO OFF
```

15. Run `ALTER SYSTEM DISABLE RESTRICTED SESSION`:

```
SQL> ALTER SYSTEM DISABLE RESTRICTED SESSION;
```

The word-size of your database is changed. You can open the database for normal use.

8 After Migrating or Upgrading the Database

This chapter guides you through the procedures to perform after you have completed a migration or upgrade of your database. If you migrated your database, then complete these tasks regardless of the method you used, including the following methods: the Migration utility, the Oracle Data Migration Assistant, Export/Import, or data copy. Similarly, if you upgraded your database, then complete these procedures regardless of the method you used, including the following methods: the Oracle Data Migration Assistant or manual upgrade.

This chapter covers the following topics:

Tasks to Complete After Migrating or Upgrading Your Database

Complete the following tasks after you have migrated or upgraded your database.

Back Up the Database

Make sure you perform a complete backup of the production database. This backup must be complete, including all datafiles, control files, online redo log files, parameter files, and SQL scripts that create objects in the new database. To accomplish a complete backup, a full database export or a cold backup is required, because a hot backup cannot afford full recoverability. This backup can be used as a return point, if necessary, in case subsequent steps adversely affect the database.

See Also: *Oracle9i User-Managed Backup and Recovery Guide* for details about backing up the database

---

Note: Using the Migration utility transforms the source database. Therefore, after migration, the source database ceases to exist except for the backup you created under "Preserve the Oracle7 Source Database" on page 4-26. This backup also can serve as the first Oracle9*i* backup for a recovery of the newly migrated database.

Change the Passwords for Oracle-Supplied Accounts

Depending on the release from which you migrated or upgraded, there may be some new Oracle-supplied accounts. Oracle Corporation recommends that you lock all Oracle-supplied accounts except for `SYS` and `SYSTEM`, and expire their passwords, requiring new passwords to be specified if the accounts are unlocked.

You can view the status of all accounts by issuing the following SQL statement:

```
SQL> SELECT username, account_status
   FROM dba_users
      ORDER BY username;
```

To LOCK and EXPIRE passwords, issue the following SQL statement:

```
ALTER USER username PASSWORD EXPIRE ACCOUNT LOCK;
```

Upgrade User NCHAR Columns

If you upgraded from a version 8 release and your database contains user tables with NCHAR columns, you must upgrade the NCHAR columns before they can be used in Oracle9*i*.

The following steps convert your NCHAR columns from the old format and character set to the new Oracle9*i* format. In addition, if your old National Character Set was UTF8, it will remain UTF8 in Oracle9*i*. However, your National Character Set will be converted to AL16UTF16 if it was not UTF8 in the old release.

You can override the default upgrade selection of the National Character Set. That is, a version 8 UTF8 National Character Set can be converted to an Oracle9*i* AL16UTF16 National Character Set or a version 8 non-UTF8 National Character Set can be converted to an Oracle9*i* UTF8 National Character Set.

You will encounter the following error when attempting to use the NCHAR columns in Oracle9*i* until you perform the steps in this section:

```
ORA-12714: invalid national character set specified
```

Note: Once you upgrade your NCHAR columns, you will not be able to downgrade to a previous release of Oracle until all NCHAR columns have been dropped.

To upgrade user tables with NCHAR columns, perform the following steps:

1. Log in to the system as the owner of the Oracle home directory.
2. At a system prompt, change to the *ORACLE_HOME*/rdbms/admin directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with SYSDBA privileges.
5. If the instance is running, shut it down using SHUTDOWN IMMEDIATE:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start up the instance in RESTRICT mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.

7. Run `utlnchar.sql`:

   ```
   SQL> @utlnchar.sql
   ```

   Alternatively, to override the default upgrade selection, run `n_switch.sql`:

   ```
   SQL> @n_switch.sql
   ```

8. Shut down the instance:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

9. Exit SQL*Plus.

Downgrading SQL NCHAR Columns

Once you have upgraded your SQL NCHAR columns (NCHAR, NVARCHAR2, and NCLOB) to Oracle9*i*, you will not be able to downgrade to a previous release until all SQL NCHAR columns have been dropped. If you need to recover the version 8 SQL NCHAR data, you will need to reimport the data from a previous backup.

Migrate Your Initialization Parameter File to a Server Parameter File

If you are currently using a traditional initialization parameter file, perform the following steps to migrate to a server parameter file:

1. If the initialization parameter file is located on a client machine, transfer the file from the client machine to the server machine.

   Note: If you are using Oracle9*i* Real Application Clusters, you must combine all of your instance-specific initialization parameter files into a single initialization parameter file. Instructions for doing this, and other actions unique to using a server parameter file for cluster databases, are discussed in:

   - *Oracle9i Real Application Clusters Installation and Configuration*
   - *Oracle9i Real Application Clusters Administration*

2. Create a server parameter file using the `CREATE SPFILE` statement. This statement reads the initialization parameter file to create a server parameter file. The database does not have to be started to issue a `CREATE SPFILE` statement.

3. Start up the instance using the newly created server parameter file.

   See Also:

   - *Oracle9i Database Administrator's Guide* for more information about creating server parameter files
   - *Oracle9i SQL Reference* for information about the `CREATE SPFILE` statement

Install Component Java Classes

The components listed in this section require the Java option. If you did not have Java installed in a previous release, then complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the new release.
2. At a system prompt, change to the *`ORACLE_HOME/`*`javavm/install` directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with `SYSDBA` privileges.
5. If the instance is running, shut it down using `SHUTDOWN IMMEDIATE`:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start up the instance in `RESTRICT` mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the `PFILE` option to specify the location of your initialization parameter file.
7. Run `initjvm.sql`:

   ```
   SQL> @initjvm.sql
   ```

8. Shut down the instance:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

9. Exit SQL*Plus.

The following components require initialization that is separate from the procedures discussed in "Upgrading JServer" on page 7-25:

- Oracle Change Data Capture
- Summary Advisor
- ODCI and CartridgeServices
- SQLJ

To install one or more of these components, complete the following steps:

1. Log in to the system as the owner of the Oracle home directory of the new release.
2. At a system prompt, change to the *ORACLE_HOME/*rdbms/admin directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with SYSDBA privileges.
5. If the instance is running, shut it down using SHUTDOWN IMMEDIATE:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

6. Start up the instance in RESTRICT mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.

7. Run one or more of the following scripts, depending on the components you are installing.

   If you are installing Oracle Change Data Capture, run initcdc.sql:

   ```
   SQL> @initcdc.sql
   ```

   If you are installing Summary Advisor, run initqsma.sql:

   ```
   SQL> @initqsma.sql
   ```

   If you are installing ODCI and CartridgeServices, run initsoxx.sql:

   ```
   SQL> @initsoxx.sql
   ```

   If you are installing SQLJ, run initsjty.sql:

   ```
   SQL> @initsjty.sql
   ```

8. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

9. Exit SQL*Plus.

Migrate Tables from LONGs to LOBs

LOB datatypes (BFILE, BLOB, CLOB, and NCLOB) can provide many advantages over LONG datatypes. See *Oracle9i Database Concepts* for information about the differences between LONG and LOB datatypes.

Change LONGs to LOBs

In Oracle9*i*, the ALTER TABLE statement can be used to change the datatype of a LONG column to CLOB and that of a LONG RAW column to BLOB.

In the following example, the LONG column named long_col in table long_tab is changed to datatype CLOB:

```
ALTER TABLE Long_tab MODIFY ( long_col CLOB );
```

After using this method to change LONG columns to LOBs, all the existing constraints and triggers on the table will still be usable. However, all the indexes, including Domain indexes and Functional indexes, on all columns of the table will become unusable and will have to be rebuilt using an ALTER INDEX ... REBUILD statement. Also, the Domain indexes on the LONG column will have to be dropped before changing the LONG column to a LOB.

See Also: *Oracle9i Application Developer's Guide - Large Objects (LOBs)* for information about modifying applications to use LOB data

Copy LONGs to LOBs

In release 8.1, the TO_LOB SQL function copies data from a LONG column in a table to a LOB column. The datatype of the LONG and LOB must correspond for a successful copy. For example, LONG RAW data must be copied to BLOB data, and LONG data must be copied to CLOB data.

In the examples in the following procedure, the LONG column named long_col in table long_tab is copied to a LOB column named lob_col in table lob_tab. These tables include an id column that contains identification numbers for each row in the table.

Complete the following steps to copy data from a LONG column to a LOB column:

1. Create a new table with the same definition as the table that contains the LONG column, but use a LOB datatype in place of the LONG datatype.

   For example, suppose you have a table with the following definition:

   ```
   CREATE TABLE long_tab (
       id NUMBER,
       long_col LONG);
   ```

   Create a new table using the following SQL statement:

   ```
   CREATE TABLE lob_tab (
       id NUMBER,
       clob_col CLOB);
   ```

   Note: When you create the new table, make sure you preserve the table's schema, including integrity constraints, triggers, grants, and indexes. The TO_LOB function only copies data; it does not preserve the table's schema.

2. Issue an INSERT statement using the TO_LOB function to insert the data from the table with the LONG datatype into the table with the LOB datatype.

   For example, issue the following SQL statement:

   ```
   INSERT INTO lob_tab
       SELECT id,
       TO_LOB(long_col)
       FROM long_tab;
   ```

3. When you are certain that the copy was successful, drop the table with the LONG column.

   For example, issue the following SQL statement to drop the long_tab table:

   ```
   DROP TABLE long_tab;
   ```

4. Create a synonym for the new table using the name of the table with LONG data. The synonym ensures that your database and applications continue to function properly.

   For example, issue the following SQL statement:

   ```
   CREATE SYNONYM long_tab FOR lob_tab;
   ```

Once the copy is complete, any applications that use the table must be modified to use the LOB data.

> **See Also:** *Oracle9i Application Developer's Guide - Large Objects (LOBs)* for information about modifying applications to use LOB data.

Check for Bad Date Constraints

A bad date constraint involves invalid date manipulation, which is a date manipulation that implicitly assumes the century in the date, causing problems at the year 2000. The `utlconst.sql` script runs through all of the check constraints in the database and marks constraints as bad if they include any invalid date manipulation. This script selects all the bad constraints at the end. Oracle7 allowed you to create constraints with a two-digit year date constant. However, release 8.0 and higher returns an error if the check constraint date constant does not include a four-digit year.

To run the `utlconst.sql` script, complete the following steps:

1. At a system prompt, change to the *ORACLE_HOME/*`rdbms/admin` directory.
2. Start SQL*Plus.
3. Connect to the database instance as a user with SYSDBA privileges.
4. Enter the following:

   ```
   SQL> SPOOL utlresult.log
   SQL> @utlconst.sql
   SQL> SPOOL OFF
   ```

After you run the script, the `utlresult.log` log file includes all the constraints that have invalid date constraints.

> **Note:** The `utlconst.sql` script does not correct bad constraints, but instead it disables them. You should either drop the bad constraints or recreate them after you make the necessary changes.

Migrate Your Server Manager Line Mode Scripts to SQL*Plus

Oracle9*i* no longer supports the use of Server Manager. If you run SQL scripts using Server Manager line mode, you must modify these scripts so that they are compatible with SQL*Plus. Chapter 11, "Migrating from Server Manager to

SQL*Plus" contains instructions for modifying your Server Manager line mode scripts to work with SQL*Plus.

Avoid Problems with Parallel Execution

Starting with release 8.1, parallel execution message buffers can be allocated from the large pool. In past releases, this allocation was from the shared pool. To avoid problems resulting from this change, you may need to adjust the following initialization parameters in your initialization parameter file:

- SHARED_POOL_SIZE
- LARGE_POOL_SIZE

See Also: "Parallel Execution Allocated from Large Pool" on page B-12 for information about adjusting these parameters.

Modify Your listener.ora File

You need to modify your `listener.ora` file only if one of the following conditions is true:

- You did not use the Oracle Data Migration Assistant to migrate or upgrade your database.
- You used the Oracle Data Migration Assistant to migrate or upgrade your database but chose not to have the `listener.ora` file updated automatically.

If neither of these conditions is true, then skip this section. If one of these conditions is true, then you need to modify your `listener.ora` file.

See Also: "listener.ora" on page F-14 for information about modifying your `listener.ora` file.

Migrate or Upgrade Your Standby Database

If you are using a standby database, then the primary and standby databases must run the same maintenance release of Oracle, and both databases must have the same setting for the `COMPATIBLE` initialization parameter.

For example, if your primary database is running release 8.1.7 with the `COMPATIBLE` initialization parameter set to 8.1.0, then the standby database can run any production 8.1 release, such as release 8.1.5, 8.1.6, or 8.1.7, as long as it also has `COMPATIBLE` set to 8.1.0. However, in this case, the standby database cannot run

Oracle7 or release 8.0, and `COMPATIBLE` cannot be set to any value other than 8.1.0 for the standby database.

To migrate the standby database from Oracle7 to Oracle9*i*, or to upgrade the standby database from version 8 to Oracle9*i*, perform the following steps:

1. Apply all redo logs created under Oracle7 or version 8 to the standby database.
2. Make sure the primary database has been opened successfully under Oracle9*i*.
3. Install the release 9.0.1 Oracle software on the computer that is running the standby database.
   - Install the new release of Oracle in a new Oracle home that is separate from the old release of Oracle.
   - At the Installation Types screen of the Oracle Universal Installer, choose Custom. Do not choose Standard Edition or Enterprise Edition because either of these choices will install a starter database along with your Oracle software. You can avoid installing a starter database if you select Custom.
   - At the Upgrading or Migrating an Existing Database screen of the Oracle Universal Installer, leave the Upgrade or Migrate an Existing Database check box unselected.
   - If the Oracle Universal Installer asks you whether you want to create a starter database, instruct it not to create a starter database.

   See Also: The Oracle installation documentation for your operating system for information about installing Oracle.

4. Complete the following additional steps if you migrated from Oracle7. These steps are not required if you upgraded from a version 8 release.

   a. Create a control file for your standby database by issuing an `ALTER DATABASE CREATE STANDBY CONTROLFILE AS` *`file_name`* statement, which creates a modified copy of the primary database's control file.

   b. Transfer the control file you created in Step a to the standby database site.

   c. Use your operating system to create a copy of the initial datafile at your primary database. If you are unsure about which datafile is the initial datafile, the following SQL statement provides that information:

   ```
   SELECT file_name, file_id FROM dba_data_files
     WHERE file_id = 1;
   ```

d. Transfer the copy of the initial datafile you made in Step c from the primary database to the standby database.

e. Deinstall the old Oracle software on the standby database computer (optional).

See Also: *Oracle9i Data Guard Concepts and Administration* for more information about standby database.

Add New Features as Appropriate

Oracle9i Database New Features describes many of the new features available in Oracle9*i*. Determine which of these new features can benefit the database and applications; then, develop a plan for using these features.

It is not necessary to make any immediate changes to begin using your Oracle9*i* database. You may prefer to introduce these enhancements into your database and corresponding applications gradually.

Chapter 10, "Upgrading Your Applications" describes ways to enhance your applications so that you can take advantage of the new Oracle9*i* features. However, before you implement new Oracle9*i* features, test your applications and successfully run them with the database you migrated or upgraded.

Develop New Administrative Procedures as Needed

After familiarizing yourself with the Oracle9*i* features, review your database administration scripts and procedures to determine whether any changes are necessary.

Coordinate your changes to the database with the changes that are necessary for each application. For example, by enabling integrity constraints in the database, you may be able to remove some data checking from your applications.

Adjust Your Initialization Parameter File for the New Release

Each new release of Oracle introduces new initialization parameters, changes some parameters, and obsoletes some parameters. You should adjust your initialization parameter file to account for these changes and to take advantage of new initialization parameters that may be beneficial to your system.

> **See Also:** Appendix B, "Changes to Initialization Parameters" for lists of the new, changed, and obsoleted initialization parameters in release 9.0.1, and *Oracle9i Database Reference* for detailed information about each parameter.

The `COMPATIBLE` initialization parameter controls the compatibility level of your database. Set the `COMPATIBLE` initialization parameter in your initialization parameter file based on the compatibility level you want for your new database.

> **See Also:** "Setting the COMPATIBLE Initialization Parameter" on page 9-7 for information.

Normalize Filenames on Windows Platforms

You only need to normalize filenames if you are running Oracle on a Windows platform. You do not need to perform these steps on UNIX operating systems.

The control file and the recovery catalog both store filenames so that they can access files that are required by the database, such as:

- Datafiles
- Control files
- Online and archived redo logs used by Oracle
- Datafile copies and on-disk backup pieces used by Recovery Manager

In releases prior to release 8.1.6 on Windows platforms, a flawed filename normalization mechanism allowed two different filenames to refer to the same physical file. For example, because of this flaw, Oracle may not record the fully specified pathname for a file in the control file. That is, Oracle may record only `dbfile1.dbf` instead of `c:\oracle\oradata\dbfile1.dbf`. If this happens, then, in subsequent statements that modify `c:\oracle\oradata\dbfile1.dbf`, Oracle might conclude that this file is different than `dbfile1.dbf`.

Also, because of this behavior, SQL statements and Recovery Manager commands that refer to existing files must be specified exactly as they were originally entered or they are not recognized. An example of a SQL statement that refers to existing files is the `ALTER DATABASE RENAME FILE` statement.

In release 8.1.6 and higher, the flawed filename normalization mechanism is corrected. However, existing filenames in the control file and recovery catalog must be normalized with the new filename normalization mechanism.

Note: Do not perform the following procedure on Oracle releases prior to release 8.1.6.

To normalize these filenames, complete the following steps:

1. Using SQL*Plus, connect to the database as a user with `SYSDBA` privileges.
2. Shut down the database using `SHUTDOWN NORMAL` or `SHUTDOWN IMMEDIATE`:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

3. Make an operating system backup of your control file.

   See Also: *Oracle9i User-Managed Backup and Recovery Guide* for more information about operating system backups.

4. Run `STARTUP MOUNT` to mount the database without opening it:

   ```
   SQL> STARTUP MOUNT
   ```

5. Run the `DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAMES` procedure to normalize the filenames in your control file:

   ```
   SQL> EXECUTE DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAMES;
   ```

6. When the `DBMS_BACKUP_RESTORE.RENORMALIZEALLFILENAMES` procedure has completed successfully, open the database:

   ```
   SQL> ALTER DATABASE OPEN;
   ```

7. Exit SQL*Plus.
8. Log in to Recovery Manager and connect to a target database and recovery catalog.

   For example, if the network service name for the target database is `TGT_DB` and the network service name for the recovery catalog database is `CAT_DB`, then you can enter the following, substituting the appropriate schema names and passwords:

   ```
   rman target sys/password@tgt_db catalog rcat_schema/rcat_password@cat_db
   ```

9. Issue the `RENORMALIZE CATALOG` command to normalize the filenames in the recovery catalog for this target database:

   ```
   RMAN> renormalize catalog;
   ```

Note: The `RENORMALIZE CATALOG` command is not considered part of the Recovery Manager syntax and is not documented in the *Oracle9i Recovery Manager User's Guide*. The command is only intended for use on databases migrated or upgraded from a release prior to release 8.1.6 on Windows platforms.

10. Repeat Steps 8 and 9 for each release 8.1.6 or higher target database registered in this recovery catalog.

Your filenames are now normalized.

Note: If you need to restore a control file for a point-in-time recovery from a backup that was taken before you completed the filename normalization procedure described above, then first restore the backup control file, then perform Steps 1 to 7, and finally perform the recovery.

Tasks to Complete Only After Migrating Your Database

Complete the following tasks only if you migrated your database from Oracle7 or version 6. These tasks are *not* required if you upgraded your database from a version 8 release.

Rebuild Unusable Bitmap Indexes

During migration, some bitmap indexes may become unusable. To find these indexes, issue the following SQL statement:

```
SELECT index_name, index_type, table_owner, status
   FROM dba_indexes
      WHERE index_type = 'BITMAP'
      AND status = 'UNUSABLE';
```

Rebuild the unusable bitmap indexes listed.

See Also: *Oracle9i Database Performance Guide and Reference* and *Oracle9i Database Concepts* for more information about using bitmap indexes

Migrate Partition Views to Partition Tables

Partition views are not recommended for new applications in Oracle9*i*, and existing partition views should be converted to partitioned tables. You can convert partition views created for Oracle7 databases to partitioned tables by using the `EXCHANGE PARTITION` option of the `ALTER TABLE` statement.

> **See Also:** *Oracle9i Database Administrator's Guide* for information about converting partitioned views to partitioned tables and *Oracle9i Database Concepts* for background information about partition views and partitioned tables.

Migrate or Upgrade to the New Release of Oracle Net (Optional)

Migrating or upgrading to the new release of Oracle Net is not required. However, Oracle Net provides significant advantages over SQL*Net V2, including simplified configuration and expanded functionality. The new release of Oracle Net also provides the following advantages over past releases of Oracle Net and SQL*Net:

- **Service naming** enables clients to access a service as a whole, using the service name, rather than a specific database instance. Service naming logically separates the service name from any particular instance name and replaces the SID parameter, enabling one instance to serve multiple services. Individual instances also can be part of multiple services.
- **Instance registration** is automatic. Instances register themselves with the listener when they are started. In past releases, information about the instance was configured manually in the `listener.ora` file.
- **Failover** is automatic. If an instance is down, a client connect request is sent to a different listener automatically.
- **Load balancing** distributes connections over the available listeners.

> **See Also:** *Oracle Net Services Administrator's Guide* for more information about the advantages of Oracle Net, and see Appendix F, "Migration and Compatibility for Oracle Net Services" for detailed instructions on migrating or upgrading to the new release of Oracle Net.

Test the Database and Compare Results

Test the Oracle9*i* database using the testing plan you developed in "Develop a Testing Plan" on page 3-22. Compare the results of the test with the results obtained with the original database and make certain the same, or better, results are achieved.

Generally, the performance of the migrated Oracle9*i* database should be as good as, or better than, the performance of the source database. If you notice any decline in database performance with Oracle9*i*, then make sure the initialization parameters are set properly, because improperly set initialization parameters can hurt performance.

Tune the Migrated Database

If you want to improve the performance of the migrated database, then tune the database. Most of the actions normally used to tune Oracle7 databases and related applications either have the same effect on, or are unnecessary for, Oracle9*i* databases. Therefore, actions you used to tune your source database and applications should not impair the performance of the migrated Oracle9*i* database.

See Also: *Oracle9i Database Performance Guide and Reference* for tuning information

Task to Complete Only After Upgrading Your Database

Complete the following task only if you upgraded your database from a version 8 release. This task is *not* required if you migrated your database from Oracle7 or version 6.

Rebuild Unusable Function-Based Indexes

During an upgrade, some function-based indexes may become unusable. To find these indexes, issue the following SQL statement:

```
SELECT owner, index_name, funcidx_status
   FROM dba_indexes WHERE funcidx_status = 'DISABLED';
```

Rebuild the unusable function-based indexes listed.

9

Compatibility and Interoperability

This chapter describes compatibility and interoperability issues that may arise because of differences between Oracle releases. These differences may affect general database administration and existing applications.

This chapter covers the following topics:

- What Is Compatibility?
- Features Requiring 9.0.0 or Higher Compatibility Level
- Features Requiring 8.1.0 or Higher Compatibility Level
- What Is Interoperability?
- Compatibility and Interoperability Issues

> **See Also:** *Oracle9i Database New Features* for information about desupported features

What Is Compatibility?

When you upgrade to a new release of Oracle, certain new features may make your database incompatible with your previous release. Your upgraded Oracle database becomes incompatible with your previous release under the following conditions:

- A new feature stores any data on disk (including data dictionary changes) that cannot be processed with your previous release.
- An existing feature behaves differently in the new environment as compared to the old environment. This type of incompatibility is classified as a **language incompatibility**.

The COMPATIBLE Initialization Parameter

Oracle enables you to control the compatibility of your database with the `COMPATIBLE` initialization parameter. By default, when the `COMPATIBLE` initialization parameter is not set in your initialization parameter file, it defaults to the lowest possible setting for the release, which is 8.1.0 for release 9.0.1. You cannot use new features that would make your database incompatible until you reset the `COMPATIBLE` initialization parameter to a higher value.

This default behavior has the following advantages:

- Because compatibility with your previous release is maintained by default, it is easier to downgrade.
- If you are operating in an environment with more than one database, then your upgraded database remains compatible with databases that have not yet been upgraded.

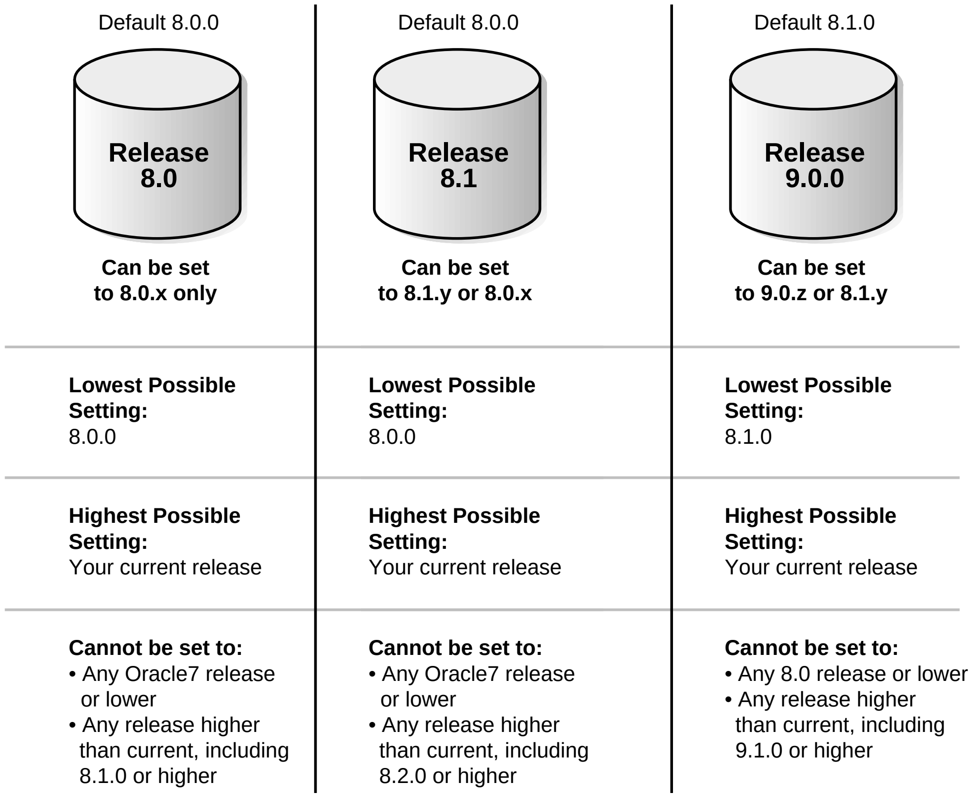
Of course, the major disadvantage of the default setting is that many of the features of the new release are not available to you if you leave the `COMPATIBLE` initialization parameter unset.

> **See Also:** "Features Requiring 9.0.0 or Higher Compatibility Level" on page 9-9 and "Features Requiring 8.1.0 or Higher Compatibility Level" on page 9-12 for a list of these features in the new release.

Depending on the products you chose to install during your release 9.0.1 installation of Oracle, the Oracle Universal Installer may set the `COMPATIBLE` initialization parameter to a higher value, such as 9.0.0. Check your initialization parameter file if you are unsure of the current setting of the `COMPATIBLE` initialization parameter.

Figure 9–1 illustrates the default settings and the possible settings for release 8.0, release 8.1, and release 9.0.1 of Oracle.

Figure 9–1 The COMPATIBLE Initialization parameter

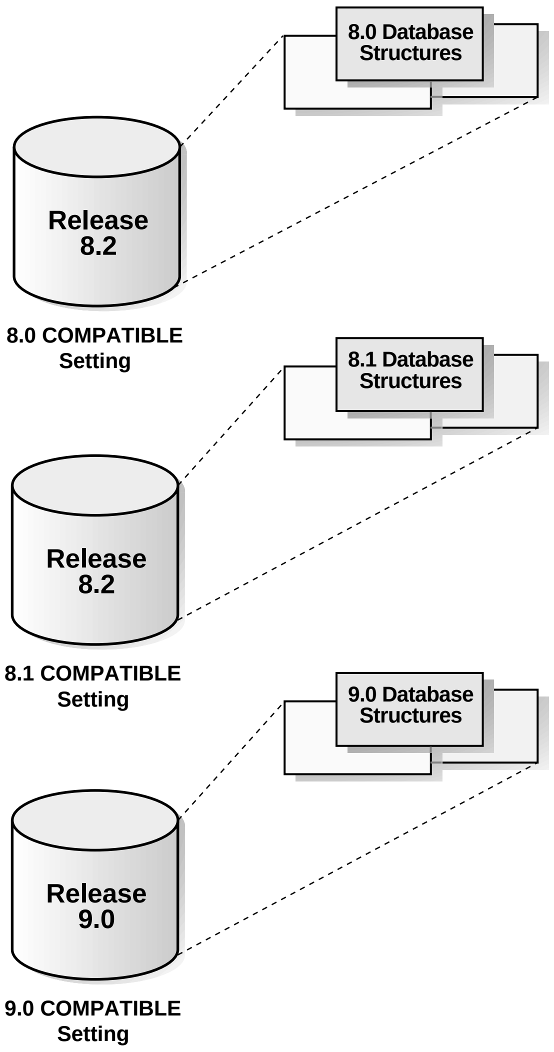

Default 8.0.0
Release 8.0
Can be set to 8.0.x only
Lowest Possible Setting: 8.0.0
Highest Possible Setting: Your current release
Cannot be set to:
• Any Oracle7 release or lower
• Any release higher than current, including 8.1.0 or higher
Default 8.0.0
Release 8.1
Can be set to 8.1.y or 8.0.x
Lowest Possible Setting: 8.0.0
Highest Possible Setting: Your current release
Cannot be set to:
• Any Oracle7 release or lower
• Any release higher than current, including 8.2.0 or higher
Default 8.1.0
Release 9.0.0
Can be set to 9.0.z or 8.1.y
Lowest Possible Setting: 8.1.0
Highest Possible Setting: Your current release
Cannot be set to:
• Any 8.0 release or lower
• Any release higher than current, including 9.1.0 or higher

How the COMPATIBLE Initialization Parameter Operates

The `COMPATIBLE` initialization parameter operates in the following way:

- It controls the behavior of your Oracle database. For example, if you run a release 9.0.1 database with the `COMPATIBLE` initialization parameter set to 8.1.0, then the release 9.0.1 database generates release 8.1 compatible database structures on disk. Therefore, the `COMPATIBLE` setting enables or disables the use of features. If you try to use any of the new features that make the database incompatible with the `COMPATIBLE` initialization parameter setting, then an error is returned. However, any new features that do not make incompatible changes on disk are enabled.
- It makes sure that the database is compatible with its setting. If the database becomes incompatible with its setting, then the database does not start and terminates with an error. If this happens, then you must set the `COMPATIBLE` initialization parameter in your initialization parameter file to an appropriate value for the database.

Figure 9–2 Database Structures Depend on the COMPATIBLE Setting



See Also: *Oracle9i Database Concepts* for more information about database structures

Downgrading and Compatibility

Once you upgrade or migrate to a new release, you can set the `COMPATIBLE` initialization parameter to match the new release. Doing so enables you to use all of the features of the new release, but may make it more difficult for you to downgrade to your previous release. If you want to downgrade, then you must remove all of the incompatibilities with the release to which you are downgrading, which is a process that may require a great deal of time and effort.

> **See Also:** Chapter 13, "Downgrading to Release 8.1" for more information about downgrading.

Compatibility Level

The compatibility level of your database corresponds to your `COMPATIBLE` initialization parameter setting. For example, if you set the `COMPATIBLE` initialization parameter to 8.1.6, then the database runs at 8.1.6 compatibility level.

Checking Your Current COMPATIBLE Initialization Parameter Setting

To check your current `COMPATIBLE` initialization parameter setting, issue the following SQL statement:

```
SQL> SELECT name, value, description FROM v$parameter
        WHERE name = 'compatible';
```

Checking the Compatibility Level of Specific Features

To check the compatibility level of specific features, issue the following SQL statement:

```
SQL> SELECT * FROM v$compatibility;
```

Features with a compatibility level of 0.0.0.0.0 are not currently in use.

When to Set the COMPATIBLE Initialization Parameter

You should set the `COMPATIBLE` initialization parameter at a specific point in your migration, upgrade, or downgrade process. Follow the procedure in the appropriate chapter and set the `COMPATIBLE` initialization parameter only when you are instructed to do so.

Note: After the migration, upgrade, or downgrade procedure is complete, you can set the `COMPATIBLE` initialization parameter whenever necessary.

Setting the COMPATIBLE Initialization Parameter

Complete the following steps to set the `COMPATIBLE` initialization parameter:

1. Perform a backup of your database before you set the `COMPATIBLE` initialization parameter (optional).

   Setting the `COMPATIBLE` initialization parameter may cause your database to become incompatible with earlier releases of Oracle, and a backup ensures that you can return to the earlier release if necessary.

   See Also: *Oracle9i User-Managed Backup and Recovery Guide* for more information about performing a backup.

2. If you are changing the `COMPATIBLE` initialization parameter to a lower setting, complete the following steps. If you are changing the `COMPATIBLE` initialization parameter to a higher setting, skip to Step 3:

   a. Make sure your database does not have any incompatibilities with the intended lower setting.

   If you plan to lower the `COMPATIBLE` initialization parameter to an 8.1.x setting, see Chapter 14, "Removing Incompatibilities Before Downgrading to Release 8.1" and follow the instructions for removing incompatibilities with 8.1 releases.

   b. Run `ALTER DATABASE RESET COMPATIBILITY`:

   ```
   ALTER DATABASE RESET COMPATIBILITY;
   ```

   See Also: "About ALTER DATABASE RESET COMPATIBILITY" on page 9-8 for more information.

3. Shutdown the database:

   ```
   SHUTDOWN IMMEDIATE
   ```

4. Edit the initialization parameter file to enter or change the `COMPATIBLE` setting.

   For example, to set the `COMPATIBLE` initialization parameter to 8.1.0, enter the following in the initialization parameter file:

   ```
   COMPATIBLE=8.1.0
   ```

5. Start the database using `STARTUP`.

About ALTER DATABASE RESET COMPATIBILITY

You use the `ALTER DATABASE RESET COMPATIBILITY` statement to instruct Oracle that you want to change the compatibility level to a lower release. Some Oracle features, such as external tables for example, require a compatibility level of 9.0.0 or higher. If you set the `COMPATIBLE` initialization parameter to 9.0.0 or higher and then create an external table, then the external table is a 9.0.0 compatible object in the database.

`ALTER DATABASE RESET COMPATIBILITY` checks for each feature that may have created an object that is incompatible with the lowest possible compatibility level, which is 8.1.0. If the check indicates that no incompatible objects exist for a certain feature, then the compatibility level of the feature is set to 0.0.0, which means that the feature is not in use. If, however, the check indicates that incompatible objects created by a certain feature exist, then the compatibility level for that feature is set to the required compatibility level.

For example, if one or more automatic segment-space managed tablespaces exist, then the compatibility level for the automatic segment-space managed tablespaces feature is set to 9.0.0, because 9.0.0 is the required compatibility level for that feature. It is important to understand, however, that `ALTER DATABASE RESET COMPATIBILITY` cannot raise the compatibility level of your database. You must first set the `COMPATIBLE` initialization parameter to a higher value, such as 9.0.0, before you can create database objects that require 9.0.0 compatibility level.

If you close the database, reset the `COMPATIBLE` initialization parameter to a lower setting, and then open the database, Oracle checks the compatibility level of each feature. If a feature has a compatibility level higher than the compatibility level specified by the `COMPATIBLE` initialization parameter in the initialization parameter file, then the database fails to open and displays an error message indicating the incompatible feature or features.

If you remove all of the incompatibilities that exist in your database, but fail to run the `ALTER DATABASE RESET COMPATIBILITY` statement before shutting down the database, then the database will still fail to open, even if no incompatibilities exist. The database will fail to open because it was not instructed to check the compatibility level of each feature against the objects that exist in the database. Because it did not reset the compatibility level for these features, Oracle simply remembers that incompatible objects were created at some time in the past. Running the `ALTER DATABASE RESET COMPATIBILITY` statement instructs Oracle to explicitly check for incompatible objects, and resets the compatibility level if no incompatible objects exist.

Features Requiring 9.0.0 or Higher Compatibility Level

To use the features listed in the following tables, the `COMPATIBLE` initialization parameter must be set to 9.0.0 or higher, unless stated otherwise.

The features listed *do not* represent a complete list of the new features introduced in release 9.0.1. Instead, the features listed are only those new release 9.0.1 features that require a 9.0.0 or higher compatibility level; some new features do not require this compatibility level.

See Also: *Oracle9i Database New Features* for more information about the features listed below and for information about other new release 9.0.1 features. You also can check the *Oracle9i Documentation Master Index* for entries relating to the new features listed below.

Tablespaces

Table 9–1 Tablespaces: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	**Features Requiring 9.0.0 or Higher Compatibility Level**
Tablespaces	Automatic Segment-Space Managed Tablespaces Default Temporary Tablespaces Automatic Undo Managed Tablespaces

Schema Objects

Table 9–2 Schema Objects: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
Tables	External Tables
Index-Organized Tables	Hash Partitioned Index-Organized Tables Index-Organized Tables with Mapping Tables LOBs in Range Partitioned Index-Organized Tables
Indexes	B-Tree Indexes on `UROWID` Datatypes for Heap and Index-Organized Tables Bitmap Indexes for Index-Organized Tables Domain Indexes on Index-Organized Tables Indexes With Large Keys
Hash Clusters	Single-Table Hash Clusters
Length Semantics	Character Semantics for the sizing of `CHAR` and `VARCHAR2` datatypes. See the parameter `NLS_LENGTH_SEMANTICS` for more information. Bitmap Indexes for Index-Organized Tables

Partitioning

Table 9–3 Partitioning: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
Partitioning of Tables	Hash Partitioning of Index-Organized Tables

Built-In Datatypes

Table 9–4 Built-In Datatypes: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
LOBs (large objects)	LONG API for LOBs Support in Range Partitioned Index-Organized Tables

User-Defined Datatypes

Table 9–5 User-Defined Datatypes: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
Aggregate Functions	User-Defined Aggregate Functions
Object Types	Type Evolution
Type Inheritance	Columns of non-final Types, or of types which have embedded non-final types Columns of subtypes, or of types which have embedded subtypes
Varrays	Support in Range Partitioned Index-Organized Tables

Data Protection

Table 9–6 Data Protection: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
Security	External Initialized Context

Distributed Databases

Table 9–7 Distributed Databases: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
Oracle Replication	Adding master sites to a master group without quiescing the master group Adding columns to a master table without quiescing its master group Replication of user-defined types and objects based on user-defined types in both multimaster and materialized view replication environments Row level dependency tracking for improved parallel propagation (specifying the `ROWDEPENDENCIES` clause in a `CREATE TABLE` statement) Fast refresh of materialized views with one to many or many to many subqueries Fast refresh of materialized views with a `UNION` operator Multi-tier materialized views

Data Access

Table 9–8 Data Access: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
SQL and PL/SQL	Nested Transactions Pipelined Table Functions Parallel Table Functions
Constraints	View Constraints

Data Warehousing

Table 9–9 Data Warehousing: Features Requiring 9.0.0 or Higher Compatibility Level

Functional Area	Features Requiring 9.0.0 or Higher Compatibility Level
Summary Management Using Materialized Views	Support in Index-Organized Tables

Features Requiring 8.1.0 or Higher Compatibility Level

To use the features listed in the following tables, the `COMPATIBLE` initialization parameter must be set to 8.1.0 or higher, unless stated otherwise.

The features listed *do not* represent a complete list of the features introduced in release 8.1. Instead, the features listed are only those release 8.1 features that require an 8.1.0 or higher compatibility level; some features do not require this compatibility level.

See Also: *Oracle9i Database New Features* for more information about the features listed below and for information about other release 8.1 features. You also can check the *Oracle9i Documentation Master Index* for entries relating to the features listed below.

Applications

Table 9–10 Applications: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Java	Enterprise JavaBeans Java code in stored procedures SQLJ Translator
Oracle Call Interface (OCI)	Support for Client Notification

Tablespaces

Table 9–11 Tablespaces: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Tablespaces	Locally Managed Tablespaces Online Read-Only Tablespaces Tablespace Migration (requires a `COMPATIBLE` setting of 8.1.6 or higher) Transportable Tablespaces

Schema Objects

Table 9–12 Schema Objects: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Tables	Drop Column Support Temporary Tables
Index-Organized Tables	Key Compression Support for Index-Organized Tables Secondary Indexes on Index-Organized Tables LOBs in Index-Organized Tables Partitioning of Index-Organized Tables
Indexes	Bitmap Index Protection (the ALTER TABLE statement no longer invalidates bitmap indexes) Extensible Indexing Function-Based Indexes Index Segment Coalesce Key Compression Support for Indexes Online Index (Re)build
Hash Clusters	Single-Table Hash Clusters

Partitioning

Table 9–13 Partitioning: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Partitioning of Tables	Partitioning of Index-Organized Tables Partitioning of Tables with LOBs Partitioning of Object Tables Partitioning of Tables with User-Defined Types, including Columns with the following Types: object, REF, VARRAY, and nested table Partitioning of Tables Using Composite Methods and Hash Methods
Partitioning of Hash Clusters	Support for Hash Partitioning

Built-In Datatypes

Table 9–14 Built-In Datatypes: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
LOBs (large objects)	Cache Reads Mode for LOBs (requires a `COMPATIBLE` setting of 8.1.6 or higher) Partitioning of Tables with LOBs Temporary LOBs Varying-Width Character Sets for CLOBs and NCLOBs
ROWIDs	UROWIDs (universal rowids) **Note:** An 8.1.0 compatibility level is required to use the new UROWID datatype as part of a database object, such as a table. However, UROWID variables can be used in PL/SQL code on a release 8.1 database with any 8.0 compatibility level.

User-Defined Datatypes

Table 9–15 User-Defined Datatypes: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Nested Tables	Collection Locators Nested Table Data in Index-Organized Tables Triggers on Nested Table View Columns User-Specified Storage Clauses for Nested Tables
Object Identifiers	User-Defined Object Identifiers
REFs	Referential Integrity Constraint on a REF Column
Varrays	Storage Parameters for Storing Varrays as LOBs Varray Data in Index-Organized Tables

Oracle Parallel Server

Table 9–16 Oracle Parallel Server: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Oracle Parallel Server	Instance Affinity for Jobs

Data Protection

Table 9–17 Data Protection: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Security	Application Context Fine-Grained Access Control N-Tier Authentication and Authorization
Database Backup and Recovery	Fast-Start On-Demand Rollback Proxy Copy Support for the Oracle Media Management API

Distributed Databases

Table 9–18 Distributed Databases: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Oracle Replication	Column Level Snapshot Subsetting for Updatable Snapshots
Heterogeneous Services	Agent Self-Registration

Data Access

Table 9–19 Data Access: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
SQL and PL/SQL	Autonomous Transactions Bulk Binds C Call Specifications CALL Statement Change FREELIST Specification in an ALTER statement (requires a `COMPATIBLE` setting of 8.1.6 or higher) Native Dynamic SQL NOCOPY Parameter Passing Mode The `utlchn1.sql` and `utlexpt1.sql` Scripts
Advanced Queuing	Addition of the Original Message ID Column for Propagated Messages Addition of a Sender's ID Column Database Event Publication Instance Affinity for Queue Tables Non-Persistent Queues Queue Level and System Level Privileges Rules Based Subscriptions Separate Storage of History Management Information **Note:** Propagation requires a compatibility level of 8.0.4 or higher.
Packages	DBMS_REPAIR Package
Constraints	DISABLE VALIDATE Constraint State
Triggers	Enhance DDL Support in Triggers (requires a `COMPATIBLE` setting of 8.1.6 or higher) Object OutBinds in Triggers Triggers on Nested Table View Columns Triggers on DATABASE and SCHEMA Triggers with a CALL to a Procedure as the Trigger Body
The Optimizer	Extensible Optimizer Optimizer Plan Stability
Database Resources	Database Resource Manager

Data Warehousing

Table 9–20 Data Warehousing: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Summary Management Using Materialized Views	Date Folding Dimensions Schema Object Query Rewrite Rewrite Privileges for Query Rewrite

Spatial and Visual Information

Table 9–21 Spatial and Visual Information: Features Requiring 8.1.0 or Higher Compatibility Level

Functional Area	Features Requiring 8.1.0 or Higher Compatibility Level
Spatial	Spatial Indextype Spatial Operators
Visual Information Retrieval (VIR)	VIR Indextype VIR Operators

Note: Spatial and VIR require that the `COMPATIBLE` initialization parameter be set to 8.1.0 or higher *before* you install them.

What Is Interoperability?

Interoperability is the ability of different releases of Oracle to communicate and work together in a distributed environment. An Oracle distributed database system can have Oracle databases of different releases, and all supported releases of Oracle can participate in a distributed database system. However, the applications that work with a distributed database must understand the functionality that is available at each node in the system.

For example, a distributed database application cannot expect an Oracle7 database to understand the object SQL extensions that are available only with release 8.0 and higher.

Note: Since this book documents migrating, upgrading, and downgrading between different releases of Oracle, this definition of interoperability is appropriate. However, other Oracle documentation may use a broader definition of the term **interoperability**; for example, in some cases, interoperability may describe communication between different hardware platforms and operating systems.

Compatibility and Interoperability Issues

The following sections describe compatibility and interoperability issues and the actions you can take to prevent problems resulting from these issues. The issues discussed in these sections occur because of differences between Oracle releases:

- Applications
- The STARTUP Command
- Tablespaces and Datafiles
- Data Dictionary
- Schema Objects
- Datatypes
- User-Defined Datatypes
- SQL and PL/SQL
- Advanced Queuing (AQ)
- Procedures and Packages
- Oracle Optimizer
- Oracle9i Real Application Clusters
- Database Security
- Database Backup and Recovery
- Distributed Databases

- SQL*Net or Oracle Net
- Export/Import
- Miscellaneous Compatibility and Interoperability Issues

Applications

You do not need to modify existing (Oracle7 and version 8) applications that do not use new release 9.0.1 features. Existing applications should achieve the same, or enhanced, functionality on release 9.0.1. To increase the likelihood that applications running against your release 9.0.1 database will continue to work if you downgrade to version 8, you can set the COMPATIBLE initialization parameter to 8.1.7 or lower.

However, the COMPATIBLE initialization parameter only restricts the use of release 9.0.1 features that change the formatting on disk, not the use of other release 9.0.1 features. Therefore, a setting of 8.1.7 or lower does not guarantee that applications developed in release 9.0.1 will run correctly if the database is downgraded to version 8.

> **See Also:** Chapter 10, "Upgrading Your Applications" for more information about upgrading applications.

General Compatibility and Interoperability Issues for Applications

This section describes general compatibility and interoperability issues for applications.

Change in Maximum VARCHAR2, CHAR, And RAW Size Oracle7 clients using VARCHAR2, CHAR, or RAW datatypes may run into buffer overflow errors in their applications. This may happen because in release 8.0 and higher, the maximum size of the VARCHAR2 datatype was increased from 2000 to 4000 and the maximum size of CHAR and RAW datatypes was increased from 255 to 2000.

Clients encountering this problem can either modify their applications to accept a larger buffer size or use the SUBSTR() operator in the offending query to limit the return size of the buffer to a length that can be processed by the application.

In the following example, column SIZE_TAB.SIZE_COL is VARCHAR(80).

```
SQL> CREATE VIEW v1 AS SELECT
       LPAD(' ',40-length(size_tab.size_col)/2,' ') size_col
       FROM size_tab;
Statement processed.
```

```
SQL> DESC v1

Column Name                    Null?    Type
------------------------------ -------- ----
SIZE_COL                                VARCHAR2(4000)

SQL> DROP VIEW v1;
View dropped.

SQL> CREATE VIEW v1 AS SELECT
       SUBSTR(lpad(' ',40-length(size_tab.size_col)/2,' '), 2000) size_col
       FROM size_tab;

SQL> DESC v1;

Column Name                    Null?    Type
------------------------------ -------- ----
SIZE_COL                                VARCHAR2(2001)
```

Index-Organized Tables Accessed by Applications If a table accessed by an application changes from a regular table to an index-organized table, then the application may require changes. The possible changes depend on whether the application uses physical rowids or universal rowids (UROWIDs).

Whether an application requires changes depends on the kind of host variables the application is using to bind or define rowid values:

- If the application uses release 8.0 or higher OCI rowid descriptors (OCIROWID * for Pro*C and SQL-ROWID for Pro*COBOL), then the application should continue to function properly without any changes.
- If the application always performs `DESCRIBE` on the host variables, then the application should continue to function properly without any changes. Make sure the application can accommodate the new SQLT_RDD datatype.
- If the application uses SQLT_RID host variables, then you must rewrite the application to use VARCHAR host variables or rowid descriptors. Rowid descriptors are preferred.
- If the application uses CHARACTER host variables, then the behavior also depends on the size of the host variables. If the size can accommodate the primary key and if the variable is a variable length string, then the application should continue to function properly without any changes. However, if the

application uses a fixed size 18 character string, then you must change the application to use longer variable strings or OCI descriptors.

For applications using UROWIDs, VARCHAR host variables may no longer be large enough to hold the rowids. If so, then change the application to increase the variable maximum size or change the application to use OCI rowid descriptors. OCI rowid descriptors are preferred because they are opaque and resize automatically.

Change in Behavior for ANALYZE TABLE VALIDATE STRUCTURE Statement Starting with release 8.1, the `ANALYZE TABLE VALIDATE STRUCTURE` statement no longer stops running at the first error. Modify any applications that depend on this behavior to account for this change.

OCI Applications

This section describes compatibility and interoperability issues relating to OCI applications.

> **See Also:** *Oracle Call Interface Programmer's Guide* for more information.

Shared Structures and Interoperability Shared structures are not supported on Oracle7 clients linked with release 8.1 libraries. To take advantage of shared structures, applications must be written with the release 8.1 or higher OCI and must be communicating with a release 8.1 or higher Oracle database server.

A release 8.1 OCI client accessing a release 8.0 Oracle database server only partially realizes the benefits of shared structures, and shared structures are not supported if both the client and the Oracle database server are release 8.0 or lower.

Thread Safety The ORLON and OLON calls are not supported in version 8. However, you still should use OLOG, even for single-threaded applications.

> **Note:** The OLOG call is required for multithreaded applications.

OCI Application Link Line For OCI applications, the Oracle9*i* link line differs from the Oracle7 link line. See the *`ORACLE_HOME`*`/rdbms/demo/demo_rdbms.mk` file for examples of using the Oracle9*i* link line as an Oracle9*i* OCI application is compiled.

Oracle7 Clients Oracle7 clients can make selective use of Oracle9*i* OCI, combining Oracle7 and Oracle9*i* calls. The degree of functionality added depends on which calls are used. The encryption API and password reset calls are independently usable as well. Use Oracle9*i* OCI for all phases of the statements being processed to enable the following functionality:

- failover
- prefetch
- piggybacked commit or cancel
- client-side conversions

Oracle7 clients must log in using Oracle9*i* calls if they want to combine Oracle7 code with Oracle9*i* code.

Using Batch Error Mode for Statement Execution Starting with release 8.1, OCI applications can use the batch error mode when executing array DMLs using *OCIStmtExecute*. To do this, both the OCI and server libraries must be release 8.1 or higher.

You can modify existing applications to use batch error mode by setting the mode parameter to OCI_BATCH_ERRORS and adding new code required for this functionality. Then, recompile and relink the application with the release 8.1 client libraries.

Support for Client Notification Starting with release 8.1, client notification is supported in OCI applications using the publish/subscribe interface. Client notification enables applications to take advantage of Database Event Publication and Advanced Queuing features. To use the client notification feature, client applications must link with release 8.1 or higher client libraries.

Support for the LISTEN Call with the Advanced Queuing Option Starting with release 8.1, the LISTEN call is supported in OCI applications. The LISTEN call is available with the Advanced Queuing Option and can be used to monitor a set of queues for a message. To use the LISTEN call, client applications must link with release 8.1 or higher client libraries.

Precompiler Applications

This section describes compatibility and interoperability issues relating to precompiler applications.

> **See Also:** *Pro*C/C++ Precompiler Programmer's Guide* and *Pro*COBOL Precompiler Programmer's Guide* for more information.

Connecting With SYSDBA Privileges in Pro*C/C++ SYSDBA privileges are no longer available by default when you issue the CONNECT statement in Pro*C/C++. In release 8.0, the following CONNECT statement connected with SYSDBA privileges in Pro*C/C++:

```
EXEC SQL CONNECT :sys IDENTIFIED BY :sys_passwd;
```

In release 8.1 and higher, issue the following CONNECT statement to connect with SYSDBA privileges in Pro*C/C++:

```
EXEC SQL CONNECT :sys IDENTIFIED BY :sys_passwd IN SYSDBA MODE;
```

Connecting With SYSDBA Privileges in Pro*COBOL SYSDBA privileges are no longer available by default when you issue the CONNECT statement in Pro*COBOL. In release 8.0, the following CONNECT statement connected with SYSDBA privileges:

```
EXEC SQL
    CONNECT :sys IDENTIFIED BY :SYS-PASSWD
END-EXEC.
```

In release 8.1 and higher, issue the following CONNECT statement to connect with SYSDBA privileges:

```
EXEC SQL
    CONNECT :sys IDENTIFIED BY :SYS-PASSWD IN SYSDBA MODE
END-EXEC.
```

Ada Support in Version 8 The Pro*ADA product was officially desupported by Oracle in release 7.3. You can upgrade Pro*ADA to the latest release of SQL*Module for Ada 8.1, which has a number of new features. However, SQL*Module for ADA 8.1 does not provide object support.

PL/SQL Backward Compatibility and Precompilers PLSQL_V2_COMPATIBILITY backward compatibility behavior is available in the precompiler environment by setting the DBMS precompiler command line option as follows:

```
... DBMS=Oracle7
```

PL/SQL Applications

This section includes compatibility and interoperability issues for PL/SQL applications.

See Also: *PL/SQL User's Guide and Reference* for more information

Integrated SQL Analysis Syntax and semantic analysis of SQL statements in PL/SQL programs is now integrated with the SQL engine. As a result, any new SQL feature that is available through SQL*Plus or OCI is also available in PL/SQL.

In Oracle9*i*, syntax and semantic analysis of SQL statements is also a little stricter than in previous releases. PL/SQL catches additional errors in SQL statements during compilation itself, rather than throwing a runtime exception for invalid SQL syntax. As a result, you may see compile-time errors with the PL/SQL:ORA- prefix in PL/SQL programs that had compiled successfully in previous releases. The new error messages point to problems in the SQL statement that must be fixed before the program can be compiled successfully.

If you are unable to immediately modify a SQL statement to satisfy the new stricter checks, Oracle provides an event to temporarily assist you in migrating PL/SQL code to Oracle9*i*:

```
alter session set events='10933 trace name context forever, level 512';
```

Please note that this event is provided only for temporary migration assistance. Oracle Corporation strongly discourages long-term use of this event, and this event will be desupported in the next major release of Oracle.

Compatibility and Object Types In Oracle9*i*, object types that are qualified as `NOT FINAL`, `NOT INSTANTIABLE`, a subtype, or a SQLJ type cannot be referred to from PL/SQL programs in earlier releases of Oracle. Specifically, such programs will fail to compile with an error.

PL/SQL V2 Compatibility Mode The PL/SQL V2 compatibility mode is available in PL/SQL release 8.0 and higher. This mode is enabled by the `PLSQL_V2_COMPATIBILITY` initialization parameter.

You can set PL/SQL V2 compatibility mode in any one of the following three ways:

- Add the following line to your initialization parameter file:

  ```
  PLSQL_V2_COMPATIBILITY = true
  ```

- Issue the following SQL statement:

  ```
  ALTER SYSTEM SET PLSQL_V2_COMPATIBILITY = true;
  ```

- Issue the following SQL statement:

  ```
  ALTER SESSION SET PLSQL_V2_COMPATIBILITY = true;
  ```

The `PLSQL_V2_COMPATIBILITY` initialization parameter provides compatibility between PL/SQL release 8.0 and higher and PL/SQL V2 in the following situations:

- The PL/SQL V2 compiler allows a record type or index table type to be referenced before its definition in the source. PL/SQL release 8.0 and higher strictly requires that the type definition precede reference to the type in the source. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 regarding type definitions.
- The PL/SQL V2 compiler allows the following illegal syntax:

  ```
  return variable-expression
  ```

  This syntax is incorrect and should be changed to the following:

  ```
  return variable-type
  ```

  The PL/SQL release 8.0 and higher compiler issues an error when it encounters the illegal syntax. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and does not issue an error.
- In PL/SQL V2 it is possible to modify or delete elements of an index table passed in as an IN parameter, as in the following example:

  ```
  function foo (x IN table_t) is
  begin
  x.delete(2);
  end;
  ```

  This use of an IN parameter is incorrect. PL/SQL release 8.0 and higher correctly enforces the read-only semantics of IN parameters and does not let index table methods modify index tables passed in as IN parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and allows the parameter.
- PL/SQL V2 allows the passing (as an OUT parameter) of fields of IN parameters that are records, but PL/SQL release 8.0 and higher does not allow this type of passing. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 and allows this type of passing.
- The PL/SQL V2 compiler permits fields of OUT parameters that are record variables to be used in expression contexts (for example, in a dot-qualified name on the right-hand side of an assignment statement).

This use of OUT parameters should not be permitted. PL/SQL release 8.0 and higher does not permit OUT parameters to be used in expression contexts. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 in this regard.

- PL/SQL V2 allows OUT parameters in the `FROM` clause of a `SELECT` list. PL/SQL release 8.0 and higher does not allow this use of OUT parameters. However, when you enable PL/SQL V2 compatibility mode, PL/SQL release 8.0 and higher behaves the same as PL/SQL V2 in this regard.

Keyword Behavior Differences: Oracle7 vs. Release 8.0 and Higher The following keywords or types included in Oracle7 and release 8.0 and higher produce slightly different error message identifiers when used as a function name in a `SELECT` list:

Table 9–22 Keyword Behavior Differences

Keywords	Release 8.0 and Higher Behavior	Oracle7 Behavior
`CHARACTER`, `COMMIT`, `DEC`, `FALSE`, `INT`, `NUMERIC`, `REAL`, `SAVEPOINT`, `TRUE`	Generates errors: ORA-06550 and PLS-00222	Generates errors: ORA-06552 and PLS-00222

The STARTUP Command

This section describes compatibility and interoperability issues related to the `STARTUP` command.

Change in Default Parameter File Selection

When the `STARTUP` command is issued without the `PFILE` option, Oracle attempts to start up the instance using a default parameter file. In Oracle9*i*, the search criteria for selecting the default parameter file has changed to facilitate the use of a server parameter file.

In previous releases of Oracle, the `STARTUP` command looked for an initialization parameter file with the name *`ORACLE_HOME/`*`dbs/init`*`SID`*`.ora`, where *`SID`* is the instance name.

In Oracle9*i*, the process of selecting a default parameter file is as follows:

- The `STARTUP` command first looks for a server parameter file with the name *`ORACLE_HOME/`*`dbs/spfile`*`SID`*`.ora`, where *`SID`* is the instance name.
- The `STARTUP` command next looks for a server parameter file with the name *`ORACLE_HOME/`*`dbs/spfile.ora`.

- If the STARTUP command cannot find a server parameter file, it defaults to the behavior of the STARTUP command in previous releases, and looks for an initialization parameter file with the name *ORACLE_HOME*/dbs/init*SID*.ora.

  See Also: *Oracle9i Database Administrator's Guide* for more information about server parameter files

Tablespaces and Datafiles

This section describes compatibility and interoperability issues related to tablespaces and datafiles.

Automatic Undo Managed Tablespaces

Oracle instances can run in one of two undo space management modes:

- Automatic undo management mode
- Manual undo management mode

All instances of the same database must run in the same undo space management mode.

The COMPATIBLE initialization parameter effects how undo space is managed. Automatic undo management mode is allowed only if COMPATIBLE is set to 9.0.0 or higher. The instance is started in manual undo management mode if the UNDO_MANAGEMENT initialization parameter is not specified or if COMPATIBLE is set below 9.0.0.

In the manual undo management mode with COMPATIBLE set to 9.0.0 or higher, CREATE, ALTER, and DROP operations on undo tablespaces are allowed. Rollback segments can coexist with undo tablespaces. That is, rollback segments can exist while running in automatic undo management mode and undo tablespaces can exist while running in manual undo management mode. Undo tablespaces cannot be brought online unless the instance is running in automatic undo management mode.

In automatic undo management mode, DROP ROLLBACK SEGMENT operations are allowed. Rollback segments cannot be brought online.

See Also: *Oracle9i Database Administrator's Guide* for more information about managing undo space.

Transportable Tablespace

There are compatibility issues when you transport a tablespace between two databases.

> **See Also:** *Oracle9i Database Administrator's Guide* for information about these compatibility issues.

Tempfiles

Release 8.1 introduced tempfiles. The information about tempfiles is in different static data dictionary views and dynamic performance views than the information about datafiles. To view information about tempfiles, consult the `DBA_TEMP_FILES` static data dictionary view and the following dynamic performance views:

- `V$TEMPFILE`
- `V$TEMP_EXTENT_MAP`
- `V$TEMP_EXTENT_POOL`
- `V$TEMP_SPACE_HEADER`
- `V$TEMPSTAT`
- `V$TEMP_PING`

Oracle automatically assigns numbers to both datafiles and tempfiles. Two datafiles cannot share the same number; similarly, two tempfiles cannot share the same number. However, a tempfile and a datafile can share the same number.

> **See Also:** *Oracle9i SQL Reference* for information about tempfiles

Active Transactions Restriction for Read-Only Tablespaces

In releases prior to release 8.1, there could not be any active transactions in your database before you made a tablespace read-only. In release 8.1 and higher, this restriction is lifted if the `COMPATIBLE` initialization parameter is set to 8.1.0 or higher. With the database at 8.1.0 or higher compatibility level, the `ALTER TABLESPACE ... READ ONLY` statement waits for active transactions to complete, and then makes the tablespace read-only. If, however, the `COMPATIBLE` initialization parameter is set below 8.1.0, then the restriction still applies.

Data Dictionary

This section describes possible compatibility and interoperability issues resulting from data dictionary changes.

See Also: Appendix C, "Changes to Static Data Dictionary Views" and Appendix D, "Changes to Dynamic Performance Views" for more information, including lists of obsolete views.

Data Dictionary Protection

The data dictionary protection mechanism introduced in release 8.0 may cause problems in any applications that create user tables in the SYS schema and access them using the '*ANY*' privileges. For example, the user must have `DELETE CATALOG ROLE` to use the `DELETE` statement to purge the audit records in the `AUD$` table.

Creating and accessing user tables in SYS schema is not secure. Therefore, applications are expected to move the objects to a different schema. Use the `O7_DICTIONARY_ACCESSIBILITY` initialization parameter for temporary compatibility. However, this parameter is only for interim use.

Applications should not attempt to connect to user SYS without SYSDBA privileges. Instead of connecting to user SYS and sharing the password, grant DBA privilege to a normal user, who will connect to the database as a user with SYSDBA privileges to connect to SYS schema.

In Oracle9*i*, a user can be granted the `SELECT ANY DICTIONARY` privilege. A user with this privilege can access objects in the SYS schema regardless of the setting of `O7_DICTIONARY_ACCESSIBILITY`.

Obsolete Data Dictionary Views

Certain data dictionary views maintained in Oracle7 for backward compatibility to version 5 and version 6 of Oracle, created in the files `catalog5.sql` and `catalog6.sql`, are obsolete in release 8.0 and higher. Remove all references to these data dictionary views from your database tools and applications.

Schema Objects

This section describes compatibility and interoperability issues relating to schema objects.

Bitmap Index Protection

In releases prior to release 8.1, it was possible to unintentionally invalidate bitmap indexes by issuing certain SQL statements. The most common causes of bitmap index invalidation were the following types of statements:

- ALTER TABLE statements that define a primary key on an existing table that did not previously have a primary key.
- ALTER TABLE statements that define a NOT NULL constraint on a column that did not previously have this constraint.

Oracle9*i* eliminates these unintentional invalidations.

Datatypes

This section describes compatibility and interoperability issues relating to datatypes.

Datetime and Interval Datatypes

When a database is migrated or upgraded to Oracle9*i*, the database time zone is set to the time zone of the environment variable ORA_SDTZ. If ORA_SDTZ is not set, the database time zone is set to the time zone of the operating system clock. If the time zone of the operating system clock is not set or is not valid, the database time zone defaults to UTC.

old Oracle DATE data with time portion can be migrated to either TIMESTAMP to support fractional seconds or TIMESTAMP WITH LOCAL TIME ZONE to support time zone adjustments in addition to fractional seconds without having legacy data rewritten. An ALTER TABLE statement must be explicitly issued to modify a DATE column to a TIMESTAMP column or a TIMESTAMP WITH LOCAL TIME ZONE column.

Large Objects (LOBs)

This section describes compatibility and interoperability issues relating to LOBs.

Varying-Width Character Sets for CLOBs and NCLOBs Release 8.0 did not allow users other than SYSTEM to create tables with the CLOB or NCLOB datatype if the database character set was varying-width. Release 8.1 and higher supports CLOB and NCLOB datatypes in tables with a varying-width character set, and the data is stored as UCS2 (2-byte fixed-width unicode).

LOB Index Clause If you used the LOB index clause to store LOB index data in a tablespace separate from the tablespace used to store the LOB, then the index data will be relocated to reside in the same tablespace as the LOB if you perform either of the following actions in release 8.1 and higher:

- Perform an Export/Import on the LOB

- Exchange the LOB into a partitioned table

If you used Export/Import to migrate from Oracle7 to Oracle9*i*, then the index data was relocated automatically during migration. However, the index data was not relocated if you used the Migration utility or the Oracle Data Migration Assistant.

Also, if you create a new table in release 8.1 and higher and specify a tablespace for the LOB index for a non-partitioned table, then the tablespace specification will be ignored and the LOB index will be located in the same tablespace as the LOB.

To check the storage of LOB indexes, issue the following SQL statement connected as a user with SYSDBA privileges:

```
SELECT index_name, index_type, tablespace_name
    FROM dba_indexes
    WHERE index_type = 'LOB';
```

Date Columns in Dynamic Performance Views

In Oracle7, all date columns in dynamic performance views were VARCHAR2(20) strings in MM/DD/YY HH24:MI:SS format. In release 8.0 and higher, every date column is a real DATE column that uses the DATE datatype. In contrast to the previous VARCHAR2(20) string, the DATE datatype provides the following benefits:

- Establishes consistency, because all date columns are in the DATE datatype.
- Makes it easier to perform date arithmetic (including sorting) in SQL and PL/SQL.
- Enables you to set your date format using NLS_DATE_FORMAT.
- Allows you to see dates in the old format by setting NLS_DATE_FORMAT to MM/DD/YY HH24:MI:SS.
- Avoids two-digit year numbers, thereby avoiding problems at the year 2000 and beyond.

Note: Although Oracle7 displays dates using the VARCHAR(20) datatype in dynamic performance views, Oracle7 is still fully year-2000 compliant. Oracle7 stores time to the nearest second in the redo log files and control files.

Oracle ROWIDs

This section describes compatibility and interoperability issues related to rowids.

UROWID Datatype Release 8.1 introduced the UROWID (universal rowid) datatype. Clients prior to release 8.1 can access columns of UROWID datatype using character host variables only; other types of variables are not supported.

New Physical ROWID Datatype Format Release 8.0 introduced a new format for physical rowids. If you use physical rowids stored in columns or in application code, then the old physical rowids are invalid and must be converted.

> **See Also:** Chapter 12, "Migration Issues for Physical Rowids" for more information about the new physical rowid format

NCHAR and NLS Use

In version 8, you can declare the use of the national character set (NCHAR) for specific columns, attributes, PL/SQL variables, parameters, and return results. Unless such an explicit declaration is made, use of NCHAR and NLS is, for the most part, invisible and has no affect on other version 8 features. An exception is that SELECT statements on either the PROPS$ or the VALUE$ dictionary view may return the CHARACTER_SET_NAME column or the NLS_NCHAR_CHARACTERSET row.

Migration Issues with NCHAR and NLS The PROPS$ dictionary table contains two rows that describe the character sets specified in the CREATE DATABASE statement. The row holding NAME='NLS_CHARACTERSET' has the database character set's name in the VALUE$ column. The row holding NAME='NLS_NCHAR_CHARACTERSET' has the national character set's name in the VALUE$ column.

Compared to release 7.3, various views contain the new column, CHARACTER_SET_NAME, whose value is:

```
DECODE(x$.CHARSETFORM,
       1, 'CHAR_CS',
       2, 'NCHAR_CS',
```

where x$ represents one of the base tables. The DATA_TYPE or COLTYPE column value of the view will not change to indicate the character set choice.

NCHAR and NLS Environment Variables and Compatibility You should set NLS_LANG to your environment as follows:

- Set the ORA_NLS32 environment variable for the release 7.3.x environment
- Set the ORA_NLS33 environment variable for the release 8.0 and higher environment

Verify that the client has the correct NLS character set environment variables. An error is generated when release 7.3 NLS code tries to load a release 8.0 and higher character set.

User-Defined Datatypes

This section describes compatibility and interoperability issues relating to user-defined datatypes.

Type Evolution

Because type evolution requires the `COMPATIBLE` initialization parameter to be set to 9.0.0 or higher, clients which are using a previous release of PL/SQL cannot access evolved types.

Subtypes and Non-Final Types

Types created in release 8.1 and earlier are considered to be `FINAL` types. Thus, they cannot be used as supertypes in Oracle9*i*. However, an `ALTER` statement can be explicitly used to change the type to be `NOT FINAL`.

If the `COMPATIBLE` initialization parameter is set below 9.0.0, subtypes cannot be created. Further, not instantiable and non-final types cannot be created. Consequently, subtables, subviews, and substitutable columns are also not permitted.

Release 8.1 Clients Accessing a Release 9.0 Server Any transfer involving data of non-final types will return an error. Release 8.1 clients cannot access the release 9.0.1 server if the type has been altered to non-final on the server.

Release 9.0 Clients Accessing a Release 8.1 Server Since the release 8.1 server can have only non-final types, no errors occur.

New Format for User-Defined Datatypes

Release 8.1 introduced a new format for user-defined datatypes. The new format can result in significant performance improvements over the format used in release 8.0. To use the new user-defined datatypes format, the `COMPATIBLE` initialization parameter must be set to 8.1.0 or higher.

You can use release 8.0 user-defined datatypes in a release 8.1 or higher database without causing compatibility problems. However, the database will not realize the performance gains possible with the new format.

Release 8.1 and Higher Clients Accessing Release 8.0 User-Defined Datatypes The user-defined datatypes format is negotiated as part of the compatibility exchange between the client and server. If you are using a release 8.0 database server, then release 8.1 and higher clients can access the database, but they are set to release 8.0.

Release 8.0 Clients Accessing Release 8.1 and Higher User-Defined Datatypes When a release 8.0 client accesses a server with release 8.1 and higher-compatible user-defined datatypes, the database converts the user-defined datatypes to release 8.0 format. Consequently, the release 8.0 client can access the data, but performance gains may not be realized.

Nested Tables

Release 8.0 clients do not support the following release 8.1 and higher nested table features:

- Collection locators
- User-specified storage for collection columns, including storage of nested table data in an index-organized table

Therefore, access fails with an incompatibility error when a release 8.0 client attempts to access a release 8.1 or higher server and a nested table is specified to be returned as a locator, or the storage for the nested table is user-specified.

Varrays Stored as LOBs

Release 8.0 clients do not support specifications of storage parameters for storing varrays as LOBs. Therefore, access fails with an incompatibility error when a release 8.0 client attempts to access a release 8.1 or higher server where there is a specification of storage parameters for storing a varray as a LOB.

SQL and PL/SQL

This section describes compatibility and interoperability issues relating to SQL and PL/SQL.

See Also: *Oracle9i SQL Reference* and *PL/SQL User's Guide and Reference* for more information about SQL and PL/SQL

Functions GREATEST_LB, LEAST_UB, and TO_LABEL Desupported

Starting with release 8.1, the built-in PL/SQL functions `GREATEST_LB`, `LEAST_UB`, and `TO_LABEL`, which operate on Trusted Oracle labels, are no longer supported.

Native Dynamic SQL in PL/SQL

The following sections describe interoperability issues related to native dynamic SQL in PL/SQL:

Server-Side PL/SQL An Oracle database server at release 8.1.0 or higher compatibility level can execute native dynamic SQL statements that contain references to objects on a remote server at any compatibility level.

For example, the following procedure contains a native dynamic SQL statement and links to a remote Oracle database server:

```
PROCEDURE dyn1 is
BEGIN
    EXECUTE IMMEDIATE 'insert into tab@remote_link
        values ('a', 10)';
END;
```

In the example, *remote_link* can be a link to any version of Oracle, such as release 7.3, 8.0, or 8.1.

Native Dynamic SQL and RPC Calls PL/SQL programs that are targets of RPC calls can use native dynamic SQL, regardless of the release of the clients making the RPC calls. For example, release 7.3 or 8.0 clients can issue RPC calls to an Oracle database server at 8.1.0 or higher compatibility level.

SQL Scripts utlchain.sql and utlchn1.sql

The release 9.0.1 installation includes the following two scripts for creating a table that stores migrated and chained rows: `utlchain.sql` and `utlchn1.sql`. The `utlchn1.sql` script can be run on index-organized tables as well as regular tables, while the `utlchain.sql` script can be run only on regular tables, but not on index-organized tables.

In Oracle9*i*, you must always run the `utlchn1.sql` script.

SQL Scripts utlexcpt.sql and utlexpt1.sql

The release 9.0.1 installation includes the following two scripts for creating a table that stores exceptions from enabling constraints: `utlexcpt.sql` and `utlexpt1.sql`. The `utlexpt1.sql` script can be run on index-organized tables as well as regular tables, while the `utlexcpt.sql` script can be run only on regular tables, but not on index-organized tables.

In Oracle9*i*, you must always run the `utlexpt1.sql` script.

Behavior Change in Parallel CREATE TABLE Statements with the AS Subquery

In release 8.0 and higher, if you use the PARALLEL clause in a CREATE TABLE statement with the AS subquery, then Oracle ignores the INITIAL storage parameter and instead uses the NEXT storage parameter. Oracle7 did not ignore the INITIAL storage parameter.

For example, consider the following SQL statement:

```
CREATE TABLE tb_2 STORAGE (INITIAL 1M NEXT 500K)
   PARALLEL (DEGREE 2)
   AS SELECT * FROM tb_1;
```

In release 8.0 and higher, the value of INITIAL is 500 KB, while in Oracle7, the value of INITIAL is 1 MB.

Advanced Queuing (AQ)

This section includes compatibility and interoperability issues for AQ.

See Also: *Oracle9i Application Developer's Guide - Advanced Queuing* for more information about AQ. The sections below only provide compatibility and interoperability information about new AQ features, while *Oracle9i Application Developer's Guide - Advanced Queuing* provides detailed information about using them.

Queue Level and System Level Privileges

To use queue level and system level privileges, the queue table must be at 8.1.0 compatibility level or higher. Specifically, to grant queue level privileges using the following procedures in the DBMS_AQADM package requires an 8.1.0 or higher queue table compatibility level:

- GRANT_QUEUE_PRIVILEGE
- REVOKE_QUEUE_PRIVILEGE

Interoperability and the Sender's ID Column

In release 8.1 and higher, the sender's ID is mapped as an additional attribute in the message properties. This new attribute is ignored when there is communication between release 8.0 and release 9.0.1 and higher databases.

For OCI applications, the sender's ID attribute is available as a new attribute in the message properties descriptor. Release 8.1 and higher OCI clients use a new RPC code to send and receive the message properties to and from the server.

Rule Based Subscriptions

When you migrate a queue table from release 8.0 to release 8.1 or higher using the `DBMS_AQADM.MIGRATE_QUEUE_TABLE` procedure, any existing subscribers are upgraded automatically to subscribers with null rules.

Message Streaming

Message streaming is supported only if the source and destination databases both are release 8.1 or higher. A `COMPATIBLE` initialization parameter setting of 8.1.0 is not required for message streaming; it is supported even if `COMPATIBLE` is set to 8.0.5 or lower on a release 8.1 and higher database.

Procedures and Packages

This section describes compatibility and interoperability issues related to procedures and packages.

> **See Also:** *Oracle9i Supplied PL/SQL Packages and Types Reference* for more information about packages

The DBMS_LOB Package and NOCOPY

If the `COMPATIBLE` initialization parameter is set to 8.1.0 or higher, then the `DBMS_LOB` package uses the new `NOCOPY` syntax for the LOB parameters, and LOB locators that are passed to the `DBMS_LOB` package follow the new `NOCOPY` semantics.

If the `COMPATIBLE` initialization parameter is set below 8.1.0, then the `NOCOPY` syntax is not supported. Therefore, if you are at an 8.0.x compatibility level, then you should not:

- Use the new `NOCOPY` syntax in the `DBMS_LOB` package
- Run the new `dbmslob.sql` script

The DBMS_REPAIR Package

The `COMPATIBLE` initialization parameter must be set to 8.1.0 or higher to use the `DBMS_REPAIR` package. The `DBMS_REPAIR` package will fail if the compatibility level is below 8.1.0.

Syntax Change for the SET_SESSION_LONGOPS Procedure

Release 8.0 introduced changes to the `DBMS_APPLICATION_INFO.SET_SESSION_LONGOPS` procedure. For information about the new syntax, refer to the `dbmsapin.sql` file. If any of your applications use this procedure, then change the applications accordingly.

Oracle Optimizer

Setting the `COMPATIBLE` initialization parameter to 8.1.0 or higher will enable the optimizer to improve its choice of execution plan. An 8.1.0 compatibility level enables the optimizer to use a new column, `AVGCLN`, in the `HIST_HEAD$` data dictionary table to determine its choice of execution plan.

Oracle9*i* Real Application Clusters

Support for different releases of Oracle within one Oracle9*i* Real Application Clusters environment is operating system-specific. See your operating system-specific Oracle documentation for information about whether or not the coexistence of different releases within one Oracle9*i* Real Application Clusters environment is supported on your operating system.

Note: Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

In release 8.0 and higher, all Oracle instances that belong to a database and are linked in Parallel Server mode to be run on a hardware cluster must match the word-size of the GMS executable. Therefore, they must all run a 32-bit executable, or they must all run a 64-bit executable.

Also, mixing word-sizes of Oracle9*i* Real Application Clusters executables across different databases is not supported in release 8.0, but this restriction does not apply to Oracle executables that are not linked in Oracle Parallel Server mode. In release 8.1 and higher, the GMS process is eliminated, and therefore this restriction is relaxed. Oracle instances that belong to different databases may run with different word-sizes in release 8.1 and higher.

INSTANCES Keyword in PARALLEL Clause

The `INSTANCES` keyword can be used in release 8.1 and higher, but it will be interpreted differently than in past releases. In Oracle7 and release 8.0, the `INSTANCES` keyword could be used in the `PARALLEL` clause of statements such as the following:

- `ALTER CLUSTER`
- `ALTER DATABASE ... RECOVER`
- `ALTER INDEX ... REBUILD`
- `ALTER TABLE`
- `CREATE CLUSTER`
- `CREATE INDEX`
- `CREATE TABLE`

It also could be used in hints. The `INSTANCES` keyword was used to specify the number of Oracle Parallel Server instances to use in a parallel operation.

Also starting with release 8.1, the syntax for specifying degree in a `PARALLEL` clause has changed. You can specify degree simply by placing a number after `PARALLEL`, as in the following example:

```
ALTER TABLE emp PARALLEL 5;
```

However, the `DEGREE` keyword remains valid if you choose to use it. The preceding syntax is equivalent to the following statement:

```
ALTER TABLE emp PARALLEL (DEGREE 5 INSTANCES 1);
```

Regardless of the syntax, the value you specify is the number of query threads used in a parallel operation. Neither syntax will affect how many instances are used to execute a query. The system will determine how many instances to use based on the instances available and the load on each of the instances. So, either syntax will produce the same result.

Continuing to Use the INSTANCES Keyword in Release 8.1 and Higher You can still use the old syntax to specify both `INSTANCES` and `DEGREE` in release 8.1 and higher, but Oracle interprets it as single keyword that specifies the degree. Therefore, the obsolete command syntax is still accepted in release 8.1 and higher, but its interpretation may be different than in past releases. Table 9–23 illustrates the way in which Oracle interprets the possible settings of `INSTANCES` and `DEGREE` if you continue to use the obsolete syntax. The columns in Table 9–23 represent the following:

- The **Degree** column indicates the setting for the `DEGREE` keyword in the `PARALLEL` clause.
- The **Instances** column indicates the setting for the `INSTANCES` keyword in the `PARALLEL` clause.

- The **8.1 Degree** column indicates Oracle's interpretation of the degree in release 8.1 and higher based on the DEGREE and INSTANCES settings.

Table 9–23 Conversion of INSTANCES Keyword in Release 8.1

Degree	Instances	8.1 Degree
Unset or 1	Unset or 1	1
x	DEFAULT	x
x	Unset or 1	x
Unset or 1	DEFAULT	DEFAULT
DEFAULT	y	y
Unset or 1	y	y
DEFAULT	Unset or 1	DEFAULT
x	y	x * y

In the table, x and y are variables representing an integer value.

If you leave a keyword unset, then Oracle uses 1 as its value.

The following scenarios illustrate the way Oracle may behave differently in release 8.1 and higher because of these interpretations:

- Setting DEGREE to x and INSTANCES to 1 does not guarantee that parallel operations use only one instance.
- Setting DEGREE to 1 and INSTANCES to y does not guarantee that parallel operations use only one query thread per instance.
- Setting DEGREE to x and INSTANCES to y does not guarantee either setting. Instead, Oracle attempts to set the degree to x multiplied by y.

Oracle Corporation recommends that you discontinue use of the INSTANCES keyword to avoid unexpected behavior. Also, consider using the PARALLEL_INSTANCE_GROUP initialization parameter.

> **See Also:** *Oracle9i SQL Reference* for more information about the PARALLEL clause and *Oracle9i Database Reference* for information about the PARALLEL_INSTANCE_GROUP initialization parameter.

Database Security

This section describes compatibility and interoperability issues relating to database security.

Password Management

Make the following changes to a version 7 (or earlier) application to enable it to work with version 8 password management:

- Use the version 8 OCI call, `OCISessionBegin()`, to connect to the server. If the server returns SUCCESS_WITH_INFO, then check to see if the password has expired and is still within the grace period. If the password has expired but is still within the grace period, then signal a warning to the user and use the `OCIChangePassword()` call to change the user’s password (the password change call is optional but recommended).
- If the password has expired and the error is returned, then use the version 8 OCI call, `OCIChangePassword()`, to change the user’s password. If `OCIChangePassword()` is not used to change the password, then the password verification routine will not check if the new password is equal to the old password and will not reject the change if they are the same.

If you do not make these changes to Oracle7 applications, then one of the Oracle tools, such as SQL*Plus, will be required to allow the password change after a user’s account expires.

This version 8 password management feature is off by default. If a version 8 server system does not implement the password expiration feature, then no change is required to Oracle7 clients for password management. The DEFAULT profile sets all the parameters to UNLIMITED, and sets the password complexity check routine to NULL.

The password verification routine is exported / imported along with its profile definition. The user's history table also can be imported / exported in version 8.

Oracle7 or Lower Client with Release 8.0 or Higher Server Oracle7 clients use Oracle7 OCI calls to connect to the server; therefore, release 8.0 and higher password expiration cannot be detected. However, other features of release 8.0 and higher password management work for Oracle7 clients. Full release 8.0 and higher password management, including password expiration handling, can operate in Oracle7 clients after you make the minor change of replacing their Oracle7 log in call with the release 8.0 and higher log in call.

Release 8.0 or Higher Client with Oracle7 or Lower Server A release 8.0 or higher client can be coded to work with Oracle7 or lower servers. An example of the code for such clients follows:

```
OCISessionBegin(...) /* call release 8.0 and higher logon OCI call */
if (SUCCESS_WITH_INFO) then
{ /* Check for password expiration and take appropriate action*/
...
OCIChangePassword(...);
...
}
```

Enterprise User Management

This section includes compatibility and interoperability issues related to enterprise user management. This functionality is part of the Oracle Advanced Security feature.

> **Note:** The Oracle Security Server (OSS) component no longer exists in Oracle8*i*; most of its functionality has been integrated into Oracle Advanced Security. Oracle does not provide a tool to migrate from OSS to Oracle Advanced Security.

Interoperability with Release 8.1.5 Release 8.0 Release 8.1.5 and 8.0 servers cannot share global users and roles with release 8.1.6 and higher servers. In addition, current user database links between release 8.1.5 and release 8.1.6 and higher are not supported. Current user database links between release 8.0 and release 8.1.6 and higher are not supported

Interoperability with Oracle7 and Version 6 Releases Because global users cannot be created or authorized on version 7 or version 6 servers, those servers cannot share global users or roles with version 8. Also, current user database links from version 8 to version 6 or version 7 are not supported.

Database Backup and Recovery

This section describes compatibility and interoperability issues related to database backup and recovery.

Recovery Manager

See Also:

- *Oracle9i Recovery Manager Reference* for compatibility and interoperability issues relating to Recovery Manager
- "Upgrading the Recovery Catalog" on page 7-38 for information about upgrading the recovery catalog.

Recovery Manager Commands

Release 8.1 of Recovery Manager introduced changes to some Recovery Manager commands. However, all commands used in prior releases will continue to work with release 8.1 and higher of Recovery Manager.

For example, the `CLONE` command is changed to the `DUPLICATE` command, but the `CLONE` command will continue to work. Also, the `CLONE` option of the `ALLOCATE` and `CONNECT` commands is now the `AUXILIARY` option, but the `CLONE` option will continue to work. Similarly, the `CLONENAME` keyword in the `COPY` and `SET` commands is now `AUXNAME`, but the `CLONENAME` keyword will continue to work.

Backup Management: EBU and Recovery Manager

EBU and Recovery Manager are client-side utilities for managing Oracle database backups. However, for managing version 8 database backups, you must use Recovery Manager. You cannot use EBU with version 8.

Both EBU and Recovery Manager use the Media Management Language (MML) to communicate with third party storage subsystems, such as Legato or EMC. Investments in tape subsystem management modules for EBU and Oracle7 should be reusable under Recovery Manager and version 8. However, backup volume formats are not reusable. You need to write new backups to the storage subsystem under version 8 because Recovery Manager produces a different format, and backups from Oracle7 generally are not useful for version 8 restores.

Note: The scripting language for Recovery Manager is completely different from the scripting language for EBU.

Datafile Backups

A datafile backup taken with Oracle7 cannot be restored with any release of version 8, with the following exception: a backup of an Oracle7 database taken after running the Migration utility can be restored and recovered with any release of version 8. If EBU is used to backup the Oracle7 database, and the database must later be restored for recovery with version 8, then you must use EBU to restore the datafiles prior to recovering them with version 8. If the Oracle7 database is backed up with operating system commands to disk files, then those disk files can be registered with Recovery Manager by using the `CATALOG DATAFILECOPY` command.

A datafile backup taken with version 8 can be restored and recovered with any later release of version 8, if a direct upgrade path between the release that backed-up the file and the release that recovers the file is supported in Table 7–1, "Upgrade Paths" on page 7-2. You can also restore and recover version 8 backups with an earlier release of version 8 if the datafile contents are compatible with the earlier release.

Standby Database

Standby database operates only on release 7.3 and higher of Oracle. The following compatibility restrictions apply to standby databases:

- The primary and standby databases should run on the same operating system. In addition, Oracle Corporation recommends that the primary and standby databases run on the same release of the operating system.
- The primary and standby databases must run the same maintenance release of Oracle. For example, if your primary database is running release 8.1.6, then the standby database can run any production 8.1 release, such as release 8.1.5, 8.1.6, or 8.1.7. However, in this case, the standby database cannot run Oracle7 or release 8.0.

> **See Also:** Your operating system-specific Oracle documentation for more information about operating system requirements for standby database.

Fast-Start On-Demand Rollback and Fast-Start Parallel Rollback

As part of the recovery process, after a session or instance is abnormally terminated, Oracle rolls back uncommitted transactions. Oracle9*i* has two new features to improve rollback performance: fast-start on-demand rollback and fast-start parallel rollback.

When a dead transaction holds a row lock on a row that another transaction needs, fast-start on-demand rollback automatically recovers the data block required by the new transaction. Other data blocks and transactions that do not block any new transaction’s progress are rolled back in the background. Fast-start on-demand rollback is enabled only when you set the `COMPATIBLE` initialization parameter to 8.1.0 or higher.

Fast-start parallel rollback improves background rollback performance by recovering each dead transaction using multiple server processes. You can use fast-start parallel rollback when the `COMPATIBLE` initialization parameter is set to any 8.0 or 8.1 release. Fast-start parallel rollback recovers each dead transaction using multiple server processes only if the following conditions are met:

- There are enough server processes to allocate one or more processes to each dead transaction.
- If `COMPATIBLE` is set to an 8.0 release, then the transaction was created with multiple server processes. If a transaction was created with only one server process, then only one server process is used in the rollback operation. This restriction does not apply if `COMPATIBLE` is set to 8.1.0 or higher.

See Also: *Oracle9i Database Concepts* for more information about fast-start on-demand rollback.

Archiving of Redo Logs

Release 8.1 and higher enables you to archive online redo log files to multiple destinations, including to a local disk-based file or to a specified standby database. The compatibility and interoperability issues described in this section may arise because of this functionality.

Re-Archiving Previously Archived Online Redo Logs Prior to release 8.1, it was possible to re-archive an online redo log that already had been successfully and fully archived. In addition, it was possible to re-archive redo log files to successfully archived destinations.

Starting with release 8.1, the following restrictions apply:

- Successfully archived online redo logs cannot be re-archived.
- Successfully archived destinations cannot be re-archived.

Archive Operation Error Detection Behavior Prior to release 8.1, when any error was detected, an archive operation stopped immediately, reported the error to the alert log, and signaled the error to the user.

Starting with release 8.1, an archive operation does not stop processing unless all of the archive destinations cannot be processed. An error at one or more destinations does not stop the archive operation; the archive operation only stops if all archive destinations cannot be processed. Specifically, archiving to a mandatory is retried once, and archiving failure on the retry halts processing.

LogMiner

LogMiner runs in a release 8.1 or higher instance and can analyze redo log files from any database that meets the following criteria:

- Has the same DBCS (Database Character Set) as the analyzing Oracle instance
- Is running on the same hardware platform as the analyzing Oracle instance
- Is a release 8.0 or higher database

LogMiner does not require a mounted database to analyze redo log files. However, to fully translate the contents of the redo log files, LogMiner requires access to a LogMiner dictionary (catalog). LogMiner uses the dictionary to translate internal object identifiers and data types to object names and external data formats. You can use the PL/SQL package DBMS_LOGMNR_D to extract a database dictionary into an external file for later use in analyzing redo log files. Without a dictionary, LogMiner returns the internal object identifiers and presents data as hex bytes.

Analyzing Archived Redo Log Files from Other Databases You can run LogMiner on an instance of a database while analyzing redo log files from a different database. To analyze archived redo log files from other databases, LogMiner must:

- Access a dictionary file that is both created from the same database as the redo log files and created with the same database character set
- Run on the same hardware platform that generated the log files, although it does not need to be on the same system
- Use redo log files that can be applied for recovery from Oracle release 8.0 and higher

Oracle Media Management API and Proxy Copy

Starting with Oracle Media Management API version 2, proxy copy functionality is supported. If a Recovery Manager proxy backup is attempted, and Oracle is linked with Oracle Media Management API release 1.1, or a version 2 that does not support proxy copy functionality, then Recovery Manager will return an error and the backup will fail.

Distributed Databases

This section describes compatibility and interoperability issues related to distributed databases.

Materialized Views

Prior to release 8.1, an Oracle materialized view always consisted of a materialized view base table and a view on the base table. For example, creating a materialized view `SNAP_EMP` creates a view `SNAP_EMP` and a base table normally called `SNAP$_SNAP_EMP`. In release 8.1 and higher, most materialized views will have only a base table with the same name as the materialized view. The view will not be created.

A view will be added to the materialized view under the following conditions:

- The materialized view was imported from a database prior to release 8.1, such as release 8.0.
- The `COMPATIBLE` initialization parameter is set below 8.1.0.
- The materialized view requires hidden columns (that is, rowid materialized views and fast-refreshable materialized views that contain subqueries).

Note: Importing version 8 snapshots into version 7 databases is not supported.

Oracle Replication

The following compatibility restrictions apply to a replicated environment:

- If you have a replicated environment with different releases of Oracle, then you cannot replicate data that is incompatible on the lower releases. For example, in a replicated environment with a database at 8.1.0 compatibility level and another database at 8.0.0 compatibility level, you cannot replicate data between them if the data is incompatible with release 8.0.
- To improve performance and protect data integrity, a number of Advanced Replication packages that were external prior to release 8.1 have been internalized in release 8.1 and higher. *Oracle9i Replication* contains a list of these internalized packages.

If one or more of your master sites is a release prior to release 8.1, then the `GENERATE_80_COMPATIBLE` flag must be unset or set to TRUE in the following procedures:

- `GENERATE_REPLICATION_SUPPORT`

- `CREATE_SNAPSHOT_REPOBJECT`
- `GENERATE_SNAPSHOT_SUPPORT`

Heterogeneous Services Agents

This section describes compatibility and interoperability issues related to Heterogeneous Services agents.

Interoperability Between Servers of Different Releases Servers at release 8.0.3 and higher can connect to and use Heterogeneous Services agents of any other server at release 8.0.3 and higher. In a connection between servers of different releases, the functionality is limited to that of the lower release.

Multithreaded Service Agents Starting with release 8.1, multithreaded Heterogeneous Services agents are supported. If you have existing agents and you want to take advantage of the multithreaded features, then create the agent initialization file and explicitly start the agents using the Agent Control Utility.

> **See Also:** *Oracle9i Heterogeneous Connectivity Administrator's Guide* for general information about Heterogeneous Services, and for information about creating the agent initialization file and starting the agents using the Agent Control utility.

SQL*Net or Oracle Net

Version 7 and version 8 releases can use SQL*Net V2 or Net8. SQL*Net V1, however, used a different network addressing scheme and *cannot* be used with version 8. Therefore, the following requirements apply to upgraded applications:

- Both the client and server must run SQL*Net V2 or Net8.
- The shared server requires SQL*Net V2 or Oracle Net on the server. Therefore, to connect using the shared server, you also must use SQL*Net V2 or Oracle Net on the client.

Upgrading SQL*Net V1 to SQL*Net V2 or Net8

Make the following changes to upgrade from SQL*Net V1 to SQL*Net V2 or Net8:

- Install SQL*Net V2 or Net8.
- Re-create each connect string as the next version's connect descriptor. SQL*Net V2 uses the syntax outlined in the *SQL*Net Version 2.0 Administrator's*

Guide or and Net8 uses the syntax outlined in the *Oracle Net Services Administrator's Guide*.

- Relink any precompiler programs and Oracle executables that you want to use with SQL*Net V2 or Net8, including SQL*Plus and SQL*Forms.

  See Also: *SQL*Net Version 2.0 Administrator's Guide* and *SQL*Net V2 Migration Guide* for complete instructions about upgrading SQL*Net from V1 to V2. See *Oracle Net Services Administrator's Guide* for complete instructions about upgrading SQL*Net V1 to Net8.

Service Naming and Connection Load Balancing

Release 8.1 and higher supports service naming and connection load balancing for services that include more than one database instance. Each service can include multiple instances, and each instance can include multiple handlers. This support enables clients to access a service rather than a specific database instance, and logically separates the service name from any particular instance name.

To support services that include multiple instances, use the following new parameters in connect descriptors:

- SERVICE_NAME
- INSTANCE_NAME

The new parameters enable connection load balancing by taking requests through the following process:

1. A client program specifies the name of the service to which it wants to connect.
2. The TNS Listener finds the least loaded instance in the service.
3. The TNS Listener finds the least loaded handler in the instance.
4. The TNS Listener redirects the client to the optimal handler, or passes the client connection to the handler, if necessary.

To use connection load balancing, perform the following actions:

- Discontinue the use of the SID parameter in connect descriptors.
- Use the `SERVICE_NAMES` and `INSTANCE_NAME` initialization parameters in your initialization parameter file.
- Use the `SERVICE_NAME` parameter in the `tnsnames.ora` file.

Note: Before configuring the TNS Listener to handle two or more instances with the same instance name, make sure no client programs use connections based on the SID parameter.

See Also: *Oracle Net Services Administrator's Guide* for more information about using connection load balancing and the `SERVICE_NAME` parameter.

Export/Import

Starting with version 5, export dump files are importable into all future major, patch, and maintenance releases of Oracle. Table 9–24 details this support.

Table 9–24 Export Dump File Forward Compatibility

Dump File	Can Be Imported Into Future Releases
Version 5 and Version 6 **Note:** For version 5, only release 5.1.22 and higher export dump files are supported.	Oracle7, Version 8, and all future releases
Oracle7, Release 8.0, and Release 8.1	Release 8.0, Release 8.1, and all future releases

The Export utility makes dump files that are *not* downward compatible with Import utilities of previous maintenance releases and versions. That is, their exported data *cannot* be imported by the Import utilities of previous maintenance releases and versions. So, a release 8.1 export dump file cannot be imported by a release 8.0 Import utility, and a Oracle9*i* export dump file cannot be imported by an Oracle7 Import utility.

However, the contents of a database can be imported into the previous production release if you use the Export and Import utilities of the previous release. Table 9–25 details this support.

Table 9–25 Backward Compatibility Support for Export/Import

To Export Data From	Into Previous Release	Use Export/Import Utility
Release 8.0	Release 7.3	Release 7.3 **Note:** Run the `catexp7.sql` script before exporting.

Table 9–25 Backward Compatibility Support for Export/Import

To Export Data From	Into Previous Release	Use Export/Import Utility
Release 8.1	Release 7.3	Release 7.3 **Note:** Run the `catexp7.sql` script before exporting.
Release 8.1	Release 8.0	Release 8.0

As Table 9–25 indicates, to export version 8 data to an Oracle7 database, you must first run the `catexp7.sql` script on your version 8 database. Then, use the Oracle7 Export utility to export the data.

For example, to export data from a release 8.1 database into an Oracle7 database, complete the following general procedure:

1. Run the `catexp7.sql` script on the release 8.1 database. The `catexp7.sql` script resides in the *`ORACLE_HOME/`*`rdbms/admin` directory.
2. Export the data from the release 8.1 database using the Oracle7 Export utility.
3. Import the dump file into the Oracle7 database using the Oracle7 Import utility.

You do *not* need to run the `catexp7.sql` script if you are moving data from release 8.1 to release 8.0. Also, a version 6 (or earlier) Export utility cannot be used against a version 8 database.

> **See Also:** *Oracle9i Database Utilities* for detailed information about using Export/Import.

Export/Import Usage on Data Incompatible with a Previous Version

When you export data to a previous release, data that is incompatible with the previous release either is not exported at all or is exported with the loss of some features. This applies if you are moving data from release 8.1 into release 8.0, or moving data from version 8 to version 7.

For example, partitioned tables are not exported by the Oracle7 Export utility. If you need to move a version 8 partitioned table to an Oracle7 database, then first reorganize the table into a non-partitioned table. Another example involves procedures that use invoker-rights in release 8.1. If you use the release 8.0 Export utility, then these procedures are exported, but they do not function properly in release 8.0 because release 8.0 does not support invoker-rights. Therefore, in general, if you need to export data to a previous release, then first remove as many incompatibilities with the previous release as possible before you export the data.

Miscellaneous Compatibility and Interoperability Issues

This section describes miscellaneous compatibility and interoperability issues related to your Oracle installation.

2 GB File Size Dependencies

Release 8.0.4 and higher can access files that are larger than 2 GB. However, this access is subject to the following operating system dependencies:

- **File Mode:** Is the file a file system file or a raw device file? Many UNIX systems support greater than 2 GB file sizes only on raw devices.
- **Asynchronous I/O:** Does the operating system support asynchronous I/O on files, for both raw and file system files? Is asynchronous I/O supported for files that are greater than 2 GB?
- **Operating System Revision:** Does your operating system release number support file size greater than 2 GB? For example, in Solaris 2.5.1, a file size of greater than 2 GB is supported only on raw devices. However, in Solaris 2.6, both raw and file system files can be greater than 2 GB.
- **Operating System I/O Subsystem Issues:** Does your operating system require a firmware upgrade to support file size greater than 2 GB? Because support for file size greater than 2 GB is fairly recent, many disk arrays or I/O subsystems need firmware upgrades to support large files. It is important to determine from the operating system vendor which firmware patches are required for large file support.

It is very important to check these operating system dependencies before using files that are greater than 2 GB in size.

10
Upgrading Your Applications

This chapter describes upgrading your current applications and covers the following topics:

- Overview of Upgrading Applications to Oracle9i
- Upgrading Precompiler and OCI Applications
- Upgrading SQL*Plus Scripts
- Upgrading Oracle7 Forms or Oracle Developer Applications

Overview of Upgrading Applications to Oracle9*i*

You do not need to modify existing Oracle7 and version 8 applications that do not use new Oracle9*i* features. Existing applications running against an Oracle9*i* database function the same as they did on prior releases and achieve the same, or enhanced, performance.

Many new features and enhancements are available after migrating or upgrading to Oracle9*i*. Some of these features provide added functionality, while others provide improved performance. Before you upgrade your applications, you should review these new features to decide which ones you want to use.

See Also: *Oracle9i Database New Features* for information about new features available in Oracle9*i*.

Compatibility Issues for Applications

There may be compatibility issues between different releases of Oracle that could affect your applications. These compatibility issues result from differences in the Oracle database server in various releases. Also, in each new release of Oracle, new Oracle reserved words may be added, changes may be made to initialization parameters, and changes may be made to the data dictionary.

When you upgrade or migrate your Oracle database server to a new release, make sure that your applications do not use any Oracle reserved words, that your applications are compatible with the initialization parameters of the server, and that your applications are compatible with the data dictionary of the server. Finally, a new release of Oracle software may require certain operating system versions or certain patch sets to be applied.

See Also:

- "Applications" on page 9-20 for information about compatibility issues that relate to applications.
- Appendix B, Appendix C, and Appendix D for information about changes to initialization parameters and the data dictionary.
- *Oracle9i SQL Reference* for a complete list of Oracle reserved words
- Your operating system-specific Oracle documentation for information about operating system requirements

SQL*Net version 2, Net8, and Oracle Net Services work with various Oracle releases. Thus, Oracle7, version 8, and Oracle9*i* databases can communicate by using SQL*Net version 2, Net8, and Oracle Net Services. SQL*Net version 1, however, uses a different network addressing scheme and *cannot* be used with release 8.0 and higher.

Upgrading Precompiler and OCI Applications

The upgrade path is very similar for precompiler and OCI applications. This section guides you through your upgrade options for these applications and notes differences between precompiler and OCI applications whenever necessary.

Create a test environment before you upgrade your production environment. Your test environment should include your upgraded application and the Oracle9*i* database. Also, your test environment should provide a realistic test of your application.

> **See Also:** *Pro*C/C++ Precompiler Programmer's Guide*, *Pro*COBOL Precompiler Programmer's Guide*, and *Oracle Call Interface Programmer's Guide* for more information about using these programming environments.

Understanding Software Upgrades and Your Client/Server Configuration

To understand your options for upgrading precompiler and OCI applications, you first need to understand the type of software upgrade you are performing and your client/server configuration.

Types of Software Upgrades

Three types of upgrades are possible for both client and server Oracle software.

Version Release Upgrade The upgrade changes the first digit of the release number.For example, upgrading from any Oracle7 release to any version 8 release is a version release upgrade.

Feature Release Upgrade The upgrade changes the second digit of the release number. For example, upgrading from any 8.0 release to any 8.1 release is a feature release upgrade.

Maintenance Release Upgrade The upgrade changes the third digit of the release number. For example, upgrading from release 8.1.6 to release 8.1.7 is a maintenance release upgrade.

Possible Client/Server Configurations

Your precompiler and OCI applications run on the client in a client/server environment, where the Oracle database server is the server. You may use one or more of the following client/server configurations in your environment.

Different Computers The client software and the server software are on different computers, and they are connected through a network. The client and server environments are separate.

Different Oracle Home Directories on the Same Computer The client software and the server software are on the same computer, but they are installed in different Oracle home directories. Again, the client and server environments are separate.

Same Oracle Home The client software and server software are installed in the same Oracle home on the same computer. In this case, any upgrade of the server software is also an upgrade of the client software.

See Also: *Oracle9i Database Concepts* and *Oracle9i Heterogeneous Connectivity Administrator's Guide* for more information about client/server environments.

Compatibility Rules for Applications When Upgrading Oracle Software

This section covers compatibility rules that apply when you upgrade Oracle server software or Oracle client software. The rules are based on the type of software upgrade you are performing and on your client/server configuration.

The following sections contain compatibility rules for the following type of upgrades:

- Upgrading the Oracle Server Software
- Upgrading the Oracle Client Software

Note: This section uses the terms introduced in "Understanding Software Upgrades and Your Client/Server Configuration" on page 10-3.

Upgrading the Oracle Server Software

The following rules apply when you upgrade the Oracle server software.

If You Do Not Change the Client Environment, Then You Do Not Need to Relink. If your client and server are on different computers or are in different Oracle home directories on the same computer, and you upgrade the Oracle server software without changing the client software, then you do not need to precompile, compile, or relink your applications. In these cases, the client software is separate from the server software and will continue to function against the server.

However, if your applications are using the same Oracle home as the Oracle database server, then your server upgrade also upgrades your client software, and you must follow the rules in "Upgrading the Oracle Client Software" on page 10-6.

Note: It is possible to upgrade the Oracle server software but not install the new precompiler or OCI client software when you are using the same Oracle home for both. In this case, the client software is not upgraded. However, such a configuration is not recommended.

Applications Can Run Against Newer or Older Oracle Server Releases When you run a precompiler or OCI application against a database server, Oracle Corporation recommends that the release of the database server software be equal to or higher than the client software release, but this configuration is not strictly required. For example, if your Oracle client software is release 8.0.6, then your Oracle server software *should* be release 8.0.6 or higher to run a precompiler application on the client against the server.

For OCI, Oracle7 client software can run against a release 8.0 or higher Oracle server, and release 8.0 or higher client software can run against an Oracle7 server. However, if the client software is version 6, then the server software must be Oracle7 or lower; in this case, the server software cannot be release 8.0 or higher. Similarly, if the client software is release 8.0 or higher, then the server software must be Oracle7 or higher; in this case, the server software cannot be version 6. Again, if a release 8.0 or higher client is running against an Oracle7 server, then the application cannot use features available in release 8.0 and higher, including object capabilities.

Upgrading the Oracle Client Software

Oracle Corporation recommends that you upgrade your client software to match the current server software. For example, if you upgrade your Oracle database server to release 9.0.1, then Oracle Corporation recommends upgrading the client software to release 9.0.1 as well. Keeping the server and client software at the same release number ensures the maximum stability for your applications. In addition, the latest Oracle client software may provide added functionality and performance enhancements that were not available with previous releases.

The following rules apply when you upgrade the Oracle client software.

Applications Can Be Linked with Newer Libraries The code generated by precompiler applications can be linked with a release of the client library that is equal to or higher than the server release. In addition, Oracle7 and release 8.0 and higher SQLLIB function calls cannot be mixed in the same application and the same transaction.

OCI applications can be linked with a version of the OCI runtime library that is equal to or higher than the version of the OCI library with which the application was developed.

Statically-Linked Applications Do Not Need to be Relinked For statically-linked applications, when you perform any type of upgrade of the client software, you do not need to relink your precompiler and OCI applications. However, relinking is recommended because it may improve performance.

Applications Do Not Need To Be Relinked with Maintenance Release Upgrades When you perform a maintenance release upgrade of the client software, you do not need to relink your applications. Your application continues to function normally without relinking. For example, if you upgrade the client software from release 8.1.6 to release 8.1.7, then you do not need to relink your applications.

Dynamically-Linked Applications Do Not Need To Be Relinked When you perform a version or feature release upgrade of your client software, you do not need to relink your dynamically-linked precompiler and OCI applications. However, relinking is recommended because it may improve performance.

Upgrading Options for Your Precompiler and OCI Applications

You have the following four options for upgrading your precompiler and OCI applications:

Option 1: Leave the application unchanged. Do not relink, precompile, or compile the application, and do not change the application code. The application will continue to work against an Oracle9*i* database.

Option 2: Relink the application with the new Oracle9*i* libraries. Do not precompile or compile the application and do not change the application code.

Option 3: Precompile and/or compile and then relink the application using the new Oracle9*i* software. Do not change the application code.

Option 4: Change the application code to use new Oracle9*i* features. Then, precompile and/or compile and then relink the code.

These options are listed in order of increasing difficulty and increasing potential benefits. That is, Option 1 is the least difficult option, but it offers the least potential benefits, while Option 4 is the most difficult option, but it offers the most potential benefits. These options are discussed in the following sections.

Option 1: Leave the Application Unchanged

You can leave the application unchanged, and it will continue to work with an Oracle9*i* database. The major advantage to this option is that it is simple and easy. In addition, this option requires the least amount of administration, because you do not need to upgrade all of your client computers. If you have a large number of client computers, then avoiding the administrative costs of upgrading all of them can become very important.

The major disadvantage to this option is that your application cannot use the new features that are available in Oracle9*i*. In addition, your application cannot leverage some of the possible performance benefits of Oracle9*i*.

Note: When you perform certain types of upgrades, you are required to relink your application with the new libraries. For example, if you perform a version or feature release upgrade of the client environment, then you must relink. See "Compatibility Rules for Applications When Upgrading Oracle Software" on page 10-4 for information about the rules to follow when you upgrade Oracle software.

Option 2: Relink the Application with the New Oracle9*i* Libraries

You can relink the application with the new Oracle9*i* libraries, without making any code changes and without recompiling. By relinking, your application may benefit from performance improvements that are available only with the new libraries. Remember that you should always relink the application in a test environment before you relink in your production environment.

Note: On operating systems that do not support shared libraries, you must relink your application if you want to include the new libraries in the executable.

Option 3: Precompile and/or Compile the Application Using the New Software

You can precompile or compile the application with the new Oracle9*i* software, without making any code changes. This option requires that you install the new Oracle client software on each client computer. However, you only need to precompile or compile, and relink your application once, regardless of the number of clients you have. The advantages, however, can be quite large.

By recompiling, you perform a syntax check of your application code. Some problems in the application code that were not detected by previous releases of the Oracle software may emerge when you precompile or compile with the new Oracle software. Therefore, precompiling and compiling with the new software often helps you detect and correct problems in the application code that may have gone unnoticed before.

Also, recompiling affords maximum stability for your application, because you are sure that it works with the new Oracle software. Further, your environment is ready for new development using the latest tools and features available. In addition, you may benefit from performance improvements that are available with the new Oracle software only after you recompile and relink.

Option 4: Change the Application Code to Use New Oracle9*i* Features

You can make code changes to your application to take advantage of new Oracle9*i* features. This option is the most difficult, but it can provide the most potential benefits. You gain all of the advantages described in Option 3. In addition, you also benefit from changes to your application that may leverage performance and scalability benefits available with Oracle9*i*. Further, you can add new features to your application that are available only with the new release of Oracle9*i*.

Become familiar with the new Oracle9*i* features by reading *Oracle9i Database New Features*. Also, consult the Oracle documentation for your development environment so that you understand how to implement the features you want to use. For the precompilers, see *Pro*C/C++ Precompiler Programmer's Guide* and *Pro*COBOL Precompiler Programmer's Guide*. For OCI, see *Oracle Call Interface Programmer's Guide*.

When you have decided on the new features you want to use, change the code of your application to use these features. Follow the appropriate instructions in the following sections based on your development environment:

- Changing Precompiler Applications
- Changing OCI Applications

Changing Precompiler Applications Complete the following steps to change your precompiler application to use Oracle9*i* features:

1. Perform one of the following actions based on whether the existing application is an Oracle7 application or a version 8 application:
   - If you have an Oracle7 application, then the existing Oracle7 application may need to be modified, or new applications written, to reflect the differences between Oracle7 and Oracle9*i*.
   - If you have a version 8 application and you want to take advantage of the new Oracle9*i* features, then incorporate code for the new Oracle9*i* functionality into the existing version 8 application.
2. Precompile the application using the Oracle precompiler.
3. Compile the application.
4. Relink the application with the Oracle9*i* runtime library, SQLLIB, which is included with the precompiler.

Changing OCI Applications Complete the following steps to change your OCI application to use Oracle9*i* features:

1. Change your OCI calls in one of the following ways:
   - If your application uses Oracle7 OCI calls, then modify the application to use only new Oracle9*i* OCI calls.
   - If your application uses Oracle7 OCI calls, then incorporate Oracle9*i* OCI calls into the existing application, while still using Oracle7 calls for some operations.
   - If your application uses only version 8 calls, then incorporate the new Oracle9*i* OCI calls into the existing application.
2. Compile the application.
3. Relink the application with the Oracle9*i* runtime library.

Upgrading SQL*Plus Scripts

To use SQL*Plus release 8.0 and higher, a release 8.0 or higher database, and PL/SQL release 8.0 and higher functionality, complete the following steps:

1. Make the following changes to SQL*Plus release 3.x scripts to convert them into scripts that are compatible with SQL*Plus release 8.0 and higher:

   a. If a script contains the line `SET COMPATIBILITY V7`, then change it to `SET COMPATIBILITY NATIVE`, or remove the line so that the default setting is used. In Oracle9*i*, the default setting is `NATIVE`.

   b. Check any `login.sql` and `glogin.sql` files and change any `SET COMPATIBILITY V7` line found to `SET COMPATIBILITY NATIVE`.

2. To use new Oracle9*i* functionality, change existing SQL scripts to use the new Oracle9*i* syntax. Existing SQL scripts run unchanged on Oracle9*i*, and require no modification, if they do not use new Oracle9*i* functionality.

   See Also:

   - *SQL*Plus User's Guide and Reference* to learn about new functionality in SQL*Plus
   - *Oracle9i SQL Reference* for more information about upgrading SQL scripts

---

Note: No changes to PL/SQL packages, procedures, or functions should be required.

Upgrading Oracle7 Forms or Oracle Developer Applications

Forms applications run the same on Oracle7, version 8, and Oracle9*i*. However, review the new features described in *Oracle9i Database New Features* to determine whether any of the new Oracle9*i* features would be beneficial to your applications or might otherwise affect them. Information about the ways in which the Oracle9*i* features interact with forms and developer applications is provided in the Oracle Developer documentation set. Also, the Oracle Developer documentation for your operating system contains instructions for upgrading your forms or developer applications.

Note: New releases of Oracle Developer may introduce new reserved words that are specific to Oracle Developer. Code changes may be required if your application uses any of these new reserved words.

11

Migrating from Server Manager to SQL*Plus

This chapter guides you through the process of changing your Server Manager line mode scripts to work with SQL*Plus. Server Manager is no longer supported in Oracle9*i*. If you run SQL scripts using Server Manager line mode, then you will need to change these scripts so that they are compatible with SQL*Plus, and then run them using SQL*Plus.

This chapter covers the following topics:

- Startup Differences
- Commands
- Syntax Differences

See Also: *SQL*Plus User's Guide and Reference* for detailed information about using SQL*Plus.

Note: For brevity, Server Manager line mode is referred to as Server Manager in the rest of this chapter.

Startup Differences

The methods for starting Server Manager and SQL*Plus are different, and your SQL scripts must be modified to properly start SQL*Plus. The following sections explain the startup differences and provide options for starting SQL*Plus.

Starting Server Manager

To start Server Manager, enter the name of the Server Manager program at a system prompt; the name of this program is operating system-specific. After you start up Server Manager, connect using the `CONNECT` command, as in the following example:

```
CONNECT hr/hr
```

Starting SQL*Plus

The following sections describe various ways to start SQL*Plus.

Starting SQL*Plus with the NOLOG Option

If you want SQL*Plus to behave in the same way as Server Manager, then use the `NOLOG` option when you start SQL*Plus, as in the following example:

```
sqlplus /nolog
```

SQL*Plus starts and you can use the `CONNECT` command to connect as a user.

Starting SQL*Plus with Connect Information

Another option for starting SQL*Plus is to enter the connect information when you start the program. For example, to start SQL*Plus and connect as user `hr` with password `hr`, enter the following:

```
sqlplus hr/hr
```

SQL*Plus starts and connects as user `hr`.

Starting SQL*Plus without Options or Connect Information

To start SQL*Plus without options or connect information, enter the following:

```
sqlplus
```

SQL*Plus prompts you for a user name and password. When you enter a valid user name and password, SQL*Plus starts and connects as the user you specified at the prompts. In your SQL scripts, however, you may not want to prompt the user to enter a user name and password.

Commands

Server Manager and SQL*Plus share certain commands that behave the same in both programs. Other commands, however, behave differently in SQL*Plus than they do in Server Manager. To successfully migrate from Server Manager to SQL*Plus, you need to understand these differences and similarities. The following sections include information about modifying your SQL scripts to use commands that are interpreted correctly by SQL*Plus.

Commands Introduced in SQL*Plus Release 8.1

Table 11–1 lists Server Manager commands that are available in SQL*Plus release 8.1 and higher. You can use these commands in SQL scripts that you run with SQL*Plus.

Note: If you run SQL scripts containing any of these commands in Oracle7 or release 8.0, you must use Server Manager to run these scripts. Versions of SQL*Plus before SQL*Plus release 8.1 will not run scripts containing these commands.

Table 11–1 Commands Introduced in SQL*Plus Release 8.1

Command	**Description**
`ARCHIVE LOG`	Starts or stops automatic archiving of online redo log files, manually (explicitly) archives specified redo log files, or displays information about archives.
`RECOVER`	Performs media recovery on one or more tablespaces, one or more datafiles, or the entire database.
`SET AUTORECOVERY`	`ON` causes the `RECOVER` command to automatically apply the default filenames of archived redo log files needed during recovery. No interaction is needed when `AUTORECOVERY` is set to `ON`, provided the necessary files are in the expected locations with the expected names.
`SET INSTANCE`	Changes the default instance for your session to the specified instance path. Does not connect to a database. The default instance is used for commands when no instance is specified.
`SET LOGSOURCE`	Specifies the location from which archive logs are retrieved during recovery. The default value is set by the `LOG_ARCHIVE_DEST` initialization parameter. Issuing the `SET LOGSOURCE` command without a pathname restores the default location.
`SHOW AUTORECOVERY`	Shows whether autorecovery is enabled.
`SHOW INSTANCE`	Shows the connect string for the default instance. `SHOW INSTANCE` returns the value `LOCAL` if you have not used `SET INSTANCE` or if you have used the `LOCAL` option of the `SET INSTANCE` command.
`SHOW LOGSOURCE`	Shows the current setting of the archive log location. Displays `DEFAULT` if the default setting is in effect, as specified by the `LOG_ARCHIVE_DEST` initialization parameter.
`SHOW PARAMETERS`	Displays the current values of one or more initialization parameters. The `SHOW PARAMETERS` command, without any string following the command, displays all initialization parameters.
`SHOW SGA`	Displays information about the current instance's System Global Area.
`SHUTDOWN`	Shuts down a currently running Oracle instance, optionally closing and dismounting a database. **Note:** The `STARTUP` and `SHUTDOWN` commands in SQL*Plus release 8.1 are not supported against an Oracle7 server.
`STARTUP`	Starts an Oracle instance with several options, including mounting and opening a database. **Note:** The `STARTUP` and `SHUTDOWN` commands in SQL*Plus release 8.1 are not supported against an Oracle7 server.

Commands Common to Server Manager and SQL*Plus

The commands listed in Table 11–2 are available in both Server Manager and SQL*Plus, and have been available in both programs in past releases of Oracle. You do not need to alter these commands in your SQL scripts to use SQL*Plus.

Note: There may be minor formatting differences in the output for these commands in the two programs.

Table 11–2 Server Manager Commands Corresponding to Existing SQL*Plus Commands **(Page 1 of 2)**

Command	Description
`CONNECT`	Connects to a database using the specified user name.
`DESCRIBE`	Describes a function, package, package body, procedure, table, or view. For example, for a table, displays the definitions of each column in the table.
`REMARK`	Enters a comment, typically in SQL script files.
`SET COMPATIBILITY`	Sets compatibility mode to `V7`, `V8`, or `NATIVE`. The compatibility mode setting affects the specification of character columns, integrity constraints, and rollback segment storage parameters. `NATIVE` matches the version of the database.
`SET ECHO`	Controls whether the `START` command lists each command in a command file as the command is executed. `ON` lists the commands; `OFF` suppresses the listing.
`SET NUMWIDTH`	Sets the default width for displaying numbers.
`SET SERVEROUTPUT`	Controls whether to display the output (that is, `DBMS_OUTPUT.PUT_LINE`) of stored procedures or PL/SQL blocks in SQL*Plus. `OFF` suppresses the output of `DBMS_OUTPUT.PUT_LINE`; `ON` displays the output.
`SET TERMOUT`	Controls the display of output generated by commands executed from a command file. `OFF` suppresses the display so that you can spool output from a command file without seeing the output on the screen. `ON` displays the output.

Table 11–2 Server Manager Commands Corresponding to Existing SQL*Plus Commands **(Page 2 of 2)**

Command	Description
`SHOW ALL`	Lists all of the system variables set by the `SET` command in alphabetical order, except ERRORS, PARAMETERS, and SGA.
`SHOW ERRORS`	Shows the errors generated from the last compilation of a procedure, package, or function, if any.
`SPOOL`	Stores query results in an operating system file and, optionally in SQL*Plus, sends the file to a printer. **Note:**The extension of spool files may differ between SQL*Plus and Server Manager. To ensure an extension, specify it when you issue the SPOOL command. Also, SQL*Plus may format white space in terminal output using tab characters in place of repeated spaces. Use `SET TAB OFF` in SQL*Plus to prevent this replacement. Server Manager never outputs tab characters.

SQL*Plus Equivalents for Server Manager Commands

Table 11–3 lists the SQL*Plus commands that correspond to Server Manager commands with different names. If you are using any of these Server Manager commands in SQL scripts, then modify the scripts to use the SQL*Plus commands instead.

Table 11–3 SQL*Plus Equivalents for Server Manager Commands **(Page 1 of 2)**

Server Manager Commands	SQL*Plus Commands	Description

SET CHARWIDTH
SET DATEWIDTH
SET LONGWIDTH
	COLUMN FORMAT	You can use the COLUMN FORMAT command in SQL*Plus to set the column width of character columns, date columns, and number columns. In your SQL scripts, replace the SET CHARWIDTH, SET DATEWIDTH, and SET LONGWIDTH Server Manager commands with the SQL*Plus command COLUMN FORMAT.
		
		Use COLUMN FORMAT for all character columns to be changed. There is no equivalent command to change all character columns with one command.
		
		For example, suppose you have the following entry in a SQL script:
		
		`SET CHARWIDTH 5`
		
		This command sets the width for all character columns to 5 in Server Manager.
		
		To specify that a particular column, such as ENAME, display with a width of 5 characters, enter the following SQL*Plus command:
		
		`COLUMN ENAME FORMAT A5`
		
		Use COLUMN FORMAT for all character columns to be changed. There is no equivalent command to change all character columns with one command.
		
		Use COLUMN FORMAT for all date columns to be changed. There is no equivalent command to change all date columns with one command.
		
		Use SET LONG to specify how much of the LONG column to fetch and display.

Table 11–3 SQL*Plus Equivalents for Server Manager Commands **(Page 2 of 2)**

Server Manager Commands	SQL*Plus Commands	Description
SET STOPONERROR	WHENEVER SQLERROR WHENEVER OSERROR	Use the WHENEVER SQLERROR and WHENEVER OSERROR commands to direct SQL*Plus to either exit or continue whenever a SQL error or operating system error occurs. Use these commands in your SQL scripts instead of the Server Manager command SET STOPONERROR. For both WHENEVER SQLERROR and WHENEVER OSERROR, the EXIT clause directs SQL*Plus to exit, while the CONTINUE clause directs SQL*Plus to continue. Other terms and clauses are also available for these commands.

Possible Differences in the SET TIMING Command

The `SET TIMING` command is available in both Server Manager and SQL*Plus, but this command may function differently in the two programs on some operating systems. Check your operating system-specific Oracle documentation for more information. If the `SET TIMING` command functions differently in these two programs on your operating system, then modify your SQL scripts so that this command functions properly with SQL*Plus.

Server Manager Commands Unavailable in SQL*Plus

The following Server Manager commands are unavailable in SQL*Plus release 8.1 and higher:

- `SET MAXDATA`
- `SET RETRIES`

Remove these commands from your SQL scripts.

Syntax Differences

The following sections explain the syntax differences between Server Manager and SQL*Plus. Modify your SQL scripts to conform with SQL*Plus syntax conventions before you attempt to run your scripts using SQL*Plus.

Comments

SQL*Plus recognizes the following types of comments:

- the SQL*Plus REMARK command (or REM)
- the SQL comment delimiters, /* ... */
- the ANSI/ISO comments, --

The *SQL*Plus User's Guide and Reference* provides detailed information about using these types of comments in SQL*Plus scripts.

Server Manager supports these types of comments, but the behavior is different for some of them. Also, certain types of comments are available in Server Manager, but not in SQL*Plus. The sections below discuss each type of comment and the syntax differences between Server Manager and SQL*Plus.

REMARK Command (or REM)

In general, the `REMARK` command works the same in Server Manager and SQL*Plus, and you do not need to change the occurrences of the `REMARK` command in your SQL scripts. There is, however, one difference: SQL*Plus interprets a hyphen that terminates a REMARK command differently than Server Manager. See "Hyphens Used as Dividing Lines" on page 11-13 for information about this difference.

SQL Comment Delimiters, /* ... */

In Server Manager, the SQL comment delimiters can be placed after a semicolon (;), but in SQL*Plus, placing a SQL comment delimiter after a semicolon is not allowed. Except for this one difference, SQL comment delimiters work the same in Server Manager and SQL*Plus.

If your SQL scripts contain any SQL comment delimiters placed after a semicolon , then either move the comment to its own line, or remove the semicolon and place a slash (/) on the next line to end the SQL statement.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
SELECT * FROM hr.employees
    WHERE job = 'CLERK'; /* Includes only clerks. */
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM hr.employees
    WHERE job = 'CLERK';
/* Includes only clerks. */
```

```
SELECT * FROM hr.employees
    WHERE job = 'CLERK' /* Includes only clerks. */
    /
```

ANSI/ISO Comments, --

In Server Manager, the ANSI/ISO comments can be placed after a semicolon (;), but in SQL*Plus, placing an ANSI/ISO comment after a semicolon is not allowed. Except for this one difference, ANSI/ISO comments work the same in Server Manager and SQL*Plus.

If your SQL scripts contain any ANSI/ISO comments that are placed after a semicolon, then either move the comment to its own line, or remove the semicolon and place a slash (/) on the next line to end the SQL statement.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
SELECT * FROM hr.employees
    WHERE job = 'CLERK'; -- Includes only clerks.
```

In SQL*Plus, replace this code with either of the following entries:

```
SELECT * FROM hr.employees
    WHERE job = 'CLERK';
-- Includes only clerks.

SELECT * FROM hr.employees
    WHERE job = 'CLERK' -- Includes only clerks.
    /
```

Server Manager Pound (#) Comments

Server Manager supports the use of the pound sign (#) to indicate a comment line. If your scripts contain these comments, then change the '#' to '--' to run the scripts using SQL*Plus.

For example, suppose you have the following Server Manager code in one of your SQL scripts:

```
# This statement returns only clerks.
SELECT * FROM hr.employees
    WHERE job = 'CLERK';
```

In SQL*Plus, replace this code with the following entry:

```
-- This statement returns only clerks.
SELECT * FROM hr.employees
    WHERE job = 'CLERK';
```

Blank Lines

Server Manager ignores blank lines within SQL statements, but when SQL*Plus encounters a blank line the default behavior is to stop recording the statement and return to the prompt.

Both products allow blank lines between distinct SQL statements. This section only applies to blank lines between clauses of SQL statements.

In SQL*Plus, the `SET SQLBLANKLINES` command alters the way blank lines are handled. When `SQLBLANKLINES` is set to `OFF`, the default setting, and there is a SQL statement containing a blank line, SQL*Plus buffers the statement at the blank line, returning to the prompt without executing the statement. This behavior allows interactive users to abort and buffer an unwanted SQL command, or to perform other SQL*Plus commands before executing or editing this buffered SQL command.

If any of your SQL scripts contain blank lines within SQL statements, then either set `SQLBLANKLINES` to `ON`, or remove the blank lines before you run these scripts using SQL*Plus.

For example, suppose you have the following SQL statement in one of your SQL scripts:

```
SELECT empno, ename, sal, comm

    FROM hr.employees

    WHERE job = 'MANAGER';
```

Either set `SQLBLANKLINES` to `ON`, or delete the blank lines:

```
SELECT empno, ename, sal, comm
    FROM hr.employees
    WHERE job = 'MANAGER';
```

If you do not remove the blank lines or set `SQLBLANKLINES` to `ON`, then SQL*Plus will treat each blank line of code as a command terminator.

The value of SQLBLANKLINES does not affect blank lines in PL/SQL blocks. These are always treated as part of the block and do not return to the SQL*Plus prompt.

Interactive users can terminate SQL or PL/SQL statements by entering a period on a line by itself, regardless of the value of SQLBLANKLINES.

The Hyphen Continuation Character

SQL*Plus supports the use of a hyphen as a continuation character for long SQL statements or SQL*Plus commands. For example, you can use the continuation character in the following way:

```
SELECT empno, ename, sal, comm FROM hr.employees -
WHERE job = 'MANAGER';
```

Server Manager does not support the use of a hyphen as a continuation character, but you may use hyphens for other purposes in your SQL scripts. If you do, then SQL*Plus may interpret a hyphen as a continuation character, which can cause unexpected output.

The following sections provide scenarios in which SQL*Plus interprets the use of hyphens in SQL scripts as continuation characters, when the hyphens were meant for another purpose. Check your SQL scripts for the use of hyphens and modify them to avoid scenarios similar to those described below.

Hyphens Used as Dividing Lines

Your SQL scripts may use a long row of hyphens following a REMARK command as a dividing line in the code. Consider the following sample lines from a SQL script:

```
Rem ---------------------------------------------------------------------
SELECT empno, ename, job
    FROM hr.employees;
```

In this statement, SQL*Plus interprets the first line of the SELECT statement as a continuation of the previous line, which is a REMARK comment. Therefore, the FROM line is interpreted as the first line of a SQL statement, and SQL*Plus returns the following error:

```
unknown command beginning "FROM hr..." - rest of line ignored.
```

If you use hyphens as dividing lines in your SQL scripts, then remove the REM command preceding the hyphens before you run the scripts using SQL*Plus.

Hyphens Used as Minus Signs

Because the hyphen is the same keyboard character as the minus sign, you may have a hyphen at the end of a line. Consider the following sample lines from a SQL script:

```
CREATE TABLE xx (
    a int,
    b int,
    c int);

INSERT INTO xx VALUES (10, 20, 30);

SELECT a + b -
    c FROM xx;
```

SQL*Plus interprets the 'c' as an alias because the minus symbol is interpreted as a continuation character:

```
SELECT a + b c FROM xx;
```

Therefore, SQL*Plus returns the following unexpected output:

```
         C
----------
        30
```

Server Manager, however, interprets this code as the following:

```
SELECT a + b - c FROM xx;
```

Therefore, Server Manager returns the following expected output:

```
A+B-C
----------
         0
```

Make sure you do not have a minus sign at the end of a line in your SQL scripts.

Ampersands

SQL*Plus interprets an ampersand (&) as a substitution variable, whereas Server Manager interprets an ampersand as a normal string. If the text following the ampersand does not have a defined value, then SQL*Plus interprets it as an undefined value and prompts the user for input, even if the ampersand is enclosed in a comment. Therefore, ampersands can cause unexpected output in SQL*Plus.

If you have SQL scripts that use ampersands as normal text strings, then you have two options:

- Use the `SET ESCAPE` command to place an escape character before each ampersand.
- Use the `SET DEFINE OFF` command to disable the recognition of substitution variables.

Note: Do not use the `SET DEFINE OFF` command if you have other, valid substitution variables; if you do, then the other variables will not be recognized.

For example, the following SQL statement prompts the user for input in SQL*Plus:

```
CREATE TABLE "Employees & Managers" (
    Employees varchar(16),
    Managers varchar(16));

Enter value for managers:
```

Using the SET ESCAPE Command

To avoid the user prompt, you can use the `SET ESCAPE` command to set an escape character. Then, place the escape character before the ampersand. A backslash (\) is often used as an escape character.

To avoid the prompt in the preceding example by using the `SET ESCAPE` command, change the entry to the following:

```
SET ESCAPE \

CREATE TABLE "Employees \& Managers" (
    Employees varchar(16),
    Managers varchar(16));
```

Using the SET DEFINE OFF Command

To avoid the prompt in the preceding example by using the `SET DEFINE OFF` command, change the entry to the following:

```
SET DEFINE OFF

CREATE TABLE "Employees & Managers" (
    Employees varchar(16),
    Managers varchar(16));
```

CREATE TYPE and CREATE LIBRARY Commands

SQL*Plus treats the `CREATE TYPE` and `CREATE LIBRARY` commands as PL/SQL blocks. Therefore, in SQL*Plus, you must use a slash (/) on a separate line to end these commands, while Server Manager allows you to end these commands with a semicolon (;).

If you end any `CREATE TYPE` or `CREATE LIBRARY` command with a semicolon in your SQL scripts, then remove the semicolon and place a slash (/) on the next line. For example, the following SQL statements are not recognized by SQL*Plus:

```
CREATE OR REPLACE TYPE sys.dummy AS OBJECT (data CHAR(1));
CREATE OR REPLACE LIBRARY DBMS_SPACE_ADMIN_LIB TRUSTED AS STATIC;
```

Edit these statements in the following way before you run them with SQL*Plus:

```
CREATE OR REPLACE TYPE sys.aq$_dummy_t AS OBJECT (data CHAR(1))
/
CREATE OR REPLACE LIBRARY DBMS_SPACE_ADMIN_LIB TRUSTED AS STATIC
/
```

COMMIT Command

SQL*Plus requires that the `COMMIT` command be terminated either with a semicolon (;) or a slash (/), but Server Manager allows the `COMMIT` command with no terminator. Therefore, if you use the `COMMIT` command in your SQL scripts without a terminator, then edit these scripts to include a terminator.

For example, suppose you have the following `COMMIT` command in a SQL script:

```
commit
```

Include a terminator for the command, as shown in either of the following examples:

```
commit;

commit
/
```

12

Migration Issues for Physical Rowids

Physical rowids in release 8.0 and higher embody new internal and external formats that enable you to use some new release 8.0 and higher features, including partitioning and global indexes.

> **See Also:** *Oracle9i Application Developer's Guide - Fundamentals* and *Oracle9i Database Concepts* for more information.

This chapter covers the following topics:

> **Note:** In the rest of this chapter, references to version 8 include all version 8 and Oracle9*i* releases. Also, the word "rowid" means "physical rowid". This chapter does not discuss the UROWID (universal rowid) datatype. See Chapter 9, "Compatibility and Interoperability", for compatibility issues relating to the UROWID datatype.

Migrating Applications and Data

Rowids can be stored in columns of ROWID datatype and in columns of character type. Stored version 7 rowids become invalid after migration of the version 7 database to version 8. Therefore, stored version 7 rowids must be converted to version 8 format.

Applications

Applications that do not attempt to manually assemble and disassemble rowids do not need to be changed or recompiled because the new rowids fit the current storage requirements for host variables.

Applications that attempt to manufacture or analyze the contents of rowids must use the `DBMS_ROWID` package, provided in release 8.0 and higher, to deal with the format and contents of the new rowids. This package contains functions that extract the information that was available directly from an Oracle7 rowid (including file and block address), plus the data object number.

Data

The columns that contain rowid values (in `ROWID` datatype format or in character format) must be migrated if they point to tables that were migrated to version 8. Otherwise, it will not be possible to retrieve any rows using their stored values. On the other hand, if the rowid values stored in the version 8 tables still point to version 7 or version 6 tables, then you do not need to migrate the columns.

Columns are migrated in two stages: definition migration and data migration. The column definition is adjusted automatically during version 7 to version 8 dictionary migration. The maximum size of rowid user columns is increased to the size of the extended disk rowids, changing the LENGTH column of COL$ for rowid columns from six to ten bytes.

The data migration can be performed only *after* the system has been opened in version 8. You can migrate different tables at different times or multiple tables in parallel. Make sure the migration is done *before* the version 7 database file limit is exceeded, thereby guarding against the creation of ambiguous block addresses.

You can use existing rowid refresh procedures that are available at your installation, or the version 8 DBMS_ROWID functionality, to migrate stored rowids from version 7 format to version 8 format.

Data migration by the Migration utility or the Oracle Data Migration Assistant applies only to rowids stored in a user-defined column. All system-stored rowids (such as in indexes) remain valid after migration by the Migration utility or the Oracle Data Migration Assistant, and do not require specific actions to be migrated. Also, indexes are not invalidated because, during migration to version 8 by the Migration utility or the Oracle Data Migration Assistant, indexes can continue to use the restricted ROWID datatype format.

Note: Importing a column containing rowids should produce a message warning that special attention might be required to re-establish the validity of the rowids. Special attention is necessary for *all* rowids being imported. Thus, migration by Export/Import requires special attention for *every* column containing rowids (not just for user-defined columns).

The DBMS_ROWID Package

The `DBMS_ROWID` PL/SQL package is provided in release 8.0 and higher and contains the following functionality:

- Creation of rowids in version 7 and version 8 format
- Interpretation of version 7 and version 8 rowids
- Conversion between version 7 and version 8 rowids

Migration of the stored rowids can be accomplished using conversion functions, as described in the following sections.

Rowid Conversion Types

You must specify the type of rowid being converted, because the rowid conversion functions perform the conversion differently depending on whether the rowid is stored in the user column of `ROWID` datatype, or in the user column of `CHAR` or `VARCHAR` datatype.

For a column of `ROWID` datatype, the caller of the conversion procedures must pass the following value as a procedure parameter:

```
rowid_convert_internal constant integer := 0;
```

For a column of CHAR or VARCHAR datatype, the caller of the conversion procedures must pass the following value as a procedure parameter:

```
rowid_convert_external constant integer := 1;
```

Rowid Conversion Functions

The following functions perform the rowid conversion:

- ROWID_TO_EXTENDED converts a rowid from the Oracle7 (restricted) format to the version 8 (extended) format.
- ROWID_TO_RESTRICTED converts a rowid from the version 8 (extended) to the Oracle7 (restricted) format.
- ROWID_VERIFY checks whether a given rowid can be converted from Oracle7 format to version 8 format.

The following sections contain detailed information about the ROWID_TO_EXTENDED and ROWID_VERIFY procedures.

The ROWID_TO_EXTENDED Conversion Procedure

ROWID_TO_EXTENDED uses the following parameters:

- **Rowid -** specifies the rowid to be converted (in External Character format).
- **Schema Name -** specifies the schema name of the table that contains a row whose rowid will be converted to the extended format.
- **Table Name -** specifies the table name of the table that contains a row whose rowid will be converted to the extended format.
- **Conversion Type -** specifies the type of rowid being converted.

> **See Also:** "Rowid Conversion Types" on page 12-3 for more information.

ROWID_TO_EXTENDED returns a version 8 (extended) rowid in External Character format, and its parameters are interpreted in the following way:

- If the schema name and table name for the target table are not specified (null), then ROWID_TO_EXTENDED attempts to fetch the page specified by the rowid to be converted. It will treat the file number stored in this rowid as the absolute file number, which can cause problems if the file has been dropped and its number has been reused prior to the migration. If the fetched page belongs to a valid table, then the rowid will be converted to an extended format using the Data Object ID of this table, but this conversion is very inefficient, and is only recommended as a last resort, when the target table is not known. You still must know the correct table name when using the converted value.
- If the schema name and table name is given (a preferred approach), then ROWID_TO_EXTENDED will verify SELECT authority on the table and convert the rowid to an extended format using the Data Object Number of this table. There is no guarantee that the converted rowid actually references a real row in this table, neither at the time of conversion nor at the time when the rowid is used.
- If a null value is supplied for the rowid, then the procedure ignores the table specification and returns a null value.
- If a value of 0, or, more generally, <n>0.<m>0.<p>0 is supplied for rowid, then the table name is ignored and a restricted rowid of the form 00000000.0000.0000 is returned.
- If a version 8 rowid is supplied, then the data object in the rowid is verified against the actual data object number (which depends on the table name specification). If these two numbers do not match, then the INVALID ROWID error appears; otherwise, the original rowid is returned.

ROWID_VERIFY

A rowid verification procedure, ROWID_VERIFY, is provided with version 8. This procedure uses the same parameters as ROWID_TO_EXTENDED and returns 0 if the rowid can be converted successfully to extended format; otherwise, it returns 1.

However, ROWID_VERIFY returns security violation errors, or an "object not found" error, if the user does not have SELECT authority on the underlying table, or if the table does not exist. ROWID_VERIFY can be used to identify bad rowids prior to migration using the ROWID_TO_EXTENDED procedure.

Conversion Procedure Examples

The following are examples of conversion procedures for rowids:

Example 1

Assume a table SCOTT.T contains a column C of ROWID datatype format. All these rowids reference a single table, SCOTT.T1.

The values of column C can be converted to extended format using the following statement:

```
UPDATE SCOTT.T SET C = DBMS_ROWID.ROWID_TO_EXTENDED(C,’SCOTT’,’T1’,0);
```

Example 2

In a more general situation, rowids stored in column C may reference different tables, but the table name can be found based on the values of some other columns in the same row. For example, assume that the column TNAME of the table T contains a name of the table which is referenced by a rowid from column C.

In this case, the values in column C can be converted to extended format using the following statement:

```
UPDATE SCOTT.T SET C = DBMS_ROWID.ROWID_TO_EXTENDED(C,’SCOTT’,TNAME,0);
```

Example 3

You can use the ROWID_TO_EXTENDED function in the CREATE ... AS SELECT statement. This use may be desirable in some cases because conversion can increase the size of the user column of ROWID datatype (typically from 6 bytes to 10 bytes, although this depends on a specific port) which may create indirect rows.

In this case, CREATE ... AS SELECT may be a better choice than UPDATE:

```
CREATE TABLE SCOTT.TNEW (A, B, C)
   AS SELECT A, B, DBMS_ROWID.ROWID_TO_EXTENDED(C, ’SCOTT’,’T1’,0) FROM SCOTT.T;
```

Example 4

If the target table for rowids stored in column C is not known, then conversion can be accomplished using the following statement:

```
UPDATE SCOTT.T SET C = DBMS_ROWID.ROWID_TO_EXTENDED(C,NULL,NULL,0);
```

Example 5

The following SQL statement may be used to find bad rowids prior to conversion:

```
SELECT ROWID,C FROM SCOTT.T WHERE DBMS_ROWID.ROWID_VERIFY(C,NULL,NULL,0)=1;
```

Snapshot Refresh

The version 8 ROWID datatype format forces all rowid snapshots to perform a complete refresh when both master and snapshot sites are upgraded to version 8.

> **See Also:** Appendix G, "Migration and Compatibility for Replication Environments", for more information about replication compatibility.

Version 7 and Version 6 Client Compatibility Issues

Version 7 and version 6 clients can access a version 8 database, and version 8 clients can access a version 7 or version 6 database. Binary and character values of the pseudo column ROWID and of columns of datatype ROWID that are returned by a prior database to a version 8 database are always in restricted format, because the prior system cannot recognize the extended format ROWID.

The DBMS_ROWID package supplied with version 8 can be used for interpreting the contents of the version 7 rowids and for creating the rowids in version 7 format.

A version 7 or version 6 client accessing a version 8 database receives the rowid in version 8 extended format. Therefore, the client cannot interpret the contents of rowids returned by the version 8 server.

Version 8 snapshot compatibility is restricted to release 7.1.4 and higher. Further, when a master site is upgraded, the version 8 upgrade script invalidates the logs so that snapshots are forced to do a complete refresh before they can do fast refreshes again.

ROWID Migration and Compatibility Issues

For backward compatibility, the restricted form of the ROWID is still supported. These ROWIDs exist in massive amounts of Oracle7 data, and the extended form of the ROWID is required only in global indexes on partitioned tables. New tables always get extended ROWIDs.

See Also: *Oracle9i Database Administrator's Guide*.

It is possible for an Oracle7 client to access an Oracle9*i* database. Similarly, an Oracle9*i* client can access an Oracle7 Server. A client in this sense can include a remote database accessing a server using database links, as well as a client 3GL or 4GL application accessing a server.

There is more information on the ROWID_TO_EXTENDED function in the *Oracle9i Supplied PL/SQL Packages and Types Reference*.

Accessing an Oracle7 Database from an Oracle9*i* Client

The ROWID values that are returned are always restricted ROWIDs. Also, Oracle9*i* uses restricted ROWIDs when returning a ROWID value to an Oracle7 or earlier server.

The following ROWID functionality works when accessing an Oracle7 Server:

- Selecting a ROWID and using the obtained value in a WHERE clause
- WHERE CURRENT OF cursor operations
- Storing ROWIDs in user columns of ROWID or CHAR type
- Interpreting ROWIDs using the hexadecimal encoding (not recommended, use the DBMS_ROWID functions)

Accessing an Oracle9*i* Database from an Oracle7 Client

Oracle9*i* returns ROWIDs in the extended format. This means that you can only:

- Select a ROWID and use it in a WHERE clause.
- Use WHERE CURRENT OF cursor operations.
- Store ROWIDs in user columns of CHAR(18) datatype.

Import and Export

It is not possible for an Oracle7 client to import an Oracle9*i* table that has a ROWID column (not the ROWID pseudocolumn), if any row of the table contains an extended ROWID value.

Frequently Asked Questions About Rowid Migration

Q: Is there any version 8 restriction on a version 7 import client?

A: A version 7 client cannot import a version 8 table with a ROWID user column if a row of this table contains the extended rowid value.

Q: Do Forms3 (and Forms4) understand the new ROWID **datatype format for base table updates?**

A: Forms applications which intend to access version 8 databases have to be relinked using the patch #380655.

Q: How do the version 8 rowid changes affect PRO* precompiled programs?

A: Programs that use rowids but do not rely on their format are not affected. Programs that rely on the version 7 ROWID datatype format must be modified to use the new package, DBMS_ROWID.

Q: Do "WHERE CURRENT of CURSOR" operations still work?

A: Yes, even when accessing a version 8 server from a version 7 or version 6 client or when accessing a version 7 or version 6 server from a version 8 client.

Q: I currently use dynamic SQL and bind as internal ROWID **datatype format. Will I need to** malloc() **more space?**

A: Version 8 rowids fit into the version 7 storage requirements for host variables; therefore, no changes or additional space allocations are necessary.

Q: Can I still define a column of my table to be of ROWID **datatype?**

A: Columns can still be defined of ROWID datatype. The ROWID column requires 10 bytes instead of the 6 bytes required in version 7. However, in most cases, this is not recommended because the ROWID values must be maintained manually.

Q: I rely on the version 7 ROWID **datatype format at present. Will the conversion algorithm be documented?**

A: The new version 8 ROWID datatype format is not documented for such use. However, version 8 provides the DBMS_ROWID (PL/SQL) package to interpret version 8 rowid contents.

Q: Will I need to rebuild any indexes?

A: Only indexes built on a column that stores the old ROWID datatype format need to be rebuilt after data migration.

Q: I use ROWID datatype in older PL/SQL, RPC, or from FORMS. Will this continue to work?

A: The format in which rowids are returned into host variables of ROWID datatype will be the same, and generally no change is needed, except in the following specific known case:

A remote mapped query from a version 7 server to a version 8 database across a dblink (considered a heterogeneous dblink) terminates with an ORA-3116 error upon a rowid fetch as a type DTYRID (without CHR conversion) through OCI. The following are ways to avoid this problem:

- Using rowid fetches as type DTYCHR instead invokes an implicit conversion and avoids the problem.
- Using SQLT_RID and a patch (available from Oracle) on the version 7 server avoids the problem without invoking CHR conversion.

13

Downgrading to Release 8.1

The information in this chapter only applies to release 9.0.1 installations of Oracle. The term **downgrading** describes transforming an Oracle database into a previous release of the same version, such as transforming a database from release 8.1.7 to release 8.1.5. The term **downgrading** also describes transforming an Oracle database into a previous version, such as transforming a database from Oracle9*i* to Oracle7. This chapter describes downgrading to release 8.1 of Oracle. If you want to downgrade to Oracle7 or release 8.0, then see Chapter 15, "Downgrading to an Older Release of Oracle".

This chapter covers the following topics:

- Perform a Full Offline Backup
- Remove Incompatibilities
- Reset Database Compatibility
- Downgrade the Database

> **See Also:** Some aspects of downgrading are operating system-specific. See your operating system-specific Oracle documentation for additional operating system-specific instructions about downgrading.

Perform a Full Offline Backup

Perform a full offline backup of your release 9.0.1 database before you downgrade.

> **See Also:** *Oracle9i User-Managed Backup and Recovery Guide* for more information

Remove Incompatibilities

The process of removing incompatibilities depends on the release to which you are downgrading. First, check the compatibility level of your database to see if your database might have incompatibilities with the release to which you are downgrading.

Checking the Compatibility Level of Your Database

If the compatibility level of your database is higher than the release to which you are downgrading, then your database may have incompatibilities with the previous release that must be removed before you downgrade. Your compatibility level matches the setting of the `COMPATIBLE` initialization parameter. Check your `COMPATIBLE` initialization parameter setting by issuing the following SQL statement:

```
SQL> SELECT name, value, description FROM v$parameter
       WHERE name='compatible';
```

You do not need to remove incompatibilities if the `COMPATIBLE` parameter is set to the release to which you are downgrading or lower. For example, if you are downgrading to release 8.1.6 and the `COMPATIBLE` parameter is set to 8.1.6 or lower, then you do not need to remove incompatibilities. In this case, no incompatibilities exist in your database with the release to which you are downgrading, and you can skip the rest of this section and procede to "Downgrade the Database" on page 13-4.

However, if the `COMPATIBLE` parameter is set higher than the release to which you are downgrading, then some incompatibilities may exist. For example, if you are downgrading to release 8.1.7 and `COMPATIBLE` is set to 9.0.0 or higher, then incompatibilities may exist.

Follow the instructions in Chapter 14, "Removing Incompatibilities Before Downgrading to Release 8.1" to remove incompatibilities with the release to which you are downgrading.

Reset Database Compatibility

After you have removed all of the incompatibilities with the release to which you are downgrading, reset the compatibility level of the database to the prior release by completing the following steps:

1. Log in to the system as the owner of the Oracle home directory.
2. If you are using any initialization parameters that were added in a release higher than the release to which you are downgrading, then remove them from your initialization parameter file.

   See Also: Appendix B, "Changes to Initialization Parameters" for lists of parameters added in each release

3. At a system prompt, change to the *ORACLE_HOME*/rdbms/admin directory.
4. Start SQL*Plus.
5. Connect to the database instance as a user with SYSDBA privileges.
6. Start up the instance in RESTRICT mode:

   ```
   SQL> STARTUP RESTRICT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.

7. If you plan to lower the value of COMPATIBLE from 9.0.0 to 8.1.x, then run utlpitl.sql:

   ```
   SQL> @utlpitl.sql
   ```

   The utlpitl.sql script removes PDML ITL incompatibilities.

8. Run ALTER DATABASE RESET COMPATIBILITY:

   ```
   SQL> ALTER DATABASE RESET COMPATIBILITY;
   ```

   See Also: "About ALTER DATABASE RESET COMPATIBILITY" on page 9-8 for more information.

9. Shut down the instance:

   ```
   SQL> SHUTDOWN IMMEDIATE
   ```

10. Exit SQL*Plus.
11. Set the `COMPATIBLE` initialization parameter in the initialization parameter file to match the release to which you are downgrading.

    For example, if you are downgrading to release 8.1.5, then set the `COMPATIBLE` parameter to the following:

    ```
    COMPATIBLE = 8.1.5
    ```

12. Open the database to ensure that it is compatible with the release you specified with the `COMPATIBLE` parameter.

    If your database fails to open, then some incompatibilities still exist. If so, then reset the `COMPATIBLE` parameter to a higher setting. Remove the incompatibilities and attempt to reset database compatibility again. All incompatibilities with the database to which you are downgrading must be removed before you proceed with the downgrade process.

    See Also: "Remove Incompatibilities" on page 13-2 for information about removing incompatibilities.

Downgrade the Database

Make sure your database is compatible with the release to which you are downgrading before you perform the downgrade steps in this section. See "Remove Incompatibilities" on page 13-2 if you have not yet removed incompatibilities.

Complete the following steps to downgrade your release 9.0.1 database to a previous release:

1. Log in to the system as the owner of the Oracle home directory of the release from which you are downgrading.
2. Copy the following files from the *`ORACLE_HOME`*`/rdbms/admin` directory of the release from which you are downgrading to a directory outside of the Oracle home, such as the temporary directory on your system:
    - `utlip.sql`
    - `utlrp.sql`

If you have Java installed, copy one or more of the following files from the *ORACLE_HOME*/`javavm/install` directory of the release from which you are downgrading to a directory outside of the Oracle home, such as the temporary directory on your system:

- `jvmd815.sql` (if you are downgrading to release 8.1.5)
- `jvmd816.sql` (if you are downgrading to release 8.1.6)
- `jvmd817.sql`, `jisd817.sql`, `jspd817.sql` (if you are downgrading to release 8.1.7)

Also, if you have XDK for Java installed, copy one of the following files from the *ORACLE_HOME*/`xdk/admin` directory of the release from which you are downgrading to a directory outside of the Oracle home, such as the temporary directory on your system:

- `xmld817.sql` (if you are downgrading to release 8.1.7)

Make a note of the new location of these files. You may need them later in the downgrade process.

3. At a system prompt, change to the *ORACLE_HOME*/`rdbms/admin` directory.
4. Start SQL*Plus.
5. Connect to the database instance as a user with `SYSDBA` privileges.
6. Start up the instance in `RESTRICT` mode:

```
SQL> STARTUP RESTRICT
```

You may need to use the `PFILE` option to specify the location of your initialization parameter file.

7. Set the system to spool results to a log file for later verification of success:

```
SQL> SPOOL catoutd.log
```

If you want to see the complete detailed output of the script you will run, then you can also issue a `SET ECHO ON` statement:

```
SQL> SET ECHO ON
```

8. Run d*old_release*.sql, where *old_release* refers to the release to which you are downgrading. See Table 13–1 to choose the correct script. Each script provides a direct downgrade to the release specified in the "Downgrading To" column.

To run a script, enter the following:

```
SQL> @dold_release.sql
```

Table 13–1 Downgrade Scripts

Downgrading To	Run Script
8.1.7	`d0801070.sql`
8.1.6	`d0801060.sql`
8.1.5	`d0801050.sql`

Note: If the release to which you are downgrading is not included in Table 13–1, then see the README files in the new installation for the correct downgrade script to run.

The following are notes about running the script:

- You must use the version of the script included with the release from which you are downgrading.
- You must run the script in the environment of the release from which you are downgrading.
- You only need to run one script, even if your downgrade spans several releases. For example, if you are downgrading to release 8.1.5, then you need to run only `d0801050.sql`.

If you encounter any problems when you run the script, or any of the scripts in the remaining steps, then correct the causes of the problems and rerun the script. You can rerun any of the scripts described in this chapter as many times as necessary.

See Also: "Running Scripts" on page 1-11 for information about the types of errors to look for when you run a script.

9. Turn off the spooling of script results to the log file:

```
SQL> SPOOL OFF
```

Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 7; the suggested name was `catoutd.log`. Correct any problems you find in this file and rerun the appropriate downgrade script if necessary.

If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

```
SQL> SET ECHO OFF
```

10. Shut down the instance:

```
SQL> SHUTDOWN IMMEDIATE
```

If you are using Oracle9*i* Real Application Clusters, then shutdown all instances.

> **Note:** Oracle9*i* Real Application Clusters is a new, breakthrough architecture with scalability and high availability features that exceed the capabilities of previous Oracle cluster-enabled software releases.

11. Exit SQL*Plus.

12. If your operating system is UNIX, then change the following environment variables to point to the directories of the release to which you are downgrading:

- `ORACLE_HOME`
- `PATH`
- `ORA_NLS33`
- `LD_LIBRARY_PATH`

> **Note:** For Oracle9*i* Real Application Clusters, perform this step on all nodes.

> **See Also:** Your operating system-specific Oracle9*i* installation documents for information about setting other important environment variables on your operating system.

13. If your operating system is Windows, then complete the following steps:

    a. Stop all Oracle services, including the Oracle service OracleService*SID* of the database you are downgrading, where *SID* is the instance name.

       For example, if your *SID* is ORCL, then enter the following at an MS-DOS prompt:

       ```
       C:\> NET STOP OracleServiceORCL
       ```

       See Also: Your *Administrator's Guide* for Windows for information about stopping services.

    b. Delete the Oracle service at the MS-DOS command prompt by issuing the ORADIM command. For example, if your *SID* is ORCL, then enter the following MS-DOS command:

       ```
       C:\> ORADIM -DELETE -SID ORCL
       ```

14. Install the release to which you are downgrading using the installation media for that release.

    For example, if you are downgrading to release 8.1.5, then use the release 8.1.5 installation media to install the release 8.1.5 distribution of Oracle.

    You must install the release 8.1 software in a new Oracle home that is separate from the 9.0.1 release.

    Note: Installation is operating system-specific. For installation instructions, see your operating system-specific installation documentation and the README for your operating system.

15. If your operating system is Windows, then complete the following steps:

    a. Shut down and restart your computer.

    b. Create the Oracle database service at the MS-DOS command prompt using the ORADIM command.

       ```
       C:\> ORADIM -NEW -SID SID -INTPWD PASSWORD -MAXUSERS USERS
            -STARTMODE AUTO -PFILE ORACLE_HOME\DATABASE\INITSID.ORA
       ```

This syntax includes the following variables:

SID	is the same SID name as the SID of the release 8.1 database you are downgrading.
PASSWORD	is the password for the database instance. This is the password for the user connected with SYSDBA privileges. The -INTPWD option is not required. If you do not specify it, then operating system authentication is used, and no password is required.
USERS	is the maximum number of users who can be granted SYSDBA and SYSOPER privileges.
ORACLE_HOME	is the Oracle home directory of the database to which you are downgrading. Ensure that you specify the full pathname with the -PFILE option, including drive letter of the Oracle home directory.

For example, if you are downgrading to release 8.1.6, if your *SID* is ORCL, your *PASSWORD* is TWxy579, the maximum number of *USERS* is 10, and the *ORACLE_HOME* directory is `C:\ORANT`, then enter the following command:

```
C:\> ORADIM -NEW -SID ORCL -INTPWD TWxy579 -MAXUSERS 10
     -STARTMODE AUTO -PFILE C:\ORANT\DATABASE\INITORCL.ORA
```

16. If you are using a server parameter file to start up the instance, or if your initialization parameter file has an `SPFILE` (server parameter file) entry, then complete the following steps:

   a. Export the server parameter file to a traditional initialization parameter file:

   ```
   CREATE PFILE[=pfile-name] [FROM spfile-name];
   ```

   The initialization parameter file will be created as a text file. In an Oracle9*i* Real Application Clusters environment, it will contain all parameter settings of all instances.

   b. If you used the `SPFILE` parameter to specify a server parameter file, then change the `SPFILE` parameter to an `IFILE` parameter in the initialization parameter file used to start up the instance. Make sure the `IFILE` parameter points to the initialization parameter file that you exported from the server parameter file.

  c. If you are using Oracle9*i* Real Application Clusters, then create instance-specific initialization parameter files. Remove all instance-specific parameters from the initialization parameter file that you exported from the server parameter file.

     You can use the `IFILE` parameter in each instance-specific parameter file to point to the initialization parameter file that you exported from the server parameter file.

17. Copy configuration files to a location outside of the Oracle home of the release from which you are downgrading:

  a. If your initialization parameter file resides within the Oracle home of the release from which you are downgrading, then copy the initialization parameter file to a location outside of the Oracle home. By default Oracle looks for the initialization parameter file in *`ORACLE_HOME/`*`dbs` on UNIX and *`ORACLE_HOME`*`\database` on Windows platforms. The initialization parameter file can reside anywhere you wish, but it should not reside in the Oracle home of the release from which you are downgrading.

  b. If your initialization parameter file has an `IFILE` (include file) entry and the file specified in the `IFILE` entry resides within the Oracle home of the release from which you are downgrading, then copy the file specified by the `IFILE` entry to a location outside of the Oracle home. The file specified in the `IFILE` entry has additional initialization parameters. After you copy this file, edit the initialization parameter file to point to its new location.

  c. If you have a password file that resides within the Oracle home of the release from which you are downgrading, then move or copy the password file to the Oracle home of the release to which you are downgrading. The name and location of the password file are operating system-specific. On UNIX operating systems, the default password file is *`ORACLE_HOME/`*`dbs/orapw`*`sid`*. On Windows operating systems, the default password file is *`ORACLE_HOME`*`\database\pwd`*`sid`*`.ora`. On both UNIX and Windows operating systems, *sid* is your Oracle instance ID.

Note: For Oracle9*i* Real Application Clusters, perform this step on all nodes. Also, set the `PARALLEL_SERVER` initialization parameter to `false`. You can change it back to `true` after the downgrade operation is complete.

---

18. Copy the following files into the *ORACLE_HOME/*`rdbms/admin` directory of the release to which you are downgrading:

    - `utlip.sql`
    - `utlrp.sql`

    If you have Java installed, copy one or more of the following files to the *ORACLE_HOME/*`javavm/install` directory of the release to which you are downgrading:

    - `jvmd815.sql` (if you are downgrading to release 8.1.5)
    - `jvmd816.sql` (if you are downgrading to release 8.1.6)
    - `jvmd817.sql`, `jisd817.sql`, `jspd817.sql` (if you are downgrading to release 8.1.7)

    Also, if you have XDK for Java installed, copy one of the following files to the *ORACLE_HOME/*`xdk/admin` directory of the release to which you are downgrading:

    - `xmld817.sql` (if you are downgrading to release 8.1.7)

    You copied these files to a directory outside of the Oracle home of the release from which you are downgrading in Step 2.

19. Add the following initialization parameter to your initialization parameter file:

    ```
    _SYSTEM_TRIG_ENABLED = false
    ```

    This initialization parameter should be removed from your initialization parameter file after the downgrade is complete.

20. At a system prompt, change to the *ORACLE_HOME/*`rdbms/admin` directory of the release to which you are downgrading.

21. If you are downgrading to an 8.1 release, start Server Manager. Otherwise, start SQL*Plus.

22. Connect to the database instance as a user with `SYSDBA` privileges.

23. Start up the instance in `RESTRICT` mode:

    ```
    STARTUP RESTRICT
    ```

    You may need to use the `PFILE` option to specify the location of your initialization parameter file.

24. Set the system to spool results to a log file for later verification of success:

    ```
    SPOOL catoutd2.log
    ```

    If you want to see the complete detailed output of the script you will run, then you can also issue a `SET ECHO ON` statement:

    ```
    SET ECHO ON
    ```

25. Run `utlip.sql`:

    ```
    @utlip.sql
    ```

    The `utlip.sql` script invalidates all existing PL/SQL modules by altering certain dictionary tables so that subsequent recompilations will happen in the format required by the database. It also reloads packages `STANDARD` and `DBMS_STANDARD`, which are necessary for any PL/SQL compilations.

    > **See Also:** "Changing Word-Size" on page 1-12 for more information about changing word-size.

26. Run `catalog.sql`:

    ```
    @catalog.sql
    ```

27. Run `catproc.sql`:

    ```
    @catproc.sql
    ```

28. Run `catrep.sql`:

    ```
    @catrep.sql
    ```

29. If the Oracle system has Oracle Parallel Server installed, then run the following catalog script supplied with the release to which you downgraded:

    ```
    @catparr.sql
    ```

30. If the Oracle system has Java installed, then run the appropriate downgrade scripts (copied to this directory in Step 18) to downgrade the Java component. When you run any of the scripts, replace *ORACLE_HOME* with the full path of the Oracle home directory of the release to which you downgraded.

    If you are downgrading to release 8.1.5, then run the following script:

    ```
    @ORACLE_HOME/javavm/install/jvmd815.sql
    ```

If you are downgrading to release 8.1.6, then run the following script:

```
@ORACLE_HOME/javavm/install/jvmd816.sql
```

If you are downgrading to release 8.1.7, then run the following scripts:

```
@ORACLE_HOME/javavm/install/jvmd817.sql
@ORACLE_HOME/xdk/admin/xmld817.sql
@ORACLE_HOME/javavm/install/jisd817.sql
@ORACLE_HOME/javavm/install/jspd817.sql
```

31. Run `utlrp.sql`. This step is optional and can be done regardless of whether there was a change in word-size.

    ```
    @utlrp.sql
    ```

    The `utlrp.sql` script recompiles all existing PL/SQL modules that were previously in an `INVALID` state, such as packages, procedures, types, and so on. These actions are optional; however, they ensure that the cost of recompilation is incurred during installation rather than in the future.

    Oracle Corporation highly recommends running `utlrp.sql`.

    If you receive compile-time errors while running `utlrp.sql`, you may need to run one or more catalog scripts supplied with the release to which you downgraded. For example, to re-create Heterogeneous Services data dictionary views, tables, and packages, run `caths.sql`:

    ```
    @caths.sql
    ```

32. Turn off the spooling of script results to the log file:

    ```
    SPOOL OFF
    ```

    Then, check the spool file and verify that the packages and procedures compiled successfully. You named the spool file in Step 24; the suggested name was `catoutd2.log`. Correct any problems you find in this file and rerun the appropriate script if necessary.

    If you specified `SET ECHO ON`, then you may want to `SET ECHO OFF` now:

    ```
    SET ECHO OFF
    ```

33. Shut down the instance:

    ```
    SHUTDOWN IMMEDIATE
    ```

    > **Note:** For Oracle Parallel Server, set the `PARALLEL_SERVER` initialization parameter to `false`. You can change it back to `true` after the downgrade operation is complete.

34. Exit Server Manager or SQL*Plus, depending on which you used to complete the downgrade.
35. Remove the following initialization parameter from your initialization parameter file:

    ```
    _SYSTEM_TRIG_ENABLED = false
    ```

    You added this initialization parameter to your initialization parameter file in Step 19.

Your database is now downgraded. Complete the procedures described in the following sections to finish downgrading specific components.

Regenerating Replication Support

If you are using Oracle Replication, then complete the actions described below based on whether the downgraded database is a master site or a materialized view site.

Master Site

If the downgraded database is a master site for one or more object groups, then complete the following steps to regenerate replication support:

1. Quiesce each object group.
2. Generate replication support for each replicated table in the group.
3. Resume master activity for the object group. If the masterdef site is at release 8.1, then make sure you specify GENERATE_80_COMPATIBLE=>TRUE in the GENERATE_REPLICATION_SUPPORT calls.

Materialized View Site

If the downgraded database is a materialized view site, then generate replication support for each updatable materialized view.

> **See Also:** *Oracle9i Replication* for more information about generating replication support

14

Removing Incompatibilities Before Downgrading to Release 8.1

This chapter provides step-by-step instructions on removing incompatibilities before downgrading to release 8.1. This chapter covers the following topics:

- Identifying Incompatibilities
- Removing Incompatibilities Before Downgrading to Release 8.1.6 or Release 8.1.7
- Removing Incompatibilities Before Downgrading to Release 8.1.5
- Continue with Your Downgrade to Release 8.1

See Also: Some aspects of downgrading are operating system-specific. See your operating system-specific Oracle documentation for additional operating system-specific instructions about downgrading.

Identifying Incompatibilities

To identify any incompatibilities that may exist with the release to which you are downgrading, perform the following steps:

1. Log in to the system as the owner of the Oracle home directory.
2. At a system prompt, change to the *ORACLE_HOME*/rdbms/admin directory.
3. Start SQL*Plus.
4. Connect to the database instance as a user with SYSDBA privileges.
5. Query the V$COMPATIBILITY dynamic performance view to identify the incompatibilities:

   ```
   SQL> SELECT * FROM v$compatibility WHERE release != ’0.0.0.0.0’;
   ```

   An incompatibility exists wherever the value in the RELEASE column is higher than the release to which you are downgrading.

   > **Note:** This query does not show features with a compatibility level of 0.0.0.0.0. These features currently are not in use, and no action is required for them.

6. Run utlincmp.sql:

   ```
   SQL> SPOOL utlincmp.out
   SQL> @utlincmp.sql
   SQL> SPOOL OFF
   ```

   The utlincmp.sql script runs all of the queries described in the rest of this chapter to identify incompatibilities. Therefore, you can perform all of the SELECT statements described in the rest of this chapter simply by running the utlincmp.sql script.

   After the utlincmp.sql script runs, view the utlincmp.out file and look for instances where a SELECT statement returned values. The values returned are incompatibilities with the previous release.

7. Drop or change all incompatibilities to make your database compatible with the release to which you are downgrading.

The following sections provide detailed information about removing incompatibilities with previous releases of Oracle. Depending on the release to which you are downgrading, you may need to read some or all of the following sections.

To remove incompatibilities, you may need to complete actions that require the privileges of user SYS. Therefore, you should log in as user SYS and connect with SYSDBA privileges to perform the actions described in the following sections, unless instructed otherwise.

Note: If you are downgrading from Oracle9*i* Enterprise Edition to Oracle9*i* (formerly Workgroup Server), then, before you downgrade, modify any applications that use the advanced features of Oracle9*i* Enterprise Edition so that they do not use these advanced features. See *Oracle9i Database New Features* for more information about the differences between the editions.

Removing Incompatibilities Before Downgrading to Release 8.1.6 or Release 8.1.7

If you are downgrading to release 8.1.6 or release 8.1.7, then complete the actions in the following sections to remove incompatibilities:

- Tablespaces
- Schema Objects
- Partitioning
- Datatypes
- User-Defined Datatypes
- SQL and PL/SQL
- Constraints and Triggers

Tablespaces

This section describes removing incompatibilities relating to tablespaces.

Discontinue Use of Automatic Segment-Space Managed Tablespaces

Before you downgrade to release 8.1.7 or lower, discontinue use of all automatic segment-space managed tablespaces. To identify existing automatic segment-space managed tablespaces, issue the following SQL statement:

```
SELECT tablespace_name FROM dba_tablespaces
    WHERE segment_space_management = 'AUTO';
```

These tablespaces must be dropped before downgrading.

Discontinue Use of Automatic Undo Managed Tablespaces

Before you downgrade to release 8.1.7 or lower, discontinue use of all automatic undo managed tablespaces. To identify existing automatic undo managed tablespaces, issue the following SQL statement:

```
SELECT tablespace_name FROM dba_tablespaces
    WHERE contents = 'UNDO';
```

These tablespaces must be dropped before downgrading.

Schema Objects

This section describes removing incompatibilities relating to schema objects.

Drop External Tables

Before you downgrade to release 8.1.7 or lower, drop all external tables. To identify existing external tables, issue the following SQL statement:

```
SELECT o.name AS TABLE_NAME, u.name AS TABLE_OWNER
    FROM sys.user$ u, sys.obj$ o, sys.tab$ t
    WHERE t.obj# = o.obj# AND o.owner# = u.user#
        AND BITAND(t.property, 2147483648) != 0;
```

Drop all tables listed.

Drop All Bitmap Secondary Indexes on Index-Organized Tables

Before you downgrade to release 8.1.7 or lower, drop all bitmap secondary indexes on non-partitioned and partitioned index organized tables in your database. To identify existing bitmap secondary indexes on index-organized tables, issue the following SQL statement:

```
SELECT index_name, i.owner, t.table_name
    FROM dba_indexes i, dba_tables t
    WHERE i.index_type = 'BITMAP' AND i.table_name = t.table_name
        AND t.owner = i.table_owner AND t.iot_type = 'IOT';
```

Rebuild Index-Organized Tables without Mapping Tables

Before you downgrade to release 8.1.7 or lower, after dropping all bitmap secondary indexes on non-partitioned and partitioned index-organized tables, you need to rebuild the corresponding index-organized tables without mapping tables.

To identify index-organized tables with mapping tables, issue the following SQL statement:

```
SELECT owner, iot_name
    FROM dba_tables
    WHERE iot_type = ’IOT_MAPPING’;
```

For each of the tables (for example iot), you can rebuild without mapping tables as follows:

```
ALTER TABLE iot MOVE NOMAPPING;
```

Drop All B-Tree Indexes on UROWID Datatypes on Heap and Index-Organized Tables

Before you downgrade to release 8.1.7 or lower, drop all B-tree indexes on heap and index organized tables. To identify such B-tree indexes, issue the following SQL statement:

```
SELECT index_owner, index_name FROM dba_ind_columns ic, dba_tab_columns tc
    WHERE tc.data_type = ’UROWID’ AND tc.table_name = ic.table_name
        AND tc.column_name = ic.column_name;
```

Remove Indexes With Large Keys

Before downgrading to release 8.1.7 or lower, remove Any index with large keys. To identify such indexes, issue the following SQL statement:

```
SELECT u.name, o.name, i.flags
    FROM sys.obj$ o, sys.user$ u, sys.ind$ i
    WHERE u.user# = o.owner#
        AND o.obj# = i.obj#
        AND BITAND(i.flags, 16384) != 0;
```

Drop any indexes identified by this statement.

Partitioning

This section describes disabling release 9.0.1 partitioning features.

Discontinue Use of Hash Partitioned Index-Organized Tables

Before you downgrade to release 8.1.7 or lower, discontinue use of all hash partitioned index-organized tables. To identify existing hash partitioned index-organized tables, issue the following SQL statement:

```
SELECT t.owner, t.table_name
    FROM dba_tables t, dba_part_tables p
    WHERE t.table_name = p.table_name AND t.owner = p.owner
        AND t.iot_type = 'IOT' AND t.partitioned = 'YES'
        AND p.partitioning_type = 'HASH';
```

If you do not need to preserve the table data, then simply drop the tables. However, if you need to preserve the table data, you can do it in one of the following ways:

- Move the table data into a range partitioned index-organized table or non-partitioned index-organized table using the `CREATE TABLE ... AS SELECT` statement.
- Export the table using the Oracle9*i* Export utility. The data can then be loaded into a non-partitioned index-organized table or a range-partitioned index-organized table using the release 8.1 Import utility.

Datatypes

This section describes disabling datatypes that are available only in release 9.0.1 and higher.

Discontinue Use of Datetime and Interval Datatypes

Before you downgrade to release 8.1.7 or lower, the following datetime and interval datatypes have to be dropped:

- `TIMESTAMP`
- `TIMESTAMP WITH TIME ZONE`
- `TIMESTAMP WITH LOCAL TIME ZONE`
- `INTERVAL YEAR TO MONTH`
- `INTERVAL DAY TO SECOND`

However, when the datatype is TIMESTAMP WITH LOCAL TIME ZONE, the TIMESTAMP WITH LOCAL TIME ZONE columns can be converted to DATE columns by explicitly issuing an ALTER TABLE statement.

The ALTER TABLE statement scans all rows of the table. If the TIMESTAMP WITH LOCAL TIME ZONE data has fractional seconds, the row data for the column will be updated by rounding the fractional seconds; if the TIMESTAMP WITH LOCAL TIME ZONE data has the minute field greater than or equal to 60, the row data for the column will be updated by subtracting 60 from its minute field. When modifying a TIMESTAMP WITH LOCAL TIME ZONE column to a DATE column, the information for fractional seconds and time zone adjustment will be lost.

Downgrading will fail if any of the following objects exist in the database:

- Any table containing columns of these new datatypes
- Any procedure or function (stand alone or inside a package) declared with arguments or a result of these new datatypes
- Any object type with attributes of these new datatypes, or member functions with arguments or a result of these new datatypes
- Any collection type whose element type is one of these new datatypes

These objects have to be dropped in order to downgrade to a previous release.

To list tables with columns of type TIMESTAMP, issue the following SQL statement:

```
SELECT owner, table_name, column_name
    FROM dba_tab_columns
    WHERE data_type LIKE 'TIMESTAMP(%)';
```

For each table listed as a result of this statement, drop its TIMESTAMP datatype columns, or drop the whole table.

To list tables with columns of type TIMESTAMP WITH TIME ZONE, issue the following SQL statement:

```
SELECT owner, table_name, column_name
    FROM dba_tab_columns
    WHERE data_type LIKE 'TIMESTAMP(%) WITH TIME ZONE';
```

For each table listed as a result of this statement, drop its TIMESTAMP WITH TIME ZONE datatype columns, or drop the whole table.

To list tables with columns of type TIMESTAMP WITH LOCAL TIME ZONE, issue the following SQL statement:

```
SELECT owner, table_name, column_name
    FROM dba_tab_columns
    WHERE data_type LIKE 'TIMESTAMP(%) WITH LOCAL TIME ZONE';
```

For each table listed as a result of this statement, drop its `TIMESTAMP WITH LOCAL TIME ZONE` datatype columns, or drop the whole table.

To list tables with columns of type `INTERVAL YEAR TO MONTH`, issue the following SQL statement:

```
SELECT owner, table_name, column_name
    FROM dba_tab_columns
    WHERE data_type LIKE 'INTERVAL YEAR(%) TO MONTH';
```

For each table listed as a result of this statement, drop its `INTERVAL YEAR TO MONTH` datatype columns, or drop the whole table.

To list tables with columns of type `INTERVAL DAY TO SECOND`, issue the following SQL statement:

```
SELECT owner, table_name, column_name
    FROM dba_tab_columns
    WHERE data_type LIKE 'INTERVAL DAY(%) TO SECOND';
```

For each table listed as a result of this statement, drop its `INTERVAL DAY TO SECOND` datatype columns, or drop the whole table.

To find a list of procedures and functions declared with arguments or a result of type `TIMESTAMP`, issue the following SQL statement:

```
SELECT owner, object_name, package_name, argument_name
    FROM all_arguments
    WHERE data_type = 'TIMESTAMP';
```

To find a list of procedures and functions declared with arguments or a result of type `TIMESTAMP WITH TIME ZONE`, issue the following SQL statement:

```
SELECT owner, object_name, package_name, argument_name
    FROM all_arguments
    WHERE data_type = 'TIMESTAMP WITH TIME ZONE';
```

To find a list of procedures and functions declared with arguments or a result of type `TIMESTAMP WITH LOCAL TIME ZONE`, issue the following SQL statement:

```
SELECT owner, object_name, package_name, argument_name
    FROM all_arguments
    WHERE data_type = 'TIMESTAMP WITH LOCAL TIME ZONE';
```

To find a list of procedures and functions declared with arguments or a result of type `INTERVAL YEAR TO MONTH`, issue the following SQL statement:

```
SELECT owner, object_name, package_name, argument_name
    FROM all_arguments
    WHERE data_type = 'INTERVAL YEAR TO MONTH';
```

To find a list of procedures and functions declared with arguments or a result of type `INTERVAL DAY TO SECOND`, issue the following SQL statement:

```
SELECT owner, object_name, package_name, argument_name
    FROM all_arguments
    WHERE data_type = 'INTERVAL DAY TO SECOND';
```

To find a list of object types with attributes of type `TIMESTAMP`, or member functions with arguments or a result of type `TIMESTAMP`, issue the following SQL statement:

```
SELECT owner, type_name, attr_name
    FROM dba_type_attrs
    WHERE attr_type_name = 'TIMESTAMP';

SELECT owner, type_name, method_name, param_name
    FROM dba_method_params
    WHERE param_type_name = 'TIMESTAMP';

SELECT owner, type_name, method_name
    FROM dba_method_results
    WHERE result_type_name = 'TIMESTAMP';
```

To find a list of object types with attributes of type `TIMESTAMP WITH TIME ZONE`, or member functions with arguments or a result of type `TIMESTAMP WITH TIME ZONE`, issue the following SQL statement:

```
SELECT owner, type_name, attr_name
    FROM dba_type_attrs
    WHERE attr_type_name = 'TIMESTAMP WITH TIME ZONE';

SELECT owner, type_name, method_name, param_name
    FROM dba_method_params
    WHERE param_type_name = 'TIMESTAMP WITH TIME ZONE';

SELECT owner, type_name, method_name
    FROM dba_method_results
    WHERE result_type_name = 'TIMESTAMP WITH TIME ZONE';
```

To find a list of object types with attributes of type `TIMESTAMP WITH LOCAL TIME ZONE`, or member functions with arguments or a result of type `TIMESTAMP WITH LOCAL TIME ZONE`, issue the following SQL statement:

```
SELECT owner, type_name, attr_name
    FROM dba_type_attrs
    WHERE attr_type_name = 'TIMESTAMP WITH LOCAL TIME ZONE';

SELECT owner, type_name, method_name, param_name
    FROM dba_method_params
    WHERE param_type_name = 'TIMESTAMP WITH LOCAL TIME ZONE';

SELECT owner, type_name, method_name
    FROM dba_method_results
    WHERE result_type_name = 'TIMESTAMP WITH LOCAL TIME ZONE';
```

To find a list of object types with attributes of type `INTERVAL YEAR TO MONTH`, or member functions with arguments or a result of type `INTERVAL YEAR TO MONTH`, issue the following SQL statement:

```
SELECT owner, type_name, attr_name
    FROM dba_type_attrs
    WHERE attr_type_name = 'INTERVAL YEAR TO MONTH';

SELECT owner, type_name, method_name, param_name
    FROM dba_method_params
    WHERE param_type_name = 'INTERVAL YEAR TO MONTH';

SELECT owner, type_name, method_name
    FROM dba_method_results
    WHERE result_type_name = 'INTERVAL YEAR TO MONTH';
```

To find a list of object types with attributes of type `INTERVAL DAY TO SECOND`, or member functions with arguments or a result of type `INTERVAL DAY TO SECOND`, issue the following SQL statement:

```
SELECT owner, type_name, attr_name
    FROM dba_type_attrs
    WHERE attr_type_name = 'INTERVAL DAY TO SECOND';

SELECT owner, type_name, method_name, param_name
    FROM dba_method_params
    WHERE param_type_name = 'INTERVAL DAY TO SECOND';

SELECT owner, type_name, method_name
```

```
    FROM dba_method_results
    WHERE result_type_name = 'INTERVAL DAY TO SECOND';
```

To find a list of collection types with elements of type `TIMESTAMP`, issue the following SQL statement:

```
SELECT owner, type_name, coll_type
    FROM dba_coll_types
    WHERE elem_type_name = 'TIMESTAMP';
```

To find a list of collection types with elements of type `TIMESTAMP WITH TIME ZONE`, issue the following SQL statement:

```
SELECT owner, type_name, coll_type
    FROM dba_coll_types
    WHERE elem_type_name = 'TIMESTAMP WITH TIME ZONE';
```

To find a list of collection types with elements of type `TIMESTAMP WITH LOCAL TIME ZONE`, issue the following SQL statement:

```
SELECT owner, type_name, coll_type
    FROM dba_coll_types
    WHERE elem_type_name = 'TIMESTAMP WITH LOCAL TIME ZONE';
```

To find a list of collection types with elements of type `INTERVAL YEAR TO MONTH`, issue the following SQL statement:

```
SELECT owner, type_name, coll_type
    FROM dba_coll_types
    WHERE elem_type_name = 'INTERVAL YEAR TO MONTH';
```

To find a list of collection types with elements of type `INTERVAL DAY TO SECOND`, issue the following SQL statement:

```
SELECT owner, type_name, coll_type
    FROM dba_coll_types
    WHERE elem_type_name = 'INTERVAL DAY TO SECOND';
```

Discontinue Use of LOB Columns in Partitioned Index-Organized Tables

Before you downgrade to release 8.1.7 or lower, discontinue use of all `LOB` columns in partitioned index-organized tables. To identify existing partitioned index-organized tables with `LOB` columns, issue the following SQL statement:

```
SELECT column_name, t.owner, t.table_name
    FROM dba_lobs l, dba_tables t
    WHERE l.table_name = t.table_name AND l.owner = t.owner
        AND t.iot_type = 'IOT' AND t.partitioned = 'YES';
```

If you do not need to preserve the LOB columns and their data, simply drop the columns. However, if you need to preserve the LOB columns, you can create corresponding non-partitioned index-organized tables. For example, issue the following SQL statement to create non-partitioned index-organized tables corresponding to the tables listed in the previous statement:

```
CREATE lob_iot (c1 primary key, c2) AS SELECT * FROM lob_piot;
```

User-Defined Datatypes

This section describes disabling features related to user-defined datatypes that are only available in release 9.0.1 and higher.

Drop User-Defined Aggregate Functions

Before you downgrade to release 8.1.7 or lower, drop all user-defined aggregate functions. To identify existing user-defined aggregate functions, issue the following SQL statement:

```
SELECT procedure_name FROM dba_procedures
    WHERE aggregate = 'YES';
```

Drop all aggregate functions listed.

Remove All Evolved Types and Their Dependent Types and Tables

Before you downgrade to release 8.1.7 or lower, all evolved types and their dependent types and tables must be removed. To identify all evolved types, issue the following SQL statement:

```
SELECT UNIQUE owner, type_name
    FROM dba_types
    WHERE version_name != '$8.0';
```

To identify all tables that reference an evolved type, issue the following SQL statement:

```
SELECT UNIQUE owner, table_name
    FROM dba_tab_columns
    WHERE data_type_owner IS NOT NULL
        AND version_name != '$8.0';
```

Discontinue Use of Subtypes and Non-Final Types

Before you downgrade to release 8.1.7 or lower, discontinue use of all subtypes and non-final types in tables. To identify the use of existing subtypes and non-final types in tables, issue the following SQL statement:

```
SELECT c.name AS COLUMN_NAME, o.name AS TABLE_NAME, u.name AS TABLE_OWNER
    FROM user$ u, sys.obj$ o, sys.col$ c, sys.coltype$ ct, sys.type$ t
    WHERE u.user# = o.owner# AND o.obj# = c.obj# AND c.obj# = ct.obj#
        AND c.intcol# = ct.intcol# and ct.toid = t.toid AND o.type# = 2
        AND BITAND(t.properties, 3153928) > 0;
```

Discontinue Use of Varray Columns in Partitioned Index-Organized Tables

Before you downgrade to release 8.1.7 or lower, discontinue use of all varray columns in partitioned index-organized tables. To identify existing partitioned index-organized tables with varrays, issue the following SQL statement:

```
SELECT v.parent_table_name, t.owner, t.table_name
    FROM dba_varrays v, dba_tables t
    WHERE v.parent_table_name = t.table_name AND v.owner = t.owner
        AND t.iot_type = 'IOT' AND t.partitioned = 'YES';
```

If you do not need to preserve the varray columns and their data, simply drop the columns. However, if you need to preserve the varray columns, you can create corresponding non-partitioned index-organized tables. For example, issue the following SQL statement to create non-partitioned index-organized tables corresponding to the tables listed in the previous statement:

```
CREATE lob_iot (c1 primary key, c2) AS SELECT * FROM varray_piot;
```

SQL and PL/SQL

The following sections describe specific SQL and PL/SQL downgrading issues. The actions described in these sections help you to avoid compile and runtime errors in SQL scripts and stored procedures. Although these actions are not strictly required, Oracle Corporation recommends that you perform them before you downgrade.

Discontinue Use of Pipelined Table Functions

Before you downgrade to release 8.1.7 or lower, discontinue use of all pipelined table functions. To identify existing pipelined table functions, issue the following SQL statement:

```
SELECT procedure_name FROM dba_procedures
    WHERE pipelined = 'YES';
```

Discontinue Use of Parallel Table Functions

Before you downgrade to release 8.1.7 or lower, discontinue use of all parallel table functions. To identify existing parallel table functions, issue the following SQL statement:

```
SELECT procedure_name FROM dba_procedures
    WHERE parallel = 'YES';
```

Constraints and Triggers

This section describes removing incompatibilities relating to constraints and triggers.

Drop All View Constraints

Before you downgrade to release 8.1.7 or lower, drop all view related primary key, unqiue, and foreign key constraints. To identify existing view constraints, issue the following SQL statement:

```
SELECT * FROM dba_constraints WHERE view_related = 'DEPEND_ON_VIEW';
```

Removing Incompatibilities Before Downgrading to Release 8.1.5

If you are downgrading to release 8.1.5, then complete the actions in the following sections to remove incompatibilities:

- Tablespaces
- Datatypes
- Constraints and Triggers
- Security
- Advanced Queuing (AQ)

Tablespaces

This section describes removing incompatibilities relating to tablespaces.

Drop or Convert Migrated Tablespaces

Release 8.1.6 and higher supports tablespace migration, which allows tablespaces to be migrated from dictionary managed format to locally managed format and vice versa. When a tablespace is migrated from dictionary managed to locally managed format, the tablespace is marked as a ’migrated tablespace’ under certain conditions. You cannot downgrade to a previous release of Oracle if your database has such tablespaces.

To identify incompatible migrated tablespaces, enter the following SQL statement:

```
SELECT DISTINCT(tablespace_name) FROM dba_segments
    WHERE segment_type = ’SPACE HEADER’;
```

You can either convert the tablespaces listed to dictionary managed tablespaces, or you can drop them. To convert migrated tablespaces listed to dictionary managed tablespaces, execute the DBMS_SPACE_ADMIN.TABLESPACE_MIGRATE_FROM_LOCAL procedure. For example, if a tablespace named TS1 is listed, execute the following procedure:

```
EXECUTE dbms_space_admin.tablespace_migrate_from_local(’TS1’);
```

To drop them, issue a `DROP TABLESPACE` statement. For example, if a tablespace named TS1 is listed, issue the following statement to drop the tablespace:

```
DROP TABLESPACE TS1
    INCLUDING CONTENTS
        CASCADE CONSTRAINTS;
```

Correct Transient Segments

During migration of tablespaces from dictionary managed format to locally managed format (and vice versa), the segments in the tablespace are temporarily put in a transient state. You cannot downgrade a database with these transient segments.

To identify transient segments, enter the following SQL statement:

```
SELECT DISTINCT(tablespace_name) FROM sys_dba_segs
    WHERE DECODE(BITAND(segment_flags, 16), 16, 1, 0) = 1;
```

To correct the transient segments listed, execute the `DBMS_SPACE_ADMIN.TABLESPACE_FIX_SEGMENT_STATES` procedure on the tablespace. For example, if a tablespace named TS1 is listed, execute the following procedure:

```
EXECUTE dbms_space_admin.tablespace_fix_segment_states(’TS1’);
```

Dropping Segments in Optimized Locally Managed Tablespaces

In release 8.1.6 and higher, dropping segments in locally managed tablespaces is optimized by storing some additional information in the data dictionary. You cannot downgrade until all of the temporary segments in the locally managed tablespaces are dropped. You can check for these segments either by querying the V$COMPATSEG view or by attempting to start up the database with a COMPATIBLE initialization parameter setting lower than 8.1.6.

To query the V$COMPATSEG view, issue the following SQL statement:

```
SELECT * FROM v$compatseg;
```

The incompatible segments have FASTDROP in the TYPE_ID column and a value of 8.1.6.0.0 in the RELEASE column.

If you start up the database with a COMPATIBLE setting lower than 8.1.6 and there are incompatible segments, startup fails and error messages similar to the following are displayed:

```
ORA-00402: database changes by release 8.1.6.0.0 cannot be used by release 8.1.5.0.0
ORA-00405: compatibility type "Faster segment drop"
```

When you reset database compatibility later in the downgrade process, Oracle will attempt to remove the temporary segments. If these segments are not removed the first time you reset compatibility, repeat the procedure to reset database compatibility.

> **See Also:** "Reset Database Compatibility" on page 13-3 for instructions on resetting database compatibility.

Datatypes

This section describes disabling datatypes that are available only in release 8.1.6 and higher.

Discontinue Use of CACHE READS Specification for LOBs

Before you downgrade to release 8.1.5, you must stop using the CACHE READS storage parameter for LOBs.

The following sections contain SQL statements that identify existing uses of CACHE READS specification for LOBs.

To identify tables that have LOB columns specified as cache reads, issue the following SQL statement as user SYS:

```
SELECT owner, table_name, column_name
    FROM dba_lobs
    WHERE cache = 'CACHEREADS';
```

To identify partitioned tables that have LOB columns specified as CACHE READS as default attributes at the table level, issue the following SQL statement as user SYS:

```
SELECT table_owner, table_name, column_name
  FROM dba_part_lobs
  WHERE def_cache = 'CACHEREADS';
```

To identify partitioned tables that have LOB columns specified as CACHE READS at the partition level, issue the following SQL statement as user SYS:

```
SELECT table_owner, table_name, column_name, partition_name
    FROM dba_lob_partitions
    WHERE cache = 'CACHEREADS';
```

To identify partitioned tables that have LOB columns specified as CACHE READS at the subpartition level, issue the following SQL statement as user SYS:

```
SELECT table_owner, table_name, column_name, subpartition_name
    FROM dba_lob_subpartitions
    WHERE cache = 'CACHEREADS';
```

After you have identified all of the uses of the CACHE READS specification for LOBs, you can change them to use the CACHE or NOCACHE specification, because these specifications are compatible with release 8.1.5. The following sections provide examples of the SQL statements you must issue to make these changes.

To change a LOB storage parameter from CACHE READS to CACHE or NOCACHE, use the ALTER TABLE ... MODIFY LOB statement. For example, on a table named lob_tab with a LOB column named lob_col, issue the following SQL statement to change the storage specification to CACHE:

```
ALTER TABLE lob_tab MODIFY LOB (lob_col) (CACHE);
```

This statement can also be used to modify LOB storage parameters at both the table and partition level for partitioned tables. To modify only the default table level attributes of LOB columns from CACHE READS to CACHE or NOCACHE for partitioned tables, use the ALTER TABLE ... MODIFY DEFAULT ATTRIBUTES statement. For example, on a table named lob_part_tab with a LOB column named lob_col, issue the following SQL statement to change the storage specification to NOCACHE:

```
ALTER TABLE lob_part_tab
```

```
    MODIFY DEFAULT ATTRIBUTES LOB (lob_col) (NOCACHE);
```

To modify LOB storage parameter from CACHE READS to CACHE or NOCACHE at the partition level for partitioned tables, use the ALTER TABLE ... MODIFY PARTITION statement. For example, on a table named lob_part_tab with a LOB column named lob_col and a partition named part_1, issue the following SQL statement to change the storage specification to CACHE:

```
ALTER TABLE lob_part_tab
    MODIFY PARTITION part_1 LOB (lob_col) (CACHE);
```

Constraints and Triggers

This section describes removing incompatibilities relating to constraints and triggers.

Remove Incompatible Triggers

A new feature in release 8.1.6 and higher supports triggers on all SQL DDL statements, instead of only CREATE, ALTER, and DROP statements. These triggers must be dropped before downgrading. To check for triggers that are incompatible with release 8.1.5, connect with SYSDBA privileges and issue the following SQL statement:

```
SELECT owner, trigger_name, triggering_event
    FROM dba_triggers
   WHERE base_object_type LIKE '%DATABASE%' OR base_object_type LIKE '%SCHEMA%';
```

Drop all of the triggers that have one of the following events in the TRIGGERING_EVENT column:

- ANALYZE
- ASSOCIATE STATISTICS
- AUDIT
- COMMENT
- DDL
- DISASSOCIATE STATISTICS
- GRANT
- NOAUDIT

- `RENAME`
- `REVOKE`
- `TRUNCATE`

Security

This section describes removing incompatibilities relating to database security.

Removing or Recreating Global Users Whose External Name Is NULL

The schema-independent user feature of Oracle Advanced Security, in which many enterprise users access a shared schema, is not compatible with release 8.1.5. If you are using this feature, you need to identify the shared schemas. That is, the global users whose external name is NULL. To identify the shared schemas, issue the following SQL statement:

```
SELECT username FROM dba_users
   WHERE password = 'GLOBAL' AND external_name IS NULL;
```

You then need to either remove the users listed or recreate them so that they have a non-NULL external name.

Also, any enterprise users who access the shared schema need to be created as database users so that they can now access database objects. These enterprise users did not previously exist in the database. You can create them as global users (that is, authenticated by SSL), as externally authenticated users, or as users authenticated by password.

For example, suppose you created the following global user in release 8.1.6 or higher:

```
CONNECT system/system_password
CREATE USER user1 IDENTIFIED GLOBALLY AS '';
```

To make this user compatible with release 8.1.5 by authenticating the user with a password, issue the following SQL statements:

```
CONNECT system/system_password
ALTER USER user1 IDENTIFIED BY welcome;
```

Substitute your `SYSTEM` user password to connect.

Advanced Queuing (AQ)

Complete the following tasks to disable release 8.1.6 and higher AQ features in your queue tables:

Drop Queue Tables Containing Special User-Defined Types

Release 8.1.6 and higher creates JMS types for use in queue tables. These types are dropped automatically during downgrade. So, before downgrading, you must drop the queue tables containing these JMS types. To identify these queue tables, issue the following SQL statement:

```
SELECT owner, queue_table, object_type FROM all_queue_tables
    WHERE object_type LIKE 'SYS.AQ$_JMS%';
```

To drop the queue tables listed, execute the DBMS_AQADM.DROP_QUEUE_TABLE procedure. For example, if a queue table named QTABLE1 owned by user SCOTT is listed, execute the following procedure to drop the queue table:

```
EXECUTE dbms_aqadm.drop_queue_table(queue_table => 'scott.qtable1',
    force => TRUE);
```

Continue with Your Downgrade to Release 8.1

After you have removed all of the incompatibilities with release 8.1, go to "Reset Database Compatibility" on page 13-3.

15

Downgrading to an Older Release of Oracle

The information in this chapter only applies to release 8.1 and higher installations of Oracle. The term **downgrading** describes transforming an Oracle database into a previous release of the same version, such as transforming a database from release 8.1.7 to release 8.1.5. The term **downgrading** also describes transforming an Oracle database into a previous version, such as transforming a database from Oracle9*i* to Oracle7. This chapter describes downgrading to Oracle7 or release 8.0. If you want to downgrade to release 8.1 of Oracle, then see Chapter 13, "Downgrading to Release 8.1".

This chapter covers the following topics:

- Downgrading from Oracle9i to Release 8.0
- Downgrading from Oracle9i to Oracle7

> **See Also:** Some aspects of downgrading are operating system-specific. See your operating system-specific Oracle documentation for additional instructions about downgrading.

Downgrading from Oracle9*i* to Release 8.0

An Oracle9*i* database can be downgraded to a release 8.0 database. However, few downgrade paths are available, and the necessary procedures may require a great deal of time and effort. Also, it may not be possible to preserve data that uses new Oracle9*i* features that are not available with release 8.0.

The procedure for downgrading depends on whether the Oracle9*i* database contains new or changed data that must be preserved. Use the procedure that applies to your Oracle9*i* database:

- Downgrading a Database That Does Not Contain New or Changed Data
- Downgrading a Database That Contains New or Changed Data

Downgrading a Database That Does Not Contain New or Changed Data

If the Oracle9*i* database contains *no new or changed data* that must be preserved, then simply restore the complete backup of the previous release 8.0 source database and open it again. Make sure the restore includes the initialization parameters that were used in the previous release 8.0 database.

See Also: *Oracle9i User-Managed Backup and Recovery Guide* for more information about restoring a backed up database.

Any new or changed data in the Oracle9*i* database is lost when you use the method described in the previous paragraph. If your Oracle9*i* database has new or changed data that must be preserved, then use the procedure described in "Downgrading a Database That Contains New or Changed Data" on page 15-2.

Downgrading a Database That Contains New or Changed Data

If the Oracle9*i* database contains *new or changed data* that must be preserved, then complete the following steps:

1. Use the release 8.0 Export utility to export the parts of the Oracle9*i* database containing the new or changed data.

   See Also: *Oracle9i Database Utilities* for more information about the Export and Import utilities.

2. Restore the complete backup of the previous release 8.0 database, and make sure the restore includes the previous initialization parameters.

   See Also: *Oracle9i User-Managed Backup and Recovery Guide* for more information about restoring a backed up database.

3. Open the restored release 8.0 database.
4. Use the release 8.0 Import utility to import the file previously exported from the Oracle9*i* database into the restored release 8.0 database.

Alternative Downgrading Methods

Using Export/Import to return data to a release 8.0 database is relatively simple if only a few tables have been updated using Oracle9*i*. However, copying an entire database of tables can be a long and complicated task; therefore, you should decide whether you need to return to release 8.0 before you update many tables using Oracle9*i*.

The following alternate method is available for downgrading an Oracle9*i* database to release 8.0:

1. Follow the instructions in Chapter 13, "Downgrading to Release 8.1", and downgrade your Oracle9*i* database to release 8.1.
2. Use your release 8.1 Oracle documentation to further downgrade the database to release 8.0.

   See Also: *Oracle8i Migration* for more information about downgrading from release 8.1 to release 8.0.

Downgrading from Oracle9*i* to Oracle7

An Oracle9*i* database can be downgraded to an Oracle7 database (such as release 7.3). However, few downgrade paths are available, and the necessary procedures may require a great deal of time and effort. Also, it may not be possible to preserve data that uses new Oracle9*i* features that are not available with Oracle7.

Oracle does not support downgrading from Oracle9*i* to Oracle7 using the Oracle9*i* Migration utility; in other words, the Oracle9*i* Migration utility does not support backward migration. Oracle9*i* provides no other facilities specifically for downgrading.

The procedure for downgrading depends on whether the Oracle9*i* database contains new or changed data that must be preserved. Use the procedure that applies to your Oracle9*i* database:

- Downgrading a Database That Does Not Contain New or Changed Data
- Downgrading a Database That Contains New or Changed Data

Downgrading a Database That Does Not Contain New or Changed Data

If the Oracle9*i* database contains *no new or changed data* that must be preserved, then simply restore the complete backup of the previous Oracle7 source database and open it again. Make sure the restore includes the initialization parameters that were used in the previous Oracle7 database.

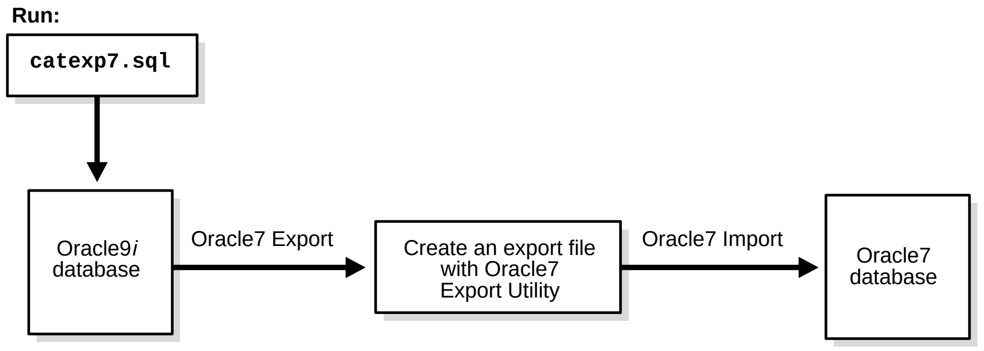
> **See Also:** *Oracle7 Server Administrator's Guide* for more information about restoring a backed up Oracle7 database.

Any new or changed data in the Oracle9*i* database is lost when you use the method described in the previous paragraph. If your Oracle9*i* database has new or changed data that must be preserved, then use the procedure described in "Downgrading a Database That Contains New or Changed Data" on page 15-5.

Downgrading a Database That Contains New or Changed Data

If the Oracle9*i* database contains *new or changed data* that must be preserved, then complete the procedure illustrated in Figure 15–1.

Figure 15–1 Downgrading to Oracle7



The following steps describe the procedure in more detail:

1. Run the `catexp7.sql` script on the Oracle9*i* database. This script is in the *`ORACLE_HOME`*`/rdbms/admin` directory.
2. Use the Oracle7 Export utility to export the parts of the Oracle9*i* database containing the new or changed data.

   See Also: *Oracle9i Database Utilities* for more information about performing an Oracle7 export from an Oracle9*i* database.

   Note: Importing version 8 or Oracle9*i* materialized views into a version 7 database is not supported. You must recreate any materialized views in your version 7 environment.

3. Restore the complete backup of the previous Oracle7 database, and make sure the restore includes the previous initialization parameters.

   See Also: *Oracle7 Server Administrator's Guide* for more information about restoring a backed up Oracle7 database.

4. Open the restored Oracle7 database.
5. Use the Oracle7 Import utility to import the file previously exported from the Oracle9*i* database into the restored Oracle7 database.

Alternative Downgrading Methods

Several other methods are available for sending table data from the Oracle9*i* database back to the Oracle7 database. These methods of returning data to Oracle7 are relatively simple if only a few tables have been updated using Oracle9*i*. However, copying an entire database of tables can be a long and complicated task; therefore, you should decide whether you need to return to Oracle7 before you update many tables using Oracle9*i*.

The following alternate methods are available for downgrading an Oracle9*i* database to Oracle7:

- Use SQL*Plus to create non-Oracle text files from the new Oracle9*i* data and then use SQL*Loader to load the data back into the Oracle7 database.
- Use the SQL*Plus `COPY` command to copy the new data from the Oracle9*i* database tables into the tables in the earlier Oracle7 database.

  See Also: *SQL*Plus User's Guide and Reference* for more information about the COPY command.

- If you are still running the earlier Oracle7 database, then re-create the table where data has been added or changed (using `CREATE TABLE ... AS SELECT`). Select the data in the Oracle9*i* database table through a distributed query from the Oracle7 database using a database link.

  See Also: *Oracle9i SQL Reference* for more information on the `AS` clause of the `CREATE TABLE` statement.

A

Troubleshooting Migration Problems

This appendix describes corrective action for common problems that you may encounter during the migration or upgrade process when you are using the Migration utility, the Oracle Data Migration Assistant, or the manual upgrade scripts.

This appendix covers the following topics:

Expected Errors During Upgrade

The following errors may occur during an upgrade to Oracle9*i*.

```
ORA-00942: table or view does not exist
ORA-01418: specified index does not exist
ORA-01432: public synonym to be dropped does not exist
ORA-01434: private synonym to be dropped does not exist
ORA-01918: user user does not exist
ORA-01919: role role does not exist
ORA-02443: Cannot drop constraint - nonexistent constraint
ORA-04043: object name does not exist
```

These errors can generally be ignored. Many objects modified during an upgrade may not exist in the original database, so these errors simply reflect the fact that databases created in earlier releases may not contain all of the more recent database objects.

The objects will be created correctly for Oracle9*i* when `catalog.sql` and `catproc.sql` are run at the end of the upgrade, or when additional catalog scripts are run to upgrade optional features.

```
ORA-01430: column being added already exists in table
```

The following columns may already exist, depending on the release being upgraded to Oracle9*i*:

- `externtype`
- `externname`
- `expctoid`
- `nttrigatt`
- `sys_evts`
- `avgcln`
- `supertoid`
- `hashcode`
- `local_attrs`
- `xflags`
- `typidcol#`
- `pls_type`

```
ORA-00955: name is already used by an existing object
```

This error occurs frequently while running `catproc.sql`. Any objects that existed in the prior release database cannot be re-created and the attempt to recreate them resulted in this error. However, any changes to these objects for use with Oracle9*i* have already been made by the upgrade scripts, so the views and PL/SQL packages created by the script should compile correctly.

```
ORA-01921: role name name conflicts with another user or role
```

A number of roles are not dropped to avoid having to re-grant privileges to them.

```
ORA-06550: line n, column n
ORA-06512: at line n
```

When PL/SQL blocks fail due to missing objects (for example, such as `ORA-00942: table or view does not exist, or missing JAVA packages`), this error is generated in addition to the original error.

```
ORA-24001: cannot create QUEUE_TABLE, SYSTEM.DEF$_AQCALL already exists
ORA-24006: cannot create QUEUE, SYSTEM.DEF$_AQCALL already exists
```

These errors occur if Advanced Queueing existed in the original database.

```
ORA-32006: name initialization parameter has been deprecated
```

This error occurs if there are obsolete initialization parameters in the initialization parameter file.

Problems Using the Migration Utility or the Oracle Data Migration Assistant

General migration problems may occur when you run either the Migration utility or the Oracle Data Migration Assistant, but they are caused by your database system's configuration. While either the utility or the assistant is performing the necessary actions to migrate the database, an error is generated by your Oracle software. Typically, when such an error occurs, the utility or assistant stops and displays one or more error messages.

If you encounter one of the following problems when you run either the utility or the assistant, then perform the suggested actions, and then rerun the utility or assistant.

Note: If you are using the Oracle Data Migration Assistant, then you must restore the backup of the database being migrated or upgraded before you rerun the assistant.

Insufficient Space in the SYSTEM Tablespace

This problem may return an error message similar to the following:

```
ORA-00604: error occurred at recursive SQL level 1
ORA-01653: unable to extend table SYS by 473 in tablespace SYSTEM
```

You need to add a new datafile to the `SYSTEM` tablespace and allocate enough space to the new datafile to successfully complete the migration.

It is also possible to run out of space in the temporary tablespace during migration. If you do, then add a new datafile to the temporary tablespace and allocate enough space to the new datafile to successfully complete the migration.

See Also: If you are using the Migration utility, then see "Space Requirements" on page 4-5 and Step 2 on page 4-22. If you are using the Oracle Data Migration Assistant, then see "Space Requirements" on page 5-5 and Step 9 on page 5-12.

These sections provide more information about the space requirements for the `SYSTEM` tablespace, and information about adding a new datafile to increase its available space.

Incorrect AUDIT_TRAIL Parameter Setting

This problem may return error messages similar to the following:

```
ORA-00604: error occurred at recursive SQL level string
ORA-01552: cannot use system rollback segment for non-system tablespace 'string'
ORA-02002: error while writing to audit trail
```

You will encounter these errors only under the following conditions:

- The `AUDIT_TRAIL` initialization parameter is set to either `DB` or to `TRUE`
- The `SYS.AUD$` table is located in a tablespace other than `SYSTEM`

To correct this problem, complete the following steps:

1. Shut down the database if it is running.

2. Set the AUDIT_TRAIL initialization parameter in the initialization parameter file in the following way:

   ```
   AUDIT_TRAIL = NONE
   ```

3. Rerun the Migration utility, or restore the backup of the database being migrated and rerun the Oracle Data Migration Assistant.

OPTIMAL Setting for the SYSTEM Rollback Segment

This problem may return error messages similar to the following:

```
ORA-01562: failed to extend rollback segment number 0
ORA-01628: max # extents (n) reached for rollback segment SYSTEM
```

These messages indicate that the SYSTEM rollback segment is too small to complete the migration. You must ensure that the SYSTEM rollback segment is large enough for the migration to complete successfully.

Both the Migration utility and the Oracle Data Migration Assistant take all non-SYSTEM rollback segments offline and then freeze the size of the SYSTEM rollback segment by altering MAXEXTENTS to the number of extents currently allocated. This action prevents any space operations, such as an extent allocation, while the utility or assistant handles the space management tables.

If the SYSTEM rollback segment has an OPTIMAL setting, then extents are de-allocated dynamically when their data is no longer needed for active transactions. The dynamic de-allocation may cause the number of currently allocated extents to be small when the SYSTEM rollback segment is frozen. Therefore, the SYSTEM rollback segment may not be large enough to handle the transactions involving the space management tables during migration.

The solution is to change the following settings:

1. Turn off the OPTIMAL setting for rollback segment.
2. Double the NEXT EXTENT of the System Rollback Segment.
3. Double the MULTIPLIER value.
4. Add space to the system tablespace to make sure there is enough free space to handle undo segment (at least 50 MB).

See Also: If you are using the Migration utility, then see Step 6 on page 4-12. If you are using the Oracle Data Migration Assistant, then see Step 7 on page 5-11. These sections provide instructions for checking your OPTIMAL setting and resetting it if necessary.

Small MULTIPLIER Option Setting

This problem may return an error message similar to the following:

```
ORA-01632: max # extents (%s) reached in index %s.%s
```

The Migration utility is using the default value of 15 for the `MULTIPLIER` option, and this value is too low. To correct the problem, increase the value of the `MULTIPLIER` option.

If you are using the Migration utility, then, when you run it from the command line, enter the following to raise the `MULTIPLIER` option to 30:

```
mig MULTIPLIER=30
```

If, however, you are running the Migration utility in the background by using the Oracle Data Migration Assistant, then restore the backup of the database being migrated and then rerun the assistant. Choose the Custom migration option in the assistant. When you are prompted for the `MULTIPLIER` value, enter a value greater than the default of 15.

See Also: "Review Migration Utility Command-Line Options" on page 4-17 for more information about the MULTIPLIER option.

Problems at the ALTER DATABASE CONVERT Statement

You may encounter one of the problems described in this section when you issue the `ALTER DATABASE CONVERT` statement during the migration process after you run the Migration utility. Typically, the conversion will stop and one or more error messages will be displayed. If you encounter one of the following problems when you run the `ALTER DATABASE CONVERT` statement, then perform the suggested actions to correct the problem.

Note: These problems should not occur if you are using the Oracle Data Migration Assistant, because the assistant runs the ALTER DATABASE CONVERT statement automatically in the background and avoids the conditions that cause these problems. Therefore, these problems only apply if you are using the Migration utility.

Oracle7 Control Files Exist

This problem may return the following error messages:

```
ORA-00200: cannot create control file name
ORA-00202: controlfile: name
ORA-27038: skgfrcre: file exists
```

The old Oracle7 control files must be renamed or removed before you issue the ALTER DATABASE CONVERT statement.

See Also: Step 1 on page 4-27 in the "Perform Migration Steps in the Oracle9i Environment" section.

Database Started in Mode Other Than NOMOUNT

This problem may return the following error messages:

```
ORA-00227:  corrupt block detected in controlfile: (block num, # blocks num)
ORA-00202:  control file: '%s'
```

The old Oracle7 control files must be renamed or removed before you issue the ALTER DATABASE CONVERT statement. Also, the database must be started in NOMOUNT mode when you issue the ALTER DATABASE CONVERT statement. This error indicates that the database was started in a mode other than NOMOUNT.

See Also: Step 1 on page 4-27 and Step 9 on page 4-31 in the "Perform Migration Steps in the Oracle9i Environment" section.

Convert File Not Found

This problem may return the following error messages:

```
ORA-00404: convert file not found: name
ORA-27037: unable to obtain file status
```

The convert file (conv*sid*.dbf on UNIX and convert.ora on Windows platforms) generated by the Migration utility was not found in the expected location. On UNIX, the expected location is the *ORACLE_HOME*/dbs directory in the Oracle9*i* environment; on Windows platforms, the expected location is the *ORACLE_HOME*\rdbms directory in the Oracle9*i* environment. The convert file must be moved to this location before you issue the ALTER DATABASE CONVERT statement.

> **See Also:** Step 2 on page 4-28 in the "Perform Migration Steps in the Oracle9i Environment" section.

REMOTE_LOGIN_PASSWORDFILE Initialization Parameter Set to EXCLUSIVE

This problem may return the following error message:

```
ORA-00600: internal error code, arguments: [kzsrsdn: 1], [32]
```

You will encounter this error under the following conditions:

- Your database is using a password file, and the password file was not moved to the correct directory. On UNIX, the correct directory is *ORACLE_HOME*/dbs in the Oracle9*i* environment; on Windows platforms, the correct directory is *ORACLE_HOME*\database in the Oracle9*i* environment.
- The REMOTE_LOGIN_PASSWORDFILE initialization parameter is set to EXCLUSIVE in the initialization parameter file.

To continue with the migration, complete the following steps:

1. Shutdown the database.
2. Set REMOTE_LOGIN_PASSWORDFILE to NONE in the initialization parameter file:

   ```
   REMOTE_LOGIN_PASSWORDFILE = NONE
   ```

3. Startup mount the database by entering the following SQL statement:

   ```
   SQL> STARTUP MOUNT
   ```

   You may need to use the PFILE option to specify the location of your initialization parameter file.
4. Issue the ALTER DATABASE OPEN RESETLOGS statement:

   ```
   SQL> ALTER DATABASE OPEN RESETLOGS;
   ```

5. Continue with the migration process starting with Step 12 on page 4-33.

You cannot use the existing password file because it is no longer valid. If you want to use a password file with Oracle9*i*, then recreate the password file and repopulate it with users. Remember to set `REMOTE_LOGIN_PASSWORDFILE` correctly.

Database Name Mismatch

This problem may return the following error message:

```
ORA-01103: database name 'string' in controlfile is not 'string'
```

There is a mismatch in the database name. This mismatch is in one or more of the following places:

- The database name specified by the `DB_NAME` initialization parameter in the initialization parameter file does not match the database name in the `conv`*`sid`*`.dbf` filename.
- The Oracle9*i* instance ID set by the `ORACLE_SID` environment variable does not match the database name in the `conv`*`sid`*`.dbf` filename.

Note: This problem only occurs on UNIX operating systems. It does not apply to Windows platforms.

To correct the problem, make sure the correct database name is specified in each of the following places:

- the `ORACLE_SID` environment variable
- the `DB_NAME` initialization parameter in the initialization parameter file
- the *`sid`* part of the `conv`*`sid`*`.dbf` filename

For example, if your `ORACLE_SID` environment variable and the `DB_NAME` initialization parameter in the initialization parameter file are both set to DB1, then the `conv`*`sid`*`.dbf` filename should be the following:

```
convDB1.dbf
```

Rerunning the ALTER DATABASE CONVERT Statement

This problem may return the following error messages:

```
ORA-01122: datafile name - failed verification check
```

```
ORA-01110: data file name: str
ORA-01202: wrong incarnation of this file - wrong creation time
```

These errors usually indicate that the `ALTER DATABASE CONVERT` statement was issued previously but failed. If you encounter these errors, then you can attempt to move on to the next step in the migration process by issuing the `ALTER DATABASE OPEN RESETLOGS` statement. However, if you encounter problems, then restore the backup you created before you started the migration process, and use it to start the migration again from the beginning. Start at the beginning of Chapter 4, but make sure you performed the pre-migration actions described in Chapter 3.

Datafile Version Integrity Problem

This problem may return the following error messages:

```
ORA-01122: datafile name - failed verification check
ORA-01110: data file name: str
ORA-01211: Oracle7 data file is not from migration to Oracle9i
```

The Migration utility must be the last utility to access the database in the Oracle7 environment. The datafile specified in the error messages is either a backup taken before you ran the Migration utility, or the database was opened by Oracle7 after you ran the Migration utility. Only the datafiles that were current when the Migration utility ran can be accessed by Oracle9*i*.

To ensure datafile version integrity, the system change numbers (SCNs) in the data dictionary, the convert file, and the file headers must all be consistent when the database is converted to Oracle9*i*. If the database is opened under Oracle7 after the Migration utility has run, then the SCN checking fails when you issue the `ALTER DATABASE CONVERT` statement.

To correct the problem, complete the following steps:

1. Shutdown the database.
2. Rename the control files created by `ALTER DATABASE CONVERT` to different file names.
3. Restore the saved copy of Oracle7 control files from immediately before the issuing of the `STARTUP NOMOUNT` statement.

   If you do not have the Oracle7 control files saved, then restore the backup you made prior to starting the migration process.

4. Start the migration process over from the beginning, ensuring the database is not opened in the Oracle7 environment after the Migration utility completes. Start from the beginning of Chapter 4.

Problems While Running the Manual Migration and Upgrade Scripts

You may encounter the problem described in this section when you run the migration scripts after you run the Migration utility, or when you run the scripts required to manually upgrade your database. The problem described in this section does not apply to the Oracle Data Migration Assistant.

Script Runs for an Inordinately Long Time

If the manual migration script (`u0703040.sql`) script or a manual upgrade script runs for an inordinately long time, then it may be caused by a setting for `LARGE_POOL_SIZE` that is too large for your installation. Use the `V$PARAMETER` view to check the setting for `LARGE_POOL_SIZE`, and if it is too large, then set it to a smaller value in your initialization parameter file. See "Parallel Execution Allocated from Large Pool" on page B-12 for more information. After you adjust the `LARGE_POOL_SIZE` setting, rerun the script.

Migration Utility Messages

The Migration utility may return error messages and informational messages during migration. This section describes errors you may encounter when using the Migration utility. For each error, a description of its probable cause and instructions for corrective action are provided. Informational messages also are listed, but they require no corrective action.

If you are using the Oracle Data Migration Assistant, then the Migration utility messages are recorded in a log file. See the online help for the Oracle Data Migration Assistant for information about accessing its log files. Also, if you are using the Oracle Data Migration Assistant and the recommended action for a message includes rerunning the Migration utility, then you should rerun the Oracle Data Migration Assistant.

The following messages are listed in alphabetical order:

cannot reduce file number bits in DBA during migration

Cause: The Migration utility attempted to reduce the number of file-number bits used in a datablock address.

Action: Contact Oracle Support Services.

cannot create conversion file, records exceed *number* bytes

Cause: An internal error occurred. A valid convert file could not be created from the Oracle7 control file.

Action: Check the Oracle7 control file for corruption, fix any problems, and rerun the Migration utility.

CHECK_ONLY - estimate V8 catalog space requirement ONLY (default=FALSE)

Cause: This is an informational message about the `CHECK_ONLY` command line argument.

Action: No user action is required.

CHECK_ONLY and NO_SPACE_CHECK are mutually exclusive options

Cause: These two mutually exclusive command-line options were passed to the Migration utility.

Action: Rerun the Migration utility using only one of these options.

client nls_characterset does not match server nls_characterset - check that NLS_LANG environment variable is set

Cause: The `NLS_LANG` character set does not match the character set in `PROPS$`.

Action: Check the database character set in `PROPS$` and set the `NLS_LANG` environment variable to match it.

command line argument value must be TRUE or FALSE (*string*)

Cause: You entered a command-line argument with a value other than `true` or `false`.

Action: Check the syntax of the command-line argument, correct the statement, and retry the operation.

command line arguments must be of the form <keyword>=<value> (*string*)

Cause: You used a command-line argument improperly.

Action: Check the syntax of the command-line argument, correct the statement, and retry the operation.

command line arguments:

Cause: This informational message displays the command-line arguments.

Action: No user action is required.

command name not found (*string*)

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle9*i* software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

command not of form CMD (ARG1, ARG2, ...)

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle9*i* software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

copy long command must be of form COPYLONG(U1,T1,C1,U2,T2,C2,K1<,K2>)

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle7 software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

could not find single contiguous extent of *number* bytes for c_file#_block#

Cause: You do not have enough contiguous space in your `SYSTEM` tablespace.

Action: Add free space to your `SYSTEM` tablespace, and rerun the Migration utility.

could not find single contiguous extent of *number* bytes for c_ts#

Cause: You do not have enough contiguous space in your `SYSTEM` tablespace.

Action: Add free space to your `SYSTEM` tablespace, and rerun the Migration utility.

could not find single contiguous extent of *number* bytes for i_file#_block#

Cause: You do not have enough contiguous space in your `SYSTEM` tablespace.

Action: Add free space to your `SYSTEM` tablespace, and rerun the Migration utility.

could not find single contiguous extent of *number* bytes for i_ts#

Cause: You do not have enough contiguous space in your `SYSTEM` tablespace.

Action: Add free space to your `SYSTEM` tablespace, and rerun the Migration utility.

could not translate logical name *name*

Cause: An internal error has occurred.

Action: Check that the logical name is defined correctly, and rerun the Migration utility.

current version: *str* **-- Database must be Oracle7.1 or later**

Cause: Current database is an earlier version than Oracle7, release 7.1.

Action: Migrate or upgrade current database to a release supported by the Migration utility on your operating system. Then, rerun the Migration utility. See your operating system-specific Oracle documentation for information about the releases supported by the Migration utility on your operating system.

data type must be long for column *string*

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle7 software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

datafiles is found in inconsistent states (internal error) -- *filename*

Cause: An internal error occurred; a datafile was found in inconsistent state.

Action: Contact Oracle Support Services.

datafile is offline while tablespace is online - apply media recovery and bring datafile online before migration -- *datafile*

Cause: The datafile in a tablespace is offline while the tablespace is online. Migration cannot proceed until the datafile and tablespace are both either online or offline normal.

Action: Apply media recovery and bring the datafile online before rerunning the migration.

DBNAME - current database name (db_name in init.ora)

Cause: This is an informational message about the DBNAME command-line argument.

Action: No user action is required.

dictionary constant not found - *name*

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle9*i* software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

entries found in system.def$_call, def$_calldest or def$_error - push all deferred transactions before migration

Cause: Entries exist in SYSTEM.DEF$_CALL, DEF$_CALLDEST, or DEF$_ERROR.

Action: If entries are in SYSTEM.DEF$_CALL, push all deferred transactions until SYSTEM.DEF$_CALL is empty. If entries are in SYSTEM.DEF$_ERROR, resolve and re-execute any errors in the local queue until it is empty. Rerun the Migration utility.

error calling slgtd

Cause: Error in getting current time from slgtd, an internal error. The Migration utility may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle9*i* software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

error closing file *string*

Cause: An internal error has occurred. Data could not be written to disk.

Action: Check that the file access permissions are correct, that you have enough space or quota to write this file, and that the disk is not corrupt. Fix any problems, and rerun the Migration utility.

estimated space requirement for *object* **is** *number* **blocks**

Cause: In this informational message, the Migration utility displays the space required for the object.

Action: No user action is required.

file *filename* **is too large for DBA conversion**

Cause: An internal error has occurred; *filename* is too large for DBA conversion.

Action: Contact Oracle Support Services.

file header does not fit in *number* **bytes**

Cause: An internal error has occurred.

Action: Check the control file for corruption, fix any problems, and rerun the Migration utility.

fixed portion of control file does not fit in *number* **bytes**

Cause: An internal error has occurred.

Action: Check the control file for corruption, fix any problems, and rerun the Migration utility.

found NULL SQL statement

Cause: An internal error has occurred; the `migrate.bsq` script may be corrupted.

Action: Check that the version of the Migration utility, of `migrate.bsq`, and of the target Oracle9*i* software are compatible, and that no corruption exists in `migrate.bsq`. Fix any problems, and rerun the Migration utility.

free space found in system tablespace is *number* **blocks**

Cause: This informational message shows the amount of free space in the SYSTEM tablespace.

Action: No user action is needed.

free space found: *number*

Cause: This informational message shows the amount of free space in the SYSTEM tablespace.

Action: No user action is needed.

incomplete write

Cause: An internal error has occurred. Data could not be written to disk.

Action: Check that the file access permissions are correct, that you have enough space or quota to write this file, and that the disk is not corrupt. Fix any problems, and rerun the Migration utility.

insufficient space for new dictionaries, *number* **bytes needed,** *number* **found**

Cause: There is insufficient room in your SYSTEM tablespace for the new data dictionary information.

Action: Allocate the additional space required in the SYSTEM tablespace, and rerun the Migration utility.

invalid NLS_NCHAR value specified

Cause: The NLS_NCHAR value specified in the command line is invalid.

Action: Correct the NLS_NCHAR value specified in the command line, and rerun the Migration utility.

migration can't proceed - database blocksize size is less than Oracle9*i*'s minimum block size 2k

Cause: The existing database blocksize is less than 2KB.

Action: Make sure the block size of the Oracle7 database is at least 2KB. You may consider rebuilding the Oracle7 database. Then, rerun the Migration utility.

migration can't proceed with datafile online while tablespace offline -- *datafile*

Cause: The datafile in a tablespace is online while the tablespace is offline. Migration cannot proceed until the datafile and tablespace are both either online or offline normal.

Action: Make sure the online status of the datafile is the same as the online status of the tablespace, and rerun the Migration utility.

migration cannot proceed with active transaction or offline tablespaces with outstanding undo

Cause: One or more tablespaces were offline with outstanding save undo when the Migration utility attempted to migrate the database.

Action: If you are using the Migration utility, go to Step 3 on page 4-11 and make sure all offline tablespaces have been taken offline cleanly. If you are using the Oracle Data Migration Assistant, go to Step 4 on page 5-10 and make sure all offline tablespaces have been taken offline cleanly. Then, rerun the Oracle9*i* Migration utility.

mounting database ...

Cause: This is an informational message. The Migration utility is mounting the Oracle7 database.

Action: No user action is required.

MULTIPLIER - seg$/uet$ cluster index size increase factor (default=15)

Cause: This is an informational message that the Migration utility displays about the MULTIPLIER command-line setting.

Action: No user action is required.

MULTIPLIER value must be at least 2

Cause: The MULTIPLIER value, which specifies the initial size of the Oracle9*i* i_file#_block# in the command line, is less than 2.

Action: Change the MULTIPLIER value to be equal to or greater than 2, and rerun the Migration utility.

NEW_DBNAME *name* **too long - maximum length is 8 characters**

Cause: The new database name specified is more than 8 characters long.

Action: Change the specified name for the new database to 8 or fewer characters, and rerun the Migration utility.

NEW_DBNAME - new name for the database (max. 8 characters)

Cause: This informational message displays information about the NEW_DBNAME command-line argument.

Action: No user action is required.

NLS_NCHAR - specify the nchar characterset value

Cause: This informational message displays information about the NLS_NCHAR command-line argument.

Action: No user action is required.

NO_SPACE_CHECK - do not execute the space check (default=FALSE)

Cause: This is an informational message about the NO_SPACE_CHECK command-line argument.

Action: No user action is required, but make sure there is adequate space before you run the Migration utility with this option.

tablespace/datafile **number being processed is incorrect during creating convert file**

Cause: An internal error occurred while creating the convert file.

Action: Contact Oracle Support Services.

opening database ...

Cause: This is an informational message. The Migration utility is opening the Oracle7 database.

Action: No user action is required.

ORA_NLS33 environment variable is not set or incorrectly set

Cause: The ORA_NLS33 environment variable does not point to the NLS datafiles.

Action: Set the ORA_NLS33 environment variable to point to the correct files, and rerun the Migration utility.

ORA-*number*:

Cause: The Migration utility has received an ORA error and cannot retrieve the message text for the error.

Action: Take appropriate action based on the Oracle error *number* (see *Oracle9i Database Error Messages*).

parameter buffer overflow

Cause: The initialization parameter file is too large to fit in the buffer.

Action: Reduce the size of the parameter file, possibly by removing any obsolete parameters, and rerun the Migration utility.

parameter file exceeds *number* **bytes**

Cause: The parameter file for your Oracle7 database exceeds the maximum size.

Action: If possible, reduce the size of your parameter file by removing obsolete parameters. Otherwise, contact Oracle Support Services.

PFILE - use alternate init.ora file

Cause: This is an informational message that displays information about the PFILE command-line argument.

Action: No user action is required.

seek error in file *name*

Cause: An internal error has occurred reading file *name*.

Action: Make sure the file and disk are not corrupted. Fix any corruption before you rerun the Migration utility.

short read, *number* **bytes requested,** *number* **bytes read**

Cause: There is a problem reading the control file.

Action: Check the control file for corruption, fix any problems, and rerun the Migration utility.

shut down database (abort) ...

Cause: An internal error occurred.

Action: Additional error messages should inform you of the cause of the shutdown. Follow the actions suggested for these additional messages.

shutting down database ...

Cause: This is an informational message. The Migration utility is shutting down the Oracle7 database.

Action: No user action is required.

SPOOL - spool output to file

Cause: This is an informational message that displays information about the SPOOL command-line argument.

Action: No user action is required.

starting up database ...

Cause: This is an informational message. The Migration utility is starting up an Oracle7 instance.

Action: No user action is required.

string argument too long, maximum length *number*

Cause: A string in the command line argument passed to the Migration utility exceeds the maximum size.

Action: Shorten the string in the command line argument, and rerun the Migration utility.

tablespace of datafile not taken offline normal. Bring tablespace online, offline normal or drop before migration -- *tablespace*

Cause: Tablespace was taken offline using IMMEDIATE or TEMPORARY.

Action: Bring tablespace online, and then take it offline using NORMAL or drop it. Then, rerun the Migration utility.

too many args in command (*number* **max)**

Cause: You specified too many arguments on the command-line.

Action: Check the syntax of the command and specify fewer command-line options.

unable to allocate buffer space to copy longs

Cause: The Migration utility could not allocate memory to serve as a buffer for copying LONG columns in the database.

Action: Make sure enough computer resources are available for the Migration utility, and rerun the Migration utility.

unable to open file *name*

Cause: An internal error has occurred, or a file was not in the expected location, when you started the Oracle9*i* Migration utility.

Action: Check that the file exists and that its access permissions allow Oracle to open and read it. If possible, check that the file, and the disks on which the file resides, are not corrupt. Fix any problems, and rerun the Migration utility.

unable to read file *name*

Cause: An internal error occurred or a file was not in the expected location when you started the Migration utility.

Action: Check that the file exists and that its access permissions allow Oracle to open and read it. If possible, check that the file, and the disks on which the file resides, are not corrupt. Fix any problems, and rerun the Migration utility.

unable to write file *name*

Cause: An internal error occurred.

Action: Check the access permissions to make sure that Oracle can write to the file. Check that the disks to which the file is being written are not corrupt. Fix any corruption; then, rerun the Migration utility.

V8 catalog space requirement: *number*

Cause: This is an informational message that shows the amount of additional space required in your `SYSTEM` tablespace to run the Migration utility successfully.

Action: Make sure you have the specified amount of additional space before running the Migration utility.

B

Changes to Initialization Parameters

Oracle9*i* supports new initialization parameters for use in the initialization parameter file, and some initialization parameters have been renamed or have become obsolete in Oracle9*i*. Typically, the initialization parameter file is named `init`*`sid`*`.ora`, where *sid* is your database instance name. However, the initialization parameter file may be named differently in your environment. This appendix lists the new, renamed, and obsolete initialization parameters in version 8 and Oracle9*i*, and this appendix discusses compatibility issues with certain initialization parameters.

This appendix covers the following topics:

- Changes to Initialization Parameters in Oracle9i
- Changes to Initialization Parameters in Version 8
- Compatibility Issues with Initialization Parameters

See Also: *Oracle9i Database Reference* for detailed information about the initialization parameters in Oracle9*i*.

Note: Some of the initialization parameters listed in this appendix are operating system-specific. See your operating system-specific Oracle documentation for more information about these initialization parameters.

Changes to Initialization Parameters in Oracle9*i*

The following sections list the new, renamed, and obsolete initialization parameters in Oracle9*i*:

- Initialization Parameters Added in Oracle9i
- Initialization Parameters Renamed in Oracle9i
- Initialization Parameters Obsolete in Oracle9i

Initialization Parameters Added in Oracle9*i*

The initialization parameters listed in this section are new in Oracle9*i*.

Initialization Parameters Added in Release 9.0.1

The following initialization parameters were added in release 9.0.1:

ARCHIVE_LAG_TARGET	CIRCUITS
DB_*n*K_CACHE_SIZE	DB_CACHE_ADVICE
DB_CACHE_SIZE	DB_CREATE_FILE_DEST
DB_CREATE_ONLINE_LOG_DEST_*n*	DB_KEEP_CACHE_SIZE
DB_RECYCLE_CACHE_SIZE	DISPATCHERS
DRS_START	FAL_CLIENT
FAL_SERVER	FAST_START_MTTR_TARGET
GLOBAL_CONTEXT_POOL_SIZE	LOG_ARCHIVE_DEST_*n* (*n* = 6, 7, ... 10)
LOG_ARCHIVE_DEST_STATE_*n* (*n* = 6, 7, ... 10)	LOGMNR_MAX_PERSISTENT_SESSIONS
MAX_DISPATCHERS	MAX_SHARED_SERVERS
NLS_LENGTH_SEMANTICS	NLS_NCHAR_CONV_EXCP
NLS_TIMESTAMP_FORMAT	NLS_TIMESTAMP_TZ_FORMAT
PGA_AGGREGATE_TARGET	PLSQL_COMPILER_FLAGS
PLSQL_NATIVE_C_COMPILER	PLSQL_NATIVE_LIBRARY_DIR
PLSQL_NATIVE_LIBRARY_SUBDIR_COUNT	PLSQL_NATIVE_LINKER
PLSQL_NATIVE_MAKE_FILE_NAME	PLSQL_NATIVE_MAKE_UTILITY
REMOTE_ARCHIVE_ENABLE	REMOTE_LISTENER
SGA_MAX_SIZE	SHARED_SERVER_SESSIONS

SHARED_SERVERS	SPFILE
STANDBY_FILE_MANAGEMENT	STANDBY_PRESERVES_NAMES
trace_enabled	UNDO_MANAGEMENT
UNDO_RETENTION	UNDO_SUPPRESS_ERRORS
UNDO_TABLESPACE	WORKAREA_SIZE_POLICY

Initialization Parameters Renamed in Oracle9*i*

The initialization parameters listed in this section have been renamed in Oracle9*i*.

Initialization Parameters Renamed in Release 9.0.1

The following initialization parameters were renamed in release 9.0.1:

Table B–1 Initialization Parameters Renamed in Release 9.0.1

Pre-Release 9.0.1 Name	Release 9.0.1 Name
OPS_INTERCONNECTS	CLUSTER_INTERCONNECTS
PARALLEL_SERVER	CLUSTER_DATABASE
PARALLEL_SERVER_INSTANCES	CLUSTER_DATABASE_INSTANCES

Initialization Parameters Obsolete in Oracle9*i*

The initialization parameters listed in this section are obsolete in Oracle9*i*.

Initialization Parameters Obsolete in Release 9.0.1

The following initialization parameters became obsolete in release 9.0.1 and cannot be used in release 9.0.1 and higher:

ALWAYS_ANTI_JOIN	ALWAYS_SEMI_JOIN
DB_BLOCK_LRU_LATCHES	DB_BLOCK_MAX_DIRTY_TARGET
DB_FILE_DIRECT_IO_COUNT	GC_DEFER_TIME
GC_RELEASABLE_LOCKS	GC_ROLLBACK_LOCKS
HASH_MULTIBLOCK_IO_COUNT	INSTANCE_NODESET
JOB_QUEUE_INTERVAL	LM_LOCKS
LM_RESS	OPTIMIZER_PERCENT_PARALLEL

SORT_MULTIBLOCK_READ_COUNT	TEXT_ENABLE

Note: An attempt to start a release 9.0.1 database using one or more of these obsolete initialization parameters will succeed, but a warning will be returned and recorded in the alert log.

Changes to Initialization Parameters in Version 8

The following sections list the new, renamed, and obsolete initialization parameters in version 8:

- Initialization Parameters Added in Version 8
- Initialization Parameters Renamed in Version 8
- Initialization Parameters Obsolete in Version 8

Initialization Parameters Added in Version 8

The initialization parameters listed in this section are new in version 8.

Initialization Parameters Added in Release 8.0

The following initialization parameters were added in release 8.0:

ALLOW_PARTIAL_SN_RESULTS	ALWAYS_SEMI_JOIN
AQ_TM_PROCESSES	ARCH_IO_SLAVES
BACKUP_DISK_IO_SLAVES	BACKUP_TAPE_IO_SLAVES
BUFFER_POOL_KEEP	BUFFER_POOL_RECYCLE
COMPLEX_VIEW_MERGING	CONTROL_FILE_RECORD_KEEP_TIME
DB_BLOCK_MAX_DIRTY_TARGET	DB_FILE_DIRECT_IO_COUNT
DB_FILE_NAME_CONVERT	DB_WRITER_PROCESSES
DBWR_IO_SLAVES	DISK_ASYNCH_IO
FAST_FULL_SCAN_ENABLED	FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY
GC_DEFER_TIME	GC_LATCHES
HI_SHARED_MEMORY_ADDRESS	INSTANCE_GROUPS
LARGE_POOL_MIN_ALLOC	LARGE_POOL_SIZE

LGWR_IO_SLAVES	LM_LOCKS
LM_PROCS	LM_RESS
LOCAL_LISTENER	LOCK_NAME_SPACE
LOCK_SGA	LOCK_SGA_AREAS
LOG_ARCHIVE_DUPLEX_DEST	LOG_ARCHIVE_MIN_SUCCEED_DEST
LOG_FILE_NAME_CONVERT	MTS_RATE_LOG_SIZE
MTS_RATE_SCALE	NLS_CALENDAR
O7_DICTIONARY_ACCESSIBILITY	OBJECT_CACHE_MAX_SIZE_PERCENT
OBJECT_CACHE_OPTIMAL_SIZE	OGMS_HOME
OPEN_LINKS_PER_INSTANCE	OPS_ADMIN_GROUP
OPTIMIZER_FEATURES_ENABLE	OPTIMIZER_INDEX_CACHING
OPTIMIZER_INDEX_COST_ADJ	OPTIMIZER_MAX_PERMUTATIONS
PARALLEL_ADAPTIVE_MULTI_USER	PARALLEL_BROADCAST_ENABLED
PARALLEL_EXECUTION_MESSAGE_SIZE	PARALLEL_INSTANCE_GROUP
PARALLEL_MIN_MESSAGE_POOL	PARALLEL_SERVER
PARALLEL_TRANSACTION_RESOURCE_TIMEOUT	PLSQL_V2_COMPATIBILITY
PUSH_JOIN_PREDICATE	READ_ONLY_OPEN_DELAYED
REPLICATION_DEPENDENCY_TRACKING	SERIAL_REUSE
SESSION_MAX_OPEN_FILES	SHARED_MEMORY_ADDRESS
STAR_TRANSFORMATION_ENABLED	TAPE_ASYNCH_IO
TIMED_OS_STATISTICS	TRANSACTION_AUDITING
USE_INDIRECT_DATA_BUFFERS	

Initialization Parameters Added in Release 8.1

The following initialization parameters were added in release 8.1:

ACTIVE_INSTANCE_COUNT	CURSOR_SHARING
DB_BLOCK_CHECKING	FAST_START_IO_TARGET
FAST_START_PARALLEL_ROLLBACK	HS_AUTOREGISTER
INSTANCE_NAME	JAVA_MAX_SESSIONSPACE_LIMIT
JAVA_POOL_SIZE	JAVA_SOFT_SESSIONSPACE_LIMIT

LOG_ARCHIVE_DEST_*n* (*n* = 1, 2, ... 5)	LOG_ARCHIVE_DEST_STATE_*n* (*n* = 1, 2, ... 5)
LOG_ARCHIVE_MAX_PROCESSES	LOG_ARCHIVE_TRACE
MTS_CIRCUITS	MTS_SESSIONS
NLS_COMP	NLS_DUAL_CURRENCY
OPS_INTERCONNECTS	PARALLEL_AUTOMATIC_TUNING
PARALLEL_SERVER_INSTANCES	PARALLEL_THREADS_PER_CPU
QUERY_REWRITE_ENABLED	QUERY_REWRITE_INTEGRITY
RESOURCE_MANAGER_PLAN	SERVICE_NAMES
SORT_MULTIBLOCK_READ_COUNT	STANDBY_ARCHIVE_DEST

Initialization Parameters Renamed in Version 8

The initialization parameters listed in this section have been renamed in version 8.

Initialization Parameters Renamed in Release 8.0

The following initialization parameters were renamed in release 8.0:

Table B–2 Initialization Parameters Renamed in Release 8.0

Pre-Release 8.0 Name	Release 8.0 Name
ASYNC_READ	DISK_ASYNCH_IO
ASYNC_WRITE	DISK_ASYNCH_IO
CCF_IO_SIZE *	DB_FILE_DIRECT_IO_COUNT *
DB_FILE_STANDBY_NAME_CONVERT	DB_FILE_NAME_CONVERT
DB_WRITERS	DBWR_IO_SLAVES
LOG_FILE_STANDBY_NAME_CONVERT	LOG_FILE_NAME_CONVERT
SNAPSHOT_REFRESH_INTERVAL	JOB_QUEUE_INTERVAL

* The units are different for CCF_IO_SIZE (bytes) and DB_FILE_DIRECT_IO_COUNT (database blocks).

Initialization Parameters Obsolete in Version 8

The initialization parameters listed in this section are obsolete in version 8.

Initialization Parameters Obsolete in Release 8.0

The following initialization parameters became obsolete in release 8.0 and cannot be used in release 8.0 and higher:

CHECKPOINT_PROCESS	FAST_CACHE_FLUSH
GC_DB_LOCKS	GC_FREELIST_GROUPS
GC_ROLLBACK_SEGMENTS	GC_SAVE_ROLLBACK_LOCKS
GC_SEGMENTS	GC_TABLESPACES
INIT_SQL_FILES	IO_TIMEOUT
IPQ_ADDRESS	IPQ_NET
LM_DOMAINS	LM_NON_FAULT_TOLERANT
MLS_LABEL_FORMAT	OPTIMIZER_PARALLEL_PASS
PARALLEL_DEFAULT_MAX_SCANS	PARALLEL_DEFAULT_SCAN_SIZE
POST_WAIT_DEVICE	SEQUENCE_CACHE_HASH_BUCKETS
UNLIMITED_ROLLBACK_SEGMENTS	USE_IPQ
USE_POST_WAIT_DRIVER	USE_READV
USE_SIGIO	V733_PLANS_ENABLED

Note: An attempt to start a release 8.0 or higher database using one or more of these obsolete initialization parameters will result in an error, and the database will not start.

Initialization Parameters Obsolete in Release 8.1

The following initialization parameters became obsolete in release 8.1 and cannot be used in release 8.1 and higher:

ALLOW_PARTIAL_SN_RESULTS	ARCH_IO_SLAVES
B_TREE_BITMAP_PLANS	BACKUP_DISK_IO_SLAVES
CACHE_SIZE_THRESHOLD	CLEANUP_ROLLBACK_ENTRIES
CLOSE_CACHED_OPEN_CURSORS	COMPATIBLE_NO_RECOVERY
COMPLEX_VIEW_MERGING	DB_BLOCK_CHECKPOINT_BATCH
DB_BLOCK_LRU_EXTENDED_STATISTICS	DB_BLOCK_LRU_STATISTICS
DB_FILE_SIMULTANEOUS_WRITES	DELAYED_LOGGING_BLOCK_CLEANOUTS

DISCRETE_TRANSACTIONS_ENABLED	DISTRIBUTED_LOCK_TIMEOUT
DISTRIBUTED_RECOVERY_CONNECTION_ HOLD_TIMEFAST_FULL_SCAN_ENABLED	ENT_DOMAIN_NAME
FREEZE_DB_FOR_FAST_INSTANCE_RECOVERY	GC_LATCHES
GC_LCK_PROCS	JOB_QUEUE_KEEP_CONNECTIONS
LARGE_POOL_MIN_ALLOC	LGWR_IO_SLAVES
LM_PROCS	LOCK_SGA_AREAS
LOG_ARCHIVE_BUFFER_SIZE	LOG_ARCHIVE_BUFFERS
LOG_BLOCK_CHECKSUM	LOG_FILES
LOG_SIMULTANEOUS_COPIES	LOG_SMALL_ENTRY_MAX_SIZE
MAX_TRANSACTION_BRANCHES	MTS_LISTENER_ADDRESS
MTS_MULTIPLE_LISTENERS	MTS_RATE_LOG_SIZE
MTS_RATE_SCALE	MTS_SERVICE
OGMS_HOME	OPS_ADMIN_GROUP
OPTIMIZER_SEARCH_LIMIT	PARALLEL_DEFAULT_MAX_INSTANCES
PARALLEL_MIN_MESSAGE_POOL	PARALLEL_SERVER_IDLE_TIME
PARALLEL_TRANSACTION_RESOURCE_ TIMEOUT	PUSH_JOIN_PREDICATE
REDUCE_ALARM	ROW_CACHE_CURSORS
SEQUENCE_CACHE_ENTRIES	SEQUENCE_CACHE_HASH_BUCKETS
SHARED_POOL_RESERVED_MIN_ALLOC	SNAPSHOT_REFRESH_KEEP_CONNECTIONS
SNAPSHOT_REFRESH_PROCESSES	SORT_DIRECT_WRITES
SORT_READ_FAC	SORT_SPACEMAP_SIZE
SORT_WRITE_BUFFER_SIZE	SORT_WRITE_BUFFERS
SPIN_COUNT	TEMPORARY_TABLE_LOCKS
USE_ISM	

Note: An attempt to start a release 8.1 database using one or more of these obsolete initialization parameters will succeed, but a warning will be returned and recorded in the alert log.

Compatibility Issues with Initialization Parameters

The lists of new, changed, and obsolete initialization parameters earlier in this appendix show differences in initialization parameters across different releases of Oracle. However, certain initialization parameter changes require special attention because they may raise compatibility issues for your database. These parameter changes are described in this section.

New Default Value for DB_BLOCK_CHECKSUM

Starting with release 9.0.1, the `DB_BLOCK_CHECKSUM` initialization parameter has a new default value. In previous releases, the default value was `false`, but in release 9.0.1 and higher, the default value is `true`.

See Also: `DB_BLOCK_CHECKSUM` in *Oracle9i Database Reference*

Maximum Number of Job Queue Processes

In Oracle9*i*, the maximum number of job queue processes that can be spawned per instance is 1000. In previous releases, the maximum number was 36. The `JOB_QUEUE_PROCESSES` initialization parameter controls the number of job queue processes.

See Also: `JOB_QUEUE_PROCESSES` in *Oracle9i Database Reference*

The ORACLE_TRACE_ENABLE Parameter

Starting with release 8.1.7, the `ORACLE_TRACE_ENABLE` initialization parameter is dynamic. The default value is `false`.

To enable Oracle Trace collections for the server, use `ALTER SYSTEM` or `ALTER SESSION` to set `ORACLE_TRACE_ENABLE` to `true`. This setting alone does not start an Oracle Trace collection, but it allows Oracle Trace to be used with the server.

With ORACLE_TRACE_ENABLE set to true, Oracle Trace collection of server event data can then be performed in one of the following ways:

- Use the Oracle Trace Manager application (supplied with the Oracle Diagnostic Pack).
- Use the Oracle Trace command line interface (supplied with the server).
- Specify a collection name in the ORACLE_TRACE_COLLECTION_NAME server parameter.

See Also: *Oracle9i Database Reference* and *Oracle9i Database Performance Guide and Reference*.

The SERIALIZABLE Parameter

Starting with release 8.1.6, setting the SERIALIZABLE initialization parameter to true is no longer supported. This is not the same as "obsolete". The parameter still shows up as a valid parameter in the V$PARAMETER data dictionary view.

The default behavior henceforth is as if SERIALIZABLE were set to FALSE. Use the SET TRANSACTION ISOLATION LEVEL SERIALIZABLE command to achieve similar transaction isolation behavior. You can also use ALTER SESSION SET ISOLATION_LEVEL=SERIALIZABLE to get the behavior for a full session.

SORT_AREA_SIZE and SORT_DIRECT_WRITES Parameters

The SORT_DIRECT_WRITES initialization parameter is obsolete in release 8.1 and higher. If you had SORT_DIRECT_WRITES set to FALSE or AUTO in a past release, then the sort buffers were kept in the buffer cache whenever possible. Because SORT_DIRECT_WRITES is obsolete in release 8.1, the sort buffers could go directly to disk if you do not adjust your SORT_AREA_SIZE initialization parameter.

You should increase the value of SORT_AREA_SIZE in release 8.1 if either of the following conditions were true in a past release:

- SORT_DIRECT_WRITES was set to FALSE.
- SORT_DIRECT_WRITES was set to AUTO, and SORT_AREA_SIZE was set to 640 KB or less.

If either of these conditions were true in a past release, then increase the value of SORT_AREA_SIZE for better performance.

New Default Value for LOG_CHECKPOINT_TIMEOUT

Starting with release 8.1.5, the `LOG_CHECKPOINT_TIMEOUT` initialization parameter has a new default value. In previous releases, the default value was zero seconds, but in release 8.1.5 and higher, the default value is 1800 seconds. See the *Oracle9i Database Reference* for more information.

The O7_DICTIONARY_ACCESSIBILITY Parameter

The `O7_DICTIONARY_ACCESSIBILITY` initialization parameter controls whether to continue Oracle7 data dictionary behavior. Use of this initialization parameter is only a temporary expedient. Starting with release 9.0.1, the default value of this initialization parameter is `false`.

See Also: "Data Dictionary Protection" on page 9-30 for more information.

The DML_LOCKS Parameter

Oracle9*i* systems typically consume more DML locks while performing DDL operations than are required for Oracle7 systems. Nevertheless, the Oracle7 `DML_LOCKS` parameter default settings are usually adequate for Oracle9*i* systems, even for DML-intensive applications.

The default value of `DML_LOCKS` is a multiple of the number of transactions, which is calculated from the number of rollback segments. However, in Oracle9*i* fewer transactions are used per rollback segment than are used in Oracle7. Consequently, `DML_LOCKS` has a lower default value in Oracle9*i*. Under some extreme load conditions, you may need to increase the `DML_LOCKS` parameter value.

You may also need to adjust the `TRANSACTION_PER_ROLLBACK_SEGMENT` parameter setting, depending on the operating system-specific settings. An informational message about this change may be displayed during database startup operations.

The DB_DOMAIN Parameter

Starting with release 8.1, if the `DB_DOMAIN` initialization parameter is unset, then it is set to NULL by default. In prior releases of Oracle, the default setting was the following:

```
.WORLD
```

A NULL setting for DB_DOMAIN may cause database connection problems in some environments. Before you migrate or upgrade to release 8.1 or higher, make sure the DB_DOMAIN initialization parameter in your initialization parameter file is set to one of the following:

- .WORLD
- a valid domain setting for your environment

If DB_DOMAIN is not set in your current database, then set it to .WORLD before you migrate or upgrade to release 8.1 or higher.

If DB_DOMAIN is set to a valid domain for your environment in your current database, then retain the setting in your initialization parameter file when you migrate or upgrade to release 8.1 or higher.

Parallel Execution Allocated from Large Pool

Starting with release 8.1, parallel execution message buffers are allocated from the large pool whenever PARALLEL_AUTOMATIC_TUNING is set to true. In past releases, this allocation was from the shared pool. If you are migrating or upgrading to release 8.1 or higher and you choose to set PARALLEL_AUTOMATIC_TUNING to true, then you can avoid problems by modifying the settings for the following initialization parameters:

- SHARED_POOL_SIZE
- LARGE_POOL_SIZE

Typically, you should reduce the setting of SHARED_POOL_SIZE and raise the setting of LARGE_POOL_SIZE to avoid problems. Alternatively, you can reduce the setting of SHARED_POOL_SIZE and let Oracle calculate the setting of LARGE_POOL_SIZE. Oracle calculates a default LARGE_POOL_SIZE only if PARALLEL_AUTOMATIC_TUNING is set to true and LARGE_POOL_SIZE is unset.

The calculation is based on the settings of the following initialization parameters:

- PARALLEL_MAX_SERVERS
- PARALLEL_THREADS_PER_CPU
- PARALLEL_SERVER_INSTANCES
- MTS_DISPATCHERS
- DBWR_IO_SLAVES

If PARALLEL_AUTOMATIC_TUNING is unset or set to FALSE, and if LARGE_POOL_SIZE is unset, then the value of LARGE_POOL_SIZE defaults to zero.

Note: When PARALLEL_AUTOMATIC_TUNING is set to true, the new behavior applies even if your COMPATIBLE parameter is set below 8.1.0.

See Also: *Oracle9i Database Reference* and *Oracle9i Database Performance Guide and Reference* for more information about other effects of the PARALLEL_AUTOMATIC_TUNING initialization parameter.

The following scenarios illustrate the behavior that results from various initialization parameter settings when you migrate or upgrade to release 8.1 or higher.

Retaining Parameter Settings without Modifications

You do not alter the parameters from their previous settings:

Table B–3 Retaining Parameter Settings without Modifications

Parameter	Setting
PARALLEL_AUTOMATIC_TUNING	Unset (defaults to FALSE).
SHARED_POOL_SIZE	Set to a large value, including the space required for parallel execution.
LARGE_POOL_SIZE	Unset (defaults to zero).

These settings are the most common scenario. In this case, you already have accounted for the space required for parallel execution in the shared pool.

Using PARALLEL_AUTOMATIC_TUNING

You alter the parameters from their previous settings to the following settings:

Table B–4 Using PARALLEL_AUTOMATIC_TUNING

Parameter	Setting
`PARALLEL_AUTOMATIC_TUNING`	Set to `true`.
`SHARED_POOL_SIZE`	Set to a small value that accounts for all clients except parallel execution.
`LARGE_POOL_SIZE`	Unset (defaults to a large value that includes the space required for parallel execution).

In this case, parallel execution allocates buffers from the large pool based on Oracle's automatic calculation. Buffer allocation is more efficient, and failures to allocate are isolated from the clients of the shared pool.

Using PARALLEL_AUTOMATIC_TUNING and Setting LARGE_POOL_SIZE

You alter the parameters from their previous settings to the following settings:

Table B–5 Using PARALLEL_AUTOMATIC_TUNING and Setting LARGE_POOL_SIZE

Parameter	Setting
`PARALLEL_AUTOMATIC_TUNING`	Set to `true`.
`SHARED_POOL_SIZE`	Set to a small value that accounts for all clients except parallel execution.
`LARGE_POOL_SIZE`	Set to a value that includes the space required for parallel execution.

In this case, parallel execution allocates buffers from the large pool. After initial testing with `LARGE_POOL_SIZE` unset, you determined that the default calculation for `LARGE_POOL_SIZE` did not reflect your requirements for the large pool. Therefore, you decided to manually set `LARGE_POOL_SIZE`. After you set `LARGE_POOL_SIZE` properly, buffer allocation is more efficient, and failures to allocate are isolated from the clients of the shared pool.

Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE

You alter the parameters from their previous settings to the following settings:

Table B–6 Using PARALLEL_AUTOMATIC_TUNING without Modifying SHARED_POOL_SIZE

Parameter	Setting
`PARALLEL_AUTOMATIC_TUNING`	Set to `true`.
`SHARED_POOL_SIZE`	Set to a large value, including the space required for parallel execution.
`LARGE_POOL_SIZE`	Unset (defaults to a large value that includes the space required for parallel execution).

In this case, parallel execution allocates buffers from the large pool, but because you did not modify `SHARED_POOL_SIZE`, it is likely that the SGA will be unnecessarily large, causing performance problems. Therefore, avoid setting `PARALLEL_AUTOMATIC_TUNING` to `true` without modifying the settings of `SHARED_POOL_SIZE` and `LARGE_POOL_SIZE` appropriately.

Archive Log Destination Parameters

Release 8.1 and higher supports new archive log destination parameters. After you migrate or upgrade to release 8.1 or higher, you can dynamically convert from the old pre-release 8.1 parameters (`LOG_ARCHIVE_DEST` and `LOG_ARCHIVE_DUPLEX_DEST`) to the new release 8.1 and higher parameters (`LOG_ARCHIVE_DEST_`*n* and `LOG_ARCHIVE_DEST_STATE_`*n*). You can also dynamically revert to the old parameters.

Note: In Oracle9*i*, the number of archive log destinations was increased from 5 to 10.

Changing to the New Archive Log Destination Parameters

After you determine the new archive destinations, associated states, and options, complete the following steps to change from the old archive log destination parameters to the new ones:

1. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_MIN_SUCCEED_DEST` to 1.
2. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_DUPLEX_DEST` to NULL.

3. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_DEST` to NULL.
4. Use `ALTER SYSTEM` to set any `LOG_ARCHIVE_DEST_STATE_`*n* parameters to "defer" or "enable" as required. Although enable is the default, Oracle Corporation recommends that you explicitly set a state for each destination.
5. Use `ALTER SYSTEM` to set at least one `LOG_ARCHIVE_DEST_`*n* parameter to a value specifying a local destination.
6. Use `ALTER SYSTEM` to set other `LOG_ARCHIVE_DEST_`*n* parameters as required.
7. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_MIN_SUCCEED_DEST` to the required value.

For example, assume there are the following two destinations:

- `/oracle/dbs/arclog`
- `/backup/dbs/arclog`

Both destinations are mandatory (minimum succeed destination count is 2). The new destinations are the following:

- `/oracle/dbs/arclog` (local)
- `stndby1` (a standby database)
- `/backup/dbs/arclog`
- `/backup2/dbs/arclog`

The first destination, the standby destination, and either of the backup destinations are mandatory (minimum succeed destination count is 3).

With these assumptions, issue the following SQL statements to change your old archive log destination parameters to the new ones:

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 1;

ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = ' ';

ALTER SYSTEM SET LOG_ARCHIVE_DEST = ' ';

ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_1 = 'enable';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_2 = 'enable';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_3 = 'enable';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_STATE_4 = 'enable';

ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = 'LOCATION=/oracle/dbs/arclog MANDATORY';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_2 = 'SERVICE=stndby1 MANDATORY';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_3 = 'LOCATION=/backup/dbs/arclog OPTIONAL';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = 'LOCATION=/backup2/dbs/arclog OPTIONAL';
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 3;
```

Changing Back to the Old Archive Log Destination Parameters

Complete the following steps to change back to the old archive log destination parameters:

1. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_MIN_SUCCEED_DEST` to 1.
2. Use `ALTER SYSTEM` to set all `LOG_ARCHIVE_DEST_`*n* parameters to NULL.
3. Use `ALTER SYSTEM` to set the `LOG_ARCHIVE_DEST` parameter to a value specifying a local destination.
4. Use `ALTER SYSTEM` to set the `LOG_ARCHIVE_DUPLEX_DEST` parameter as required.
5. Use `ALTER SYSTEM` to set `LOG_ARCHIVE_MIN_SUCCEED_DEST` to the required value.

For example, assume there are the following two destinations:

- `/oracle/dbs/arclog` (LOG_ARCHIVE_DEST_1)
- `/backup/dbs/arclog` (LOG_ARCHIVE_DEST_4)

Both destinations are mandatory. The new destinations and minimum succeed count are the same.

With these assumptions, issue the following SQL statements to change your new archive log destination parameters to the old ones:

```
ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 1;

ALTER SYSTEM SET LOG_ARCHIVE_DEST_4 = ' ';
ALTER SYSTEM SET LOG_ARCHIVE_DEST_1 = ' ';

ALTER SYSTEM SET LOG_ARCHIVE_DEST = '/oracle/dbs/arclog';

ALTER SYSTEM SET LOG_ARCHIVE_DUPLEX_DEST = '/backup/dbs/arclog';

ALTER SYSTEM SET LOG_ARCHIVE_MIN_SUCCEED_DEST = 2;
```

Possible Errors During the Transition in Parameters

When you follow the procedures described previously in this section for changing your archive destination parameters, you may encounter the following error messages in your log files if archiving is enabled:

- In the Alert log - "Archiving not possible: No available destinations"
- In the Trace log - "ARCH: INCOMPLETE, no available destinations"

You will not encounter these errors if archiving is disabled. The errors may occur during the procedure when there are no valid archive destinations. However, when the transition in parameters is complete, the errors should cease. You should *not* disable archiving during the transition to avoid these errors.

C
Changes to Static Data Dictionary Views

Oracle9*i* supports new static data dictionary views, and some static data dictionary views have been changed or have become obsolete in Oracle9*i*. This appendix lists the new, changed, and obsolete static data dictionary views in version 8 and Oracle9*i*.

This appendix covers the following topics:

- Changes to Static Data Dictionary Views in Oracle9i
- Changes to Static Data Dictionary Views in Version 8

> **See Also:** *Oracle9i Database Reference* for detailed information about the static data dictionary views in Oracle9*i*, and for descriptions of the columns in these views.

Changes to Static Data Dictionary Views in Oracle9*i*

The following sections list the new, changed, and obsolete static data dictionary views in Oracle9*i*:

- Static Data Dictionary Views Added in Oracle9i
- Static Data Dictionary Views Renamed in Oracle9i
- Static Data Dictionary Views with Added Columns in Oracle9i

Static Data Dictionary Views Added in Oracle9*i*

The static data dictionary views listed in this section are new in Oracle9*i*.

Static Data Dictionary Views Added in Release 9.0.1

The following static data dictionary views were added in release 9.0.1:

ALL_AUDIT_POLICIES	ALL_BASE_TABLE_MVIEWS
ALL_EXTERNAL_LOCATIONS	ALL_EXTERNAL_TABLES
ALL_INDEXTYPE_COMMENTS	ALL_JOIN_IND_COLUMNS
ALL_LOG_GROUP_COLUMNS	ALL_LOG_GROUPS
ALL_MVIEW_LOGS	ALL_OPERATOR_COMMENTS
ALL_PENDING_CONV_TABLES	ALL_POLICY_CONTEXTS
ALL_POLICY_GROUPS	ALL_PROCEDURES
ALL_SECONDARY_OBJECTS	ALL_SOURCE_TAB_COLUMNS
ALL_SOURCE_TABLES	ALL_SQLJ_TYPE_ATTRS
ALL_SQLJ_TYPE_METHODS	ALL_SQLJ_TYPES
ALL_STORED_SETTINGS	ALL_SUBSCRIBED_COLUMNS
ALL_SUBSCRIBED_TABLES	ALL_SUBSCRIPTIONS
ALL_TYPE_VERSIONS	DBA_APPLICATION_ROLES
DBA_ATTRIBUTE_TRANSFORMATIONS	DBA_AUDIT_POLICIES
DBA_BASE_TABLE_MVIEWS	DBA_EXTERNAL_LOCATIONS
DBA_EXTERNAL_TABLES	DBA_FGA_AUDIT_TRAIL
DBA_GLOBAL_CONTEXT	DBA_INDEXTYPE_COMMENTS
DBA_JOIN_IND_COLUMNS	DBA_LOG_GROUP_COLUMNS

DBA_LOG_GROUPS	DBA_MVIEW_LOGS
DBA_OPERATOR_COMMENTS	DBA_PENDING_CONV_TABLES
DBA_POLICY_CONTEXTS	DBA_POLICY_GROUPS
DBA_PROCEDURES	DBA_PROXIES
DBA_REPCAT_EXCEPTIONS	DBA_REPEXTENSIONS
DBA_REPSITES_NEW	DBA_RESUMABLE
DBA_SECONDARY_OBJECTS	DBA_SNAPSHOT_LOG_FILTER_COLS
DBA_SOURCE_TAB_COLUMNS	DBA_SOURCE_TABLES
DBA_SQLJ_TYPE_ATTRS	DBA_SQLJ_TYPE_METHODS
DBA_SQLJ_TYPES	DBA_STORED_SETTINGS
DBA_SUBSCRIBED_COLUMNS	DBA_SUBSCRIBED_TABLES
DBA_SUBSCRIPTIONS	DBA_TEMPLATE_REFGROUPS
DBA_TEMPLATE_TARGETS	DBA_TRANSFORMATIONS
DBA_TYPE_VERSIONS	DBA_UNDO_EXTENTS
USER_APPLICATION_ROLES	USER_ATTRIBUTE_TRANSFORMATIONS
USER_AUDIT_POLICIES	USER_BASE_TABLE_MVIEWS
USER_EXTERNAL_LOCATIONS	USER_EXTERNAL_TABLES
USER_INDEXTYPE_COMMENTS	USER_JOIN_IND_COLUMNS
USER_LOG_GROUP_COLUMNS	USER_LOG_GROUPS
USER_MVIEW_LOGS	USER_OPERATOR_COMMENTS
USER_PENDING_CONV_TABLES	USER_POLICY_CONTEXTS
USER_POLICY_GROUPS	USER_PROCEDURES
USER_PROXIES	USER_RESUMABLE
USER_SECONDARY_OBJECTS	USER_SOURCE_TAB_COLUMNS
USER_SOURCE_TABLES	USER_SQLJ_TYPE_ATTRS
USER_SQLJ_TYPE_METHODS	USER_SQLJ_TYPES
USER_STORED_SETTINGS	USER_SUBSCRIBED_COLUMNS
USER_SUBSCRIBED_TABLES	USER_SUBSCRIPTIONS
USER_TRANSFORMATIONS	USER_TYPE_VERSIONS

Static Data Dictionary Views Renamed in Oracle9*i*

The static data dictionary views listed in this section have been renamed in Oracle9*i*. If an application requires one or more of the views listed below, then modify the application accordingly.

Static Data Dictionary Views Renamed in Release 9.0.1

The following static data dictionary views were renamed in release 9.0.1:

Note: The old names are maintained for backward compatibility with version 8. Oracle Corporation recommends migrating to the new names.

Table C–1 Static Data Dictionary Views Renamed in Release 9.0.1

Pre-Release 9.0.1 Name	Release 9.0.1 Name
ALL_REGISTERED_SNAPSHOTS	ALL_REGISTERED_MVIEWS
ALL_SNAPSHOT_REFRESH_TIMES	ALL_MVIEW_REFRESH_TIMES
DBA_REGISTERED_SNAPSHOT_GROUPS	DBA_REGISTERED_MVIEW_GROUPS
DBA_REGISTERED_SNAPSHOTS	DBA_REGISTERED_MVIEWS
DBA_SNAPSHOT_LOG_FILTER_COLS	DBA_MVIEW_LOG_FILTER_COLS
DBA_SNAPSHOT_REFRESH_TIMES	DBA_MVIEW_REFRESH_TIMES
USER_SNAPSHOT_REFRESH_TIMES	USER_MVIEW_REFRESH_TIMES
USER_REGISTERED_SNAPSHOTS	USER_REGISTERED_MVIEWS

Static Data Dictionary Views with Added Columns in Oracle9*i*

New columns were added to the static data dictionary views listed in the following sections.

See Also: *Oracle9i Database Reference* for detailed information about these views

Static Data Dictionary Views with Added Columns in Release 9.0.1

New columns were added to the following static data dictionary views in release 9.0.1:

ALL_ALL_TABLES	ALL_ARGUMENTS
ALL_ASSOCIATIONS	ALL_CLUSTERS
ALL_CONSTRAINTS	ALL_DIMENSIONS
ALL_IND_COLUMNS	ALL_IND_PARTITIONS
ALL_INDEXES	ALL_INDEXTYPE_OPERATORS
ALL_INDEXTYPES	ALL_MVIEWS
ALL_OBJECT_TABLES	ALL_OPBINDINGS
ALL_OUTLINES	ALL_PART_INDEXES
ALL_POLICIES	ALL_REPCAT_TEMPLATE_OBJECTS
ALL_REPCAT_TEMPLATE_SITES	ALL_REPCOLUMN
ALL_REPDDL	ALL_REPFLAVOR_COLUMNS
ALL_REPOBJECT	ALL_SNAPSHOT_LOGS
ALL_SUMDELTA	ALL_TAB_COLUMNS
ALL_TAB_PRIVS	ALL_TAB_PRIVS_MADE
ALL_TAB_PRIVS_RECD	ALL_TABLES
ALL_TYPE_ATTRS	ALL_TYPE_METHODS
ALL_TYPES	ALL_USTATS
ALL_VIEWS	CLU
COLS	DBA_ALL_TABLES
DBA_ASSOCIATIONS	DBA_AUDIT_EXISTS
DBA_AUDIT_OBJECT	DBA_AUDIT_SESSION
DBA_AUDIT_STATEMENT	DBA_AUDIT_TRAIL
DBA_CLUSTERS	DBA_CONSTRAINTS
DBA_CONTEXT	DBA_IND_COLUMNS
DBA_IND_PARTITIONS	DBA_INDEXES
DBA_INDEXTYPE_OPERATORS	DBA_INDEXTYPES
DBA_MVIEWS	DBA_OBJECT_TABLES
DBA_OPBINDINGS	DBA_OUTLINES
DBA_PART_INDEXES	DBA_POLICIES
DBA_REPCAT_TEMPLATE_OBJECTS	DBA_REPCAT_TEMPLATE_SITES

DBA_REPCOLUMN	DBA_REPDDL
DBA_REPFLAVOR_COLUMNS	DBA_REPOBJECT
DBA_RSRC_PLANS	DBA_SNAPSHOT_LOGS
DBA_TAB_COLUMNS	DBA_TAB_PRIVS
DBA_TABLES	DBA_TABLESPACES
DBA_TYPE_ATTRS	DBA_TYPE_METHODS
DBA_TYPES	DBA_USTATS
DBA_VIEWS	IND
TABS	USER_ALL_TABLES
USER_ARGUMENTS	USER_ASSOCIATIONS
USER_AUDIT_OBJECT	USER_AUDIT_SESSION
USER_AUDIT_STATEMENT	USER_AUDIT_TRAIL
USER_CLUSTERS	USER_CONSTRAINTS
USER_DIMENSIONS	USER_IND_COLUMNS
USER_IND_PARTITIONS	USER_INDEXES
USER_INDEXTYPE_OPERATORS	USER_INDEXTYPES
USER_MVIEWS	USER_OBJECT_TABLES
USER_OPBINDINGS	USER_OUTLINES
USER_PART_INDEXES	USER_POLICIES
USER_REPCAT_TEMPLATE_OBJECTS	USER_REPCAT_TEMPLATE_SITES
USER_REPCOLUMN	USER_REPDDL
USER_REPFLAVOR_COLUMNS	USER_REPOBJECT
USER_REPPARAMETER_COLUMN	USER_SNAPSHOT_LOGS
USER_TAB_COLUMNS	USER_TAB_PRIVS
USER_TAB_PRIVS_MADE	USER_TAB_PRIVS_RECD
USER_TABLES	USER_TABLESPACES
USER_TYPE_ATTRS	USER_TYPE_METHODS
USER_TYPES	USER_USTATS
USER_VIEWS	

Changes to Static Data Dictionary Views in Version 8

The following sections list the new, changed, and obsolete static data dictionary views in version 8:

- Static Data Dictionary Views Added in Version 8
- Static Data Dictionary Views with Added Columns in Version 8
- Static Data Dictionary Views with Dropped Columns in Version 8
- Static Data Dictionary Views with Renamed Columns in Version 8
- Static Data Dictionary Views with Columns That May Return Nulls
- Static Data Dictionary Views Obsolete in Version 8

Static Data Dictionary Views Added in Version 8

The static data dictionary views listed in this section are new in version 8.

Static Data Dictionary Views Added in Release 8.0

The following static data dictionary views were added in release 8.0:

ALL_ALL_TABLES	ALL_COLL_TYPES
ALL_DIRECTORIES	ALL_IND_PARTITIONS
ALL_LIBRARIES	ALL_LOBS
ALL_METHOD_PARAMS	ALL_METHOD_RESULTS
ALL_NESTED_TABLES	ALL_OBJECT_TABLES
ALL_OBJECTS	ALL_PART_COL_STATISTICS
ALL_PART_HISTOGRAMS	ALL_PART_INDEXES
ALL_PART_KEY_COLUMNS	ALL_PART_TABLES
ALL_PROBE_OBJECTS	ALL_REFS
ALL_REGISTERED_SNAPSHOTS	ALL_REPCOLUMN
ALL_REPGENOBJECTS	ALL_SNAPSHOT_LOGS
ALL_SNAPSHOT_REFRESH_TIMES	ALL_TAB_COL_STATISTICS
ALL_TAB_HISTOGRAMS	ALL_TAB_PARTITIONS
ALL_TYPE_ATTRS	ALL_TYPE_METHODS
ALL_TYPES	DBA_ALL_TABLES

DBA_COLL_TYPES	DBA_DIRECTORIES
DBA_IND_PARTITIONS	DBA_LIBRARIES
DBA_LOBS	DBA_METHOD_PARAMS
DBA_METHOD_RESULTS	DBA_NESTED_TABLES
DBA_OBJECT_TABLES	DBA_PART_COL_STATISTICS
DBA_PART_HISTOGRAMS	DBA_PART_INDEXES
DBA_PART_KEY_COLUMNS	DBA_PART_TABLES
DBA_PENDING_TRANSACTIONS	DBA_QUEUE_SCHEDULES
DBA_QUEUE_TABLES	DBA_QUEUES
DBA_REFS	DBA_REGISTERED_SNAPSHOT_GROUPS
DBA_REGISTERED_SNAPSHOTS	DBA_REPCOLUMN
DBA_REPGENOBJECTS	DBA_SNAPSHOT_LOG_FILTER_COLS
DBA_SNAPSHOT_REFRESH_TIMES	DBA_TAB_COL_STATISTICS
DBA_TAB_HISTOGRAMS	DBA_TAB_PARTITIONS
DBA_TYPE_ATTRS	DBA_TYPE_METHODS
DBA_TYPES	DEFLOB
DEFPROPAGATOR	FILEXT$
HS_ALL_CAPS	HS_ALL_DD
HS_ALL_INITS	HS_BASE_CAPS
HS_BASE_DD	HS_CLASS_CAPS
HS_CLASS_DD	HS_CLASS_INIT
HS_EXTERNAL_OBJECT_PRIVILEGES	HS_EXTERNAL_OBJECTS
HS_EXTERNAL_USER_PRIVILEGES	HS_FDS_CLASS
HS_FDS_INST	HS_INST_CAPS
HS_INST_DD	HS_INST_INIT
TRUSTED_SERVERS	TS_PITR_CHECK
TS_PITR_OBJECTS_TO_BE_DROPPED	USER_ALL_TABLES
USER_COLL_TYPES	USER_IND_PARTITIONS
USER_LIBRARIES	USER_LOBS
USER_METHOD_PARAMS	USER_METHOD_RESULTS
USER_NESTED_TABLES	USER_OBJECT_TABLES

USER_PART_COL_STATISTICS	USER_PART_HISTOGRAMS
USER_PART_INDEXES	USER_PART_KEY_COLUMNS
USER_PART_TABLES	USER_PASSWORD_LIMITS
USER_QUEUE_TABLES	USER_QUEUES
USER_REFS	USER_REGISTERED_SNAPSHOTS
USER_REPCOLUMN	USER_REPGENOBJECTS
USER_SNAPSHOT_REFRESH_TIMES	USER_TAB_COL_STATISTICS
USER_TAB_HISTOGRAMS	USER_TAB_PARTITIONS
USER_TYPE_ATTRS	USER_TYPE_METHODS
USER_TYPES	

Static Data Dictionary Views Added in Release 8.1

The following static data dictionary views were added in release 8.1:

ALL_ASSOCIATIONS	ALL_CONTEXT
ALL_DIM_ATTRIBUTES	ALL_DIM_CHILD_OF
ALL_DIM_HIERARCHIES	ALL_DIM_JOIN_KEY
ALL_DIM_LEVEL_KEY	ALL_DIM_LEVELS
ALL_DIMENSIONS	ALL_IND_EXPRESSIONS
ALL_IND_SUBPARTITIONS	ALL_INDEXTYPE_OPERATORS
ALL_INDEXTYPES	ALL_INTERNAL_TRIGGERS
ALL_LOB_PARTITIONS	ALL_LOB_SUBPARTITIONS
ALL_MVIEW_AGGREGATES	ALL_MVIEW_ANALYSIS
ALL_MVIEW_DETAIL_RELATIONS	ALL_MVIEW_JOINS
ALL_MVIEW_KEYS	ALL_MVIEWS
ALL_OPANCILLARY	ALL_OPARGUMENTS
ALL_OPBINDINGS	ALL_OPERATORS
ALL_PART_LOBS	ALL_PARTIAL_DROP_TABS
ALL_POLICIES	ALL_QUEUE_TABLES
ALL_QUEUES	ALL_REFRESH_DEPENDENCIES
ALL_REPCAT_REFRESH_TEMPLATES	ALL_REPCAT_TEMPLATE_OBJECTS

ALL_REPCAT_TEMPLATE_PARMS	ALL_REPCAT_TEMPLATE_SITES
ALL_REPCAT_USER_AUTHORIZATIONS	ALL_REPCAT_USER_PARM_VALUES
ALL_SUBPART_COL_STATISTICS	ALL_SUBPART_HISTOGRAMS
ALL_SUBPART_KEY_COLUMNS	ALL_SUMDELTA
ALL_SUMMARIES	ALL_SUMMARY_AGGREGATES
ALL_SUMMARY_DETAIL_TABLES	ALL_SUMMARY_JOINS
ALL_SUMMARY_KEYS	ALL_TAB_MODIFICATIONS
ALL_TAB_SUBPARTITIONS	ALL_UNUSED_COL_TABS
ALL_USTATS	ALL_VARRAYS
DATABASE_COMPATIBLE_LEVEL	DBA_ASSOCIATIONS
DBA_CONTEXT	DBA_DIM_ATTRIBUTES
DBA_DIM_CHILD_OF	DBA_DIM_HIERARCHIES
DBA_DIM_JOIN_KEY	DBA_DIM_LEVEL_KEY
DBA_DIM_LEVELS	DBA_DIMENSIONS
DBA_DMT_FREE_SPACE	DBA_DMT_USED_EXTENTS
DBA_IND_EXPRESSIONS	DBA_IND_SUBPARTITIONS
DBA_INDEXTYPE_OPERATORS	DBA_INDEXTYPES
DBA_INTERNAL_TRIGGERS	DBA_LMT_FREE_SPACE
DBA_LMT_USED_EXTENTS	DBA_LOB_PARTITIONS
DBA_LOB_SUBPARTITIONS	DBA_MVIEW_AGGREGATES
DBA_MVIEW_ANALYSIS	DBA_MVIEW_DETAIL_RELATIONS
DBA_MVIEW_JOINS	DBA_MVIEW_KEYS
DBA_MVIEWS	DBA_OPANCILLARY
DBA_OPARGUMENTS	DBA_OPBINDINGS
DBA_OPERATORS	DBA_OUTLINE_HINTS
DBA_OUTLINES	DBA_PART_LOBS
DBA_POLICIES	DBA_REPCAT_REFRESH_TEMPLATES
DBA_REPCAT_TEMPLATE_OBJECTS	DBA_REPCAT_TEMPLATE_SITES
DBA_REPCAT_USER_AUTHORIZATIONS	DBA_REPCAT_USER_PARM_VALUES
DBA_RSRC_CONSUMER_GROUP_PRIVS	DBA_RSRC_CONSUMER_GROUPS
DBA_RSRC_MANAGER_SYSTEM_PRIVS	DBA_RSRC_PLAN_DIRECTIVES

DBA_RSRC_PLANS	DBA_RULESETS
DBA_SUBPART_HISTOGRAMS	DBA_SUBPART_KEY_COLUMNS
DBA_SUMMARIES	DBA_SUMMARY_DETAIL_TABLES
DBA_TAB_MODIFICATIONS	DBA_TAB_SUBPARTITIONS
DBA_TEMP_FILES	DBA_UNUSED_COL_TABS
DBA_USTATS	DBA_VARRAYS
DBMS_ALERT_INFO	DBMS_LOCK_ALLOCATED
PLUGGABLE_SET_CHECK	PROXY_USERS
QUEUE_PRIVILEGES	SESSION_CONTEXT
STRADDLING_TS_OBJECTS	UNI_PLUGGABLE_SET_CHECK
USER_ASSOCIATIONS	USER_DIM_ATTRIBUTES
USER_DIM_CHILD_OF	USER_DIM_HIERARCHIES
USER_DIM_JOIN_KEY	USER_DIM_LEVEL_KEY
USER_DIM_LEVELS	USER_DIMENSIONS
USER_IND_EXPRESSIONS	USER_IND_SUBPARTITIONS
USER_INDEXTYPE_OPERATORS	USER_INDEXTYPES
USER_INTERNAL_TRIGGERS	USER_LOB_PARTITIONS
USER_LOB_SUBPARTITIONS	USER_MVIEW_AGGREGATES
USER_MVIEW_ANALYSIS	USER_MVIEW_DETAIL_RELATIONS
USER_MVIEW_JOINS	USER_MVIEW_KEYS
USER_MVIEWS	USER_OPANCILLARY
USER_OPARGUMENTS	USER_OPBINDINGS
USER_OPERATORS	USER_OUTLINE_HINTS
USER_OUTLINES	USER_PART_LOBS
USER_PARTIAL_DROP_TABS	USER_POLICIES
USER_QUEUE_SCHEDULES	USER_REPCAT_REFRESH_TEMPLATES
USER_REPCAT_TEMPLATE_OBJECTS	USER_REPCAT_TEMPLATE_PARMS
USER_REPCAT_TEMPLATE_SITES	USER_REPCAT_USER_AUTHORIZATION
USER_REPCAT_USER_PARM_VALUES	USER_RSRC_CONSUMER_GROUP_PRIVS
USER_RSRC_MANAGER_SYSTEM_PRIVS	USER_RULESETS
USER_SUBPART_COL_STATISTICS	USER_SUBPART_HISTOGRAMS

USER_SUBPART_KEY_COLUMNS	USER_SUMMARIES
USER_SUMMARY_DETAIL_TABLES	USER_TAB_MODIFICATIONS
USER_TAB_SUBPARTITIONS	USER_UNUSED_COL_TABS
USER_USTATS	USER_VARRAYS

Static Data Dictionary Views with Added Columns in Version 8

New columns were added to the static data dictionary views listed in the following sections.

> **See Also:** *Oracle9i Database Reference* for detailed information about these views

Static Data Dictionary Views with Added Columns in Release 8.0

New columns were added to the following static data dictionary views in release 8.0:

ALL_ARGUMENTS	ALL_CLUSTERS
ALL_CONSTRAINTS	ALL_DEPENDENCIES
ALL_INDEXES	ALL_OBJECTS
ALL_REFRESH	ALL_REFRESH_CHILDREN
ALL_REPOBJECT	ALL_REPPRIORITY
ALL_REPSCHEMA	ALL_REPSITES
ALL_SNAPSHOTS	ALL_TAB_COLUMNS
ALL_TABLES	ALL_UPDATABLE_COLUMNS
ALL_VIEWS	COL
DBA_CLUSTERS	DBA_CONSTRAINTS
DBA_DATA_FILES	DBA_DEPENDENCIES
DBA_EXTENTS	DBA_FREE_SPACE
DBA_INDEXES	DBA_OBJ_AUDIT_OPTS
DBA_OBJECTS	DBA_PROFILES
DBA_REFRESH	DBA_REFRESH_CHILDREN
DBA_REPOBJECT	DBA_REPPRIORITY
DBA_RGROUP	DBA_ROLLBACK_SEGS

DBA_SEGMENTS	DBA_SNAPSHOT_LOGS
DBA_SNAPSHOTS	DBA_TAB_COLUMNS
DBA_TABLES	DBA_TABLESPACES
DBA_UPDATABLE_COLUMNS	DBA_USERS
DBA_VIEWS	DEFERROR
DEFTRANDEST	FILE_PING
INDEX_STATS	SYS_OBJECTS
USER_ARGUMENTS	USER_CLUSTERS
USER_CONSTRAINTS	USER_DEPENDENCIES
USER_EXTENTS	USER_FREE_SPACE
USER_INDEXES	USER_OBJ_AUDIT_OPTS
USER_OBJECTS	USER_REFRESH
USER_REFRESH_CHILDREN	USER_REPOBJECT
USER_REPPRIORITY	USER_REPSCHEMA
USER_REPSITES	USER_SEGMENTS
USER_SNAPSHOT_LOGS	USER_SNAPSHOTS
USER_TAB_COLUMNS	USER_TABLES
USER_TABLESPACES	USER_UPDATABLE_COLUMNS
USER_USERS	USER_VIEWS

Static Data Dictionary Views with Added Columns in Release 8.1

New columns were added to the following static data dictionary views in release 8.1:

ALL_ALL_TABLES	ALL_ARGUMENTS
ALL_CLUSTERS	ALL_COLL_TYPES
ALL_CONSTRAINTS	ALL_IND_COLUMNS
ALL_IND_PARTITIONS	ALL_IND_SUBPARTITIONS
ALL_INDEXES	ALL_MVIEW_ANALYSIS
ALL_NESTED_TABLES	ALL_OBJECT_TABLES
ALL_OBJECTS	ALL_PART_COL_STATISTICS

ALL_PART_HISTOGRAMS	ALL_PART_INDEXES
ALL_PART_TABLES	ALL_PROBE_OBJECTS
ALL_REFS	ALL_REPCAT
ALL_REPCAT_TEMPLATE_SITES	ALL_REPCOLUMN
ALL_REPGROUP	ALL_REPOBJECT
ALL_SNAPSHOTS	ALL_SUBPART_HISTOGRAMS
ALL_TAB_COL_STATISTICS	ALL_TAB_COLUMNS
ALL_TAB_HISTOGRAMS	ALL_TAB_PARTITIONS
ALL_TABLES	ALL_TRIGGERS
DBA_ALL_TABLES	DBA_CLUSTERS
DBA_COLL_TYPES	DBA_CONSTRAINTS
DBA_CONTEXT	DBA_DATA_FILES
DBA_IND_COLUMNS	DBA_IND_PARTITIONS
DBA_IND_SUBPARTITIONS	DBA_INDEXES
DBA_JOBS	DBA_JOBS_RUNNING
DBA_MVIEW_ANALYSIS	DBA_NESTED_TABLES
DBA_OBJECT_TABLES	DBA_OBJECTS
DBA_OUTLINE_HINTS	DBA_PART_COL_STATISTICS
DBA_PART_HISTOGRAMS	DBA_PART_INDEXES
DBA_PART_TABLES	DBA_PRIV_AUDIT_OPTS
DBA_QUEUE_SCHEDULES	DBA_QUEUE_TABLES
DBA_REFS	DBA_REGISTERED_SNAPSHOT_GROUPS
DBA_REPCAT	DBA_REPCAT_TEMPLATE_SITES
DBA_REPCOLUMN	DBA_REPGROUP
DBA_REPOBJECT	DBA_SNAPSHOTS
DBA_STMT_AUDIT_OPTS	DBA_SUBPART_HISTOGRAMS
DBA_TAB_COL_STATISTICS	DBA_TAB_COLUMNS
DBA_TAB_HISTOGRAMS	DBA_TAB_PARTITIONS
DBA_TABLES	DBA_TABLESPACES
DBA_TRIGGERS	DBA_USERS
INDEX_STATS	REPCAT_REPCAT

REPCAT_REPOBJECTS	USER_ALL_TABLES
USER_ARGUMENTS	USER_CLUSTERS
USER_COLL_TYPES	USER_CONSTRAINTS
USER_IND_COLUMNS	USER_IND_PARTITIONS
USER_IND_SUBPARTITIONS	USER_INDEXES
USER_JOBS	USER_MVIEW_ANALYSIS
USER_NESTED_TABLES	USER_OBJECT_TABLES
USER_OBJECTS	USER_OUTLINE_HINTS
USER_PART_COL_STATISTICS	USER_PART_HISTOGRAMS
USER_PART_INDEXES	USER_PART_TABLES
USER_QUEUE_TABLES	USER_REFS
USER_REPCAT	USER_REPCAT_TEMPLATE_SITES
USER_REPCOLUMN	USER_REPGROUP
USER_REPOBJECT	USER_SNAPSHOTS
USER_SUBPART_HISTOGRAMS	USER_TAB_COL_STATISTICS
USER_TAB_COLUMNS	USER_TAB_HISTOGRAMS
USER_TAB_PARTITIONS	USER_TABLES
USER_TABLESPACES	USER_TRIGGERS
USER_USERS	

Static Data Dictionary Views with Dropped Columns in Version 8

The columns listed in the following sections were dropped in version 8. If an application requires one or more of the columns listed below, then modify the application accordingly.

Static Data Dictionary Views with Dropped Columns in Release 8.0

The columns listed in Table C–2 were dropped in release 8.0.

Table C–2 Static Data Dictionary Views with Dropped Columns in Release 8.0

Static Data Dictionary Views	Dropped Columns
DEFCALLDEST	DEFERRED_TRAN_DB

Table C–2 (Cont.) Static Data Dictionary Views with Dropped Columns in Release 8.0

Static Data Dictionary Views	Dropped Columns
DEFERROR	DEFERRED_TRAN_DB ERROR_TIME
DEFTRAN	COMMIT_COMMENT DEFERRED_TRAN_DB DESTINATION_LIST ORIGIN_TRAN_DB ORIGIN_TRAN_ID ORIGIN_USER
DEFTRANDEST	DEFERRED_TRAN_DB

Static Data Dictionary Views with Dropped Columns in Release 8.1

The columns listed in Table C–3 were dropped in release 8.1.

Table C–3 Static Data Dictionary Views with Dropped Columns in Release 8.1

Static Data Dictionary Views	Dropped Columns
DBA_AUDIT_OBJECT USER_AUDIT_OBJECT	OBJECT_LABEL SESSION_LABEL
DBA_AUDIT_SESSION USER_AUDIT_SESSION	SESSION_LABEL
DBA_AUDIT_STATEMENT USER_AUDIT_STATEMENT	SESSION_LABEL
DBA_AUDIT_TRAIL USER_AUDIT_TRAIL	OBJECT_LABEL SESSION_LABEL
DBA_CONTEXT	ATTRIBUTE
ALL_IND_COLUMNS DBA_IND_COLUMNS USER_IND_COLUMNS	COLUMN_EXPRESSION
ALL_JOBS DBA_JOBS USER_JOBS	CLEARANCE_HI CLEARANCE_LO CURRENT_SESSION_LABEL

Table C–3 (Cont.) Static Data Dictionary Views with Dropped Columns in Release 8.1

Static Data Dictionary Views	Dropped Columns
ALL_REFS DBA_REFS USER_REFS	HAS_REFERENTIAL_CONS REFERENTIAL_CONS_NAME

Static Data Dictionary Views with Renamed Columns in Version 8

The columns listed in the following sections were renamed in version 8. If an application requires one or more of the columns listed below, then modify the application accordingly.

Static Data Dictionary Views with Renamed Columns in Release 8.0

The columns listed in Table C–4 were renamed in release 8.0.

Table C–4 Static Data Dictionary Views with Renamed Columns in Release 8.0

Static Data Dictionary View	Oracle7 Column Name	Release 8.0 Column Name
DBA_RCHILD	TYPE	TYPE#
DEFSCHEDULE	LAST_ERROR	LAST_ERROR_NUMBER
	LAST_MSG	LAST_ERROR_MESSAGE

Static Data Dictionary Views with Columns That May Return Nulls

Starting with release 8.1, the columns in the static data dictionary views listed in Table C–5 may return nulls; in previous releases, these columns could not return nulls. If an application requires non-null values for one or more of the columns listed below, then modify the application accordingly.

Table C–5 Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
DBA_DATA_FILES	AUTOEXTENSIBLE BLOCKS BYTES INCREMENT_BY MAXBLOCKS MAXBYTES	These columns return a null if the data file is offline and therefore not readable.

Table C–5 (Cont.) Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
ALL_IND_COLUMNS DBA_IND_COLUMNS USER_IND_COLUMNS	COLUMN_NAME	This column returns a null if an index is on a function instead of a column. In this case, there is no column to list.
ALL_IND_PARTITIONS DBA_IND_PARTITIONS USER_IND_PARTITIONS	INITIAL_EXTENT MAX_EXTENT MIN_EXTENT NEXT_EXTENT PCT_INCREASE	These columns return a null if the index is partitioned using a composite method and no default value was specified for the partition.
ALL_OBJECT_TABLES DBA_OBJECT_TABLES USER_OBJECT_TABLES	TABLESPACE_NAME	This column returns a null in if an object table is partitioned or if it is a temporary table.
ALL_SEGMENTS DBA_SEGMENTS USER_SEGMENTS	BLOCKS BYTES EXTENTS NEXT_EXTENT PCT_INCREASE	The BLOCKS, BYTES, and EXTENTS columns return a null if the segment header cannot be read because the file is offline or if there is some other corruption. The NEXT_EXTENT and PCT_INCREASE columns return a null if the tablespace storing the segment is locally managed and uses the AUTOALLOCATE option, because the system chooses the extent sizes, and the algorithm cannot be explained in terms of NEXT_EXTENT and PCT_INCREASE.
ALL_TAB_PARTITIONS DBA_TAB_PARTITIONS USER_TAB_PARTITIONS	INITIAL_EXTENT MAX_EXTENT MIN_EXTENT NEXT_EXTENT PCT_INCREASE	These columns return a null if the table is partitioned using a composite method and no default value was specified for the partition.
ALL_TABLESPACES DBA_TABLESPACES USER_TABLESPACES	NEXT_EXTENT PCT_INCREASE	These columns return a null if the tablespace is locally managed and uses the AUTOALLOCATE option, because the system chooses the extent sizes, and the algorithm cannot be explained in terms of NEXT_EXTENT and PCT_INCREASE.

Table C–5 (Cont.) Columns That May Return Nulls in Release 8.1

Static Data Dictionary Views	Columns	Explanation
ALL_TRIGGERS DBA_TRIGGERS USER_TRIGGERS	TABLE_NAME	This column returns a null if the trigger is a system trigger. In this case, the base object type of the trigger will be SCHEMA or DATABASE, instead of TABLE or VIEW.

Static Data Dictionary Views Obsolete in Version 8

The static data dictionary views in this section are obsolete in version 8.

Static Data Dictionary Views Obsolete in Release 8.0

The following static data dictionary views became obsolete in release 8.0 and are not available in release 8.0 and higher:

ALL_HISTOGRAMS	DBA_HISTOGRAMS
DEFCALL	USER_HISTOGRAMS

Static Data Dictionary Views Obsolete in Release 8.1

The following static data dictionary views became obsolete in release 8.1 and are not available in release 8.1 and higher:

ALL_LABELS

D

Changes to Dynamic Performance Views

Oracle9*i* supports new dynamic performance views (V$ views), and some dynamic performance views have been changed or have become obsolete in Oracle9*i*. This appendix lists the new, changed, and obsolete dynamic performance views in version 8 and Oracle9*i*.

This appendix covers the following topics:

- Changes to Dynamic Performance Views in Oracle9i
- Changes to Dynamic Performance Views in Version 8

See Also: *Oracle9i Database Reference* for detailed information about the dynamic performance views in Oracle9*i*, and for descriptions of the columns in these views.

Changes to Dynamic Performance Views in Oracle9*i*

The following sections list the new, changed, and obsolete dynamic performance views in Oracle9*i*:

- Dynamic Performance Views Added in Oracle9i
- Dynamic Performance Views Renamed in Oracle9i
- Dynamic Performance Views with Dropped Columns in Oracle9i
- Dynamic Performance Views Obsolete in Oracle9i

Dynamic Performance Views Added in Oracle9*i*

The dynamic performance views listed in this section are new in Oracle9*i*.

Dynamic Performance Views Added in Release 9.0.1

The following dynamic performance views were added in release 9.0.1:

GV$ACTIVE_SESS_POOL_MTH	GV$ARCHIVE_DEST_STATUS
GV$ARCHIVE_GAP	GV$ENQUEUE_STAT
GV$GCSHVMASTER_INFO	GV$GCSPFMASTER_INFO
GV$GLOBALCONTEXT	GV$HVMASTER_INFO
GV$LOGMNR_CALLBACK	GV$LOGMNR_LOGFILE
GV$LOGMNR_PROCESS	GV$LOGMNR_REGION
GV$LOGMNR_SESSION	GV$LOGMNR_TRANSACTION
GV$LOGSTDBY	GV$LOGSTDBY_STATS
GV$MANAGED_STANDBY	GV$MVREFRESH
GV$PGASTAT	GV$QUEUEING_MTH
GV$REPLPROP	GV$REPLQUEUE
GV$RESUMABLE	GV$RMAN_CONFIGURATION
GV$SPPARAMETER	GV$SQL_PLAN
GV$SQL_REDIRECTION	GV$SQL_WORKAREA
GV$SQL_WORKAREA_ACTIVE	GV$STANDBY_LOG
GV$TIMEZONE_NAMES	GV$UNDOSTAT
GV$VPD_POLICY	V$ACTIVE_SESS_POOL_MTH

V$ARCHIVE_DEST_STATUS
V$ARCHIVE_GAP
V$ENQUEUE_STAT
V$FILESTATXS
V$GCSHVMASTER_INFO
V$GCSPFMASTER_INFO
V$GLOBALCONTEXT
V$HVMASTER_INFO
V$LOGMNR_CALLBACK
V$LOGMNR_LOGFILE
V$LOGMNR_PROCESS
V$LOGMNR_REGION
V$LOGMNR_SESSION
V$LOGMNR_TRANSACTION
V$LOGSTDBY
V$LOGSTDBY_STATS
V$MANAGED_STANDBY
V$MVREFRESH
V$PGASTAT
V$QUEUEING_MTH
V$REPLPROP
V$REPLQUEUE
V$RESUMABLE
V$RMAN_CONFIGURATION
V$SPPARAMETER
V$SQLXS
V$SQL_PLAN
V$SQL_REDIRECTION
V$SQL_WORKAREA
V$SQL_WORKAREA_ACTIVE
V$STANDBY_LOG
V$TEMPSTATXS
V$TIMEZONE_NAMES
V$UNDOSTAT
V$VPD_POLICY

Dynamic Performance Views Renamed in Oracle9*i*

The dynamic performance views listed in this section have been renamed in Oracle9*i*. If an application requires one or more of the views listed below, then modify the application accordingly.

Dynamic Performance Views Renamed in Release 9.0.1

The following dynamic performance views were renamed in release 9.0.1:

Table D–1 Dynamic Performance Views Renamed in Release 9.0.1

Pre-Release 9.0.1 Name	Release 9.0.1 Name
GV$BSP	GV$CR_BLOCK_SERVER
GV$CLASS_PING	GV$CLASS_CACHE_TRANSFER
GV$DLM_ALL_LOCKS	GV$GES_ENQUEUE

Table D–1 (Cont.) Dynamic Performance Views Renamed in Release 9.0.1

Pre-Release 9.0.1 Name	Release 9.0.1 Name
GV$DLM_CONVERT_LOCAL	GV$GES_CONVERT_LOCAL
GV$DLM_CONVERT_REMOTE	GV$GES_CONVERT_REMOTE
GV$DLM_LATCH	GV$GES_LATCH
GV$DLM_LOCKS	GV$GES_BLOCKING_ENQUEUE
GV$DLM_MISC	GV$GES_STATISTICS
GV$DLM_RESS	GV$GES_RESOURCE
GV$DLM_TRAFFIC_CONTROLLER	GV$GES_TRAFFIC_CONTROLLER
GV$FILE_PING	GV$FILE_CACHE_TRANSFER
GV$LOCK_ELEMENT	GV$GC_ELEMENT
GV$LOCKS_WITH_COLLISIONS	GV$GC_ELEMENTS_WITH_COLLISIONS
GV$MTS	GV$SHARED_SERVER_MONITOR
GV$PING	GV$CACHE_TRANSFER
GV$TEMP_PING	GV$TEMP_CACHE_TRANSFER
V$BSP	V$CR_BLOCK_SERVER
V$CLASS_PING	V$CLASS_CACHE_TRANSFER
V$DLM_ALL_LOCKS	V$GES_ENQUEUE
V$DLM_CONVERT_LOCAL	V$GES_CONVERT_LOCAL
V$DLM_CONVERT_REMOTE	V$GES_CONVERT_REMOTE
V$DLM_LATCH	V$GES_LATCH
V$DLM_LOCKS	V$GES_BLOCKING_ENQUEUE
V$DLM_MISC	V$GES_STATISTICS
V$DLM_RESS	V$GES_RESOURCE
V$DLM_TRAFFIC_CONTROLLER	V$GES_TRAFFIC_CONTROLLER
V$FILE_PING	V$FILE_CACHE_TRANSFER
V$LOCK_ELEMENT	V$GC_ELEMENT
V$LOCKS_WITH_COLLISIONS	V$GC_ELEMENTS_WITH_COLLISIONS
V$MTS	V$SHARED_SERVER_MONITOR

Table D–1 (Cont.) Dynamic Performance Views Renamed in Release 9.0.1

Pre-Release 9.0.1 Name	Release 9.0.1 Name
V$PING	V$CACHE_TRANSFER
V$TEMP_PING	V$TEMP_CACHE_TRANSFER

Note: The old names are maintained for backward compatibility with version 8. Oracle Corporation recommends migrating to the new names.

Dynamic Performance Views with Dropped Columns in Oracle9*i*

The columns listed in the following sections were dropped in Oracle9*i*. If an application requires one or more of the columns listed below, then modify the application accordingly.

Dynamic Performance Views with Dropped Columns in Release 9.0.1

The columns listed in Table D–2 were dropped in release 9.0.1.

Table D–2 Dynamic Performance Views with Dropped Columns in Release 9.0.1

Dynamic Performance Views	Dropped Columns
GV$CLASS_CACHE_TRANSFER	X_2_SSX X_2_SSX_FORCED_WRITE SS_2_NULL SS_2_RLS OP_2_SS SSX_2_X NULL_2_SS

Table D–2 (Cont.) Dynamic Performance Views with Dropped Columns in Release

Dynamic Performance Views	**Dropped Columns**
GV$FILE_CACHE_TRANSFER	FREQUENCY
	X_2_SSX
	X_2_SSX_FORCED_WRITE
	SS_2_NULL
	SS_2_RLS
	WRB
	WRB_FORCED_WRITE
	CBR
	CBR_FORCED_WRITE
	SSX_2_X
	NULL_2_SS
	OP_2_SS
GV$TEMP_CACHE_TRANSFER	FREQUENCY
	X_2_SSX
	X_2_SSX_FORCED_WRITE
	SS_2_NULL
	SS_2_RLS
	WRB
	WRB_FORCED_WRITE
	RBR_FORCED_STALE
	CBR
	CBR_FORCED_WRITE
	SSX_2_X
	NULL_2_SS
	OP_2_SS
V$CLASS_CACHE_TRANSFER	X_2_SSX
	X_2_SSX_FORCED_WRITE
	SS_2_NULL
	SS_2_RLS
	OP_2_SS
	SSX_2_X
	NULL_2_SS

Table D–2 (Cont.) Dynamic Performance Views with Dropped Columns in Release

Dynamic Performance Views	Dropped Columns
V$FILE_CACHE_TRANSFER	FREQUENCY X_2_SSX X_2_SSX_FORCED_WRITE SS_2_NULL SS_2_RLS WRB WRB_FORCED_WRITE CBR CBR_FORCED_WRITE SSX_2_X NULL_2_SS OP_2_SS
V$TEMP_CACHE_TRANSFER	FREQUENCY X_2_SSX X_2_SSX_FORCED_WRITE SS_2_NULL SS_2_RLS WRB WRB_FORCED_WRITE RBR_FORCED_STALE CBR CBR_FORCED_WRITE SSX_2_X NULL_2_SS OP_2_SS

Dynamic Performance Views Obsolete in Oracle9*i*

The dynamic performance views listed in this section are obsolete in Oracle9*i*.

Dynamic Performance Views Obsolete in Release 9.0.1

The following dynamic performance views became obsolete in release 9.0.1 and are not available in release 9.0.1 and higher:

GV$TARGETRBA	V$TARGETRBA

Changes to Dynamic Performance Views in Version 8

The following sections list the new, changed, and obsolete dynamic performance views in version 8:

- Dynamic Performance Views Added in Version 8
- Dynamic Performance Views with Added Columns in Version 8
- Dynamic Performance Views with Dropped Columns in Version 8
- Dynamic Performance Views with Renamed Columns in Version 8
- Dynamic Performance Views Obsolete in Version 8

Dynamic Performance Views Added in Version 8

The dynamic performance views listed in this section are new in version 8.

Dynamic Performance Views Added in Release 8.0

The following dynamic performance views were added in release 8.0:

GV$ACCESS	GV$ACTIVE_INSTANCES
GV$AQ	GV$ARCHIVE
GV$ARCHIVE_DEST	GV$ARCHIVED_LOG
GV$BACKUP	GV$BACKUP_CORRUPTION
GV$BACKUP_DATAFILE	GV$BACKUP_DEVICE
GV$BACKUP_PIECE	GV$BACKUP_REDOLOG
GV$BACKUP_SET	GV$BGPROCESS
GV$BH	GV$BUFFER_POOL
GV$CACHE	GV$CIRCUIT
GV$CLASS_PING	GV$COMPATIBILITY
GV$COMPATSEG	GV$CONTROLFILE
GV$CONTROLFILE_RECORD_SECTION	GV$COPY_CORRUPTION
GV$CURRENT_BUCKET	GV$DATABASE
GV$DATAFILE	GV$DATAFILE_COPY

GV$DATAFILE_HEADER	GV$DB_OBJECT_CACHE
GV$DB_PIPES	GV$DBFILE
GV$DBLINK	GV$DELETED_OBJECT
GV$DISPATCHER	GV$DISPATCHER_RATE
GV$DLM_CONVERT_LOCAL	GV$DLM_CONVERT_REMOTE
GV$DLM_LATCH	GV$DLM_LOCKS
GV$DLM_MISC	GV$ENABLEDPRIVS
GV$ENQUEUE_LOCK	GV$EVENT_NAME
GV$EXECUTION	GV$FALSE_PING
GV$FILE_PING	GV$FILESTAT
GV$FIXED_TABLE	GV$FIXED_VIEW_DEFINITION
GV$GLOBAL_TRANSACTION	GV$INDEXED_FIXED_COLUMN
GV$INSTANCE	GV$LATCH
GV$LATCH_CHILDREN	GV$LATCH_MISSES
GV$LATCH_PARENT	GV$LATCHHOLDER
GV$LATCHNAME	GV$LIBRARYCACHE
GV$LICENSE	GV$LOADCSTAT
GV$LOADPSTAT	GV$LOADTSTAT
GV$LOCK	GV$LOCK_ACTIVITY
GV$LOCK_ELEMENT	GV$LOCKED_OBJECT
GV$LOCKS_WITH_COLLISIONS	GV$LOG
GV$LOG_HISTORY	GV$LOGFILE
GV$LOGHIST	GV$MLS_PARAMETERS
GV$MTS	GV$MYSTAT
GV$NLS_PARAMETERS	GV$NLS_VALID_VALUES
GV$OBJECT_DEPENDENCY	GV$OFFLINE_RANGE
GV$OPEN_CURSOR	GV$OPTION
GV$PARAMETER	GV$PING
GV$PQ_SESSTAT	GV$PQ_SLAVE
GV$PQ_SYSSTAT	GV$PQ_TQSTAT
GV$PROCESS	GV$PWFILE_USERS

GV$QUEUE	GV$RECENT_BUCKET
GV$RECOVER_FILE	GV$RECOVERY_FILE_STATUS
GV$RECOVERY_LOG	GV$RECOVERY_PROGRESS
GV$RECOVERY_STATUS	GV$REQDIST
GV$RESOURCE	GV$RESOURCE_LIMIT
GV$ROLLSTAT	GV$ROWCACHE
GV$ROWCACHE_PARENT	GV$ROWCACHE_SUBORDINATE
GV$SESS_IO	GV$SESSION
GV$SESSION_CONNECT_INFO	GV$SESSION_CURSOR_CACHE
GV$SESSION_EVENT	GV$SESSION_LONGOPS
GV$SESSION_OBJECT_CACHE	GV$SESSION_WAIT
GV$SESSTAT	GV$SGA
GV$SGASTAT	GV$SHARED_POOL_RESERVED
GV$SHARED_SERVER	GV$SORT_SEGMENT
GV$SORT_USAGE	GV$SQL
GV$SQL_BIND_DATA	GV$SQL_BIND_METADATA
GV$SQL_CURSOR	GV$SQL_SHARED_MEMORY
GV$SQLAREA	GV$SQLTEXT
GV$SQLTEXT_WITH_NEWLINES	GV$STATNAME
GV$SUBCACHE	GV$SYSSTAT
GV$SYSTEM_CURSOR_CACHE	GV$SYSTEM_EVENT
GV$SYSTEM_PARAMETER	GV$TABLESPACE
GV$THREAD	GV$TIMER
GV$TRANSACTION	GV$TRANSACTION_ENQUEUE
GV$TYPE_SIZE	GV$VERSION
GV$WAITSTAT	V$AQ
V$ARCHIVE_DEST	V$ARCHIVED_LOG
V$BACKUP_CORRUPTION	V$BACKUP_DATAFILE
V$BACKUP_DEVICE	V$BACKUP_PIECE
V$BACKUP_REDOLOG	V$BACKUP_SET
V$BUFFER_POOL	V$CLASS_PING

V$CONTROLFILE_RECORD_SECTION	V$COPY_CORRUPTION
V$CURRENT_BUCKET	V$DATAFILE_COPY
V$DATAFILE_HEADER	V$DELETED_OBJECT
V$DISPATCHER_RATE	V$DLM_CONVERT_LOCAL
V$DLM_CONVERT_REMOTE	V$DLM_LATCH
V$DLM_LOCKS	V$DLM_MISC
V$ENQUEUE_LOCK	V$FILE_PING
V$GLOBAL_TRANSACTION	V$LOADPSTAT
V$OFFLINE_RANGE	V$RECENT_BUCKET
V$RECOVERY_PROGRESS	V$RESOURCE_LIMIT
V$ROWCACHE_PARENT	V$ROWCACHE_SUBORDINATE
V$SESSION_LONGOPS	V$SESSION_OBJECT_CACHE
V$SORT_USAGE	V$SUBCACHE
V$TABLESPACE	V$TRANSACTION_ENQUEUE

Dynamic Performance Views Added in Release 8.1

The following dynamic performance views were added in release 8.1:

GV$ARCHIVE_PROCESSES	GV$BACKUP_ASYNC_IO
GV$BACKUP_SYNC_IO	GV$BSP
GV$BUFFER_POOL_STATISTICS	GV$CONTEXT
GV$DB_CACHE_ADVICE	GV$DLM_ALL_LOCKS
GV$DLM_RESS	GV$DLM_TRAFFIC_CONTROLLER
GV$FAST_START_SERVERS	GV$FAST_START_TRANSACTIONS
GV$GLOBAL_BLOCKED_LOCKS	GV$HS_AGENT
GV$HS_PARAMETER	GV$HS_SESSION
GV$INSTANCE_RECOVERY	GV$LOADISTAT
GV$LOGMNR_CONTENTS	GV$LOGMNR_DICTIONARY
GV$LOGMNR_LOGS	GV$LOGMNR_PARAMETERS
GV$OBSOLETE_PARAMETER	GV$PARALLEL_DEGREE_LIMIT_MTH
GV$PROXY_ARCHIVEDLOG	GV$PROXY_DATAFILE

GV$PX_PROCESS	GV$PX_PROCESS_SYSSTAT
GV$PX_SESSION	GV$PX_SESSTAT
GV$RESERVED_WORDS	GV$RSRC_CONSUMER_GROUP
GV$RSRC_CONSUMER_GROUP_CPU_MTH	GV$RSRC_PLAN
GV$RSRC_PLAN_CPU_MTH	GV$SQL_SHARED_CURSOR
GV$TEMP_EXTENT_MAP	GV$TEMP_EXTENT_POOL
GV$TEMP_PING	GV$TEMP_SPACE_HEADER
GV$TEMPFILE	GV$TEMPORARY_LOBS
GV$TEMPSTAT	V$ARCHIVE_PROCESSES
V$BACKUP_ASYNC_IO	V$BACKUP_SYNC_IO
V$BSP	V$BUFFER_POOL_STATISTICS
V$CONTEXT	V$DB_CACHE_ADVICE
V$DLM_ALL_LOCKS	V$DLM_RESS
V$DLM_TRAFFIC_CONTROLLER	V$FAST_START_SERVERS
V$FAST_START_TRANSACTIONS	V$GLOBAL_BLOCKED_LOCKS
V$HS_AGENT	V$HS_PARAMETER
V$HS_SESSION	V$INSTANCE_RECOVERY
V$LOADISTAT	V$LOGMNR_CONTENTS
V$LOGMNR_DICTIONARY	V$LOGMNR_LOGS
V$LOGMNR_PARAMETERS	V$OBSOLETE_PARAMETER
V$PARALLEL_DEGREE_LIMIT_MTH	V$PROXY_ARCHIVEDLOG
V$PROXY_DATAFILE	V$PX_PROCESS
V$PX_PROCESS_SYSSTAT	V$PX_SESSION
V$PX_SESSTAT	V$RESERVED_WORDS
V$RSRC_CONSUMER_GROUP	V$RSRC_CONSUMER_GROUP_CPU_MTH
V$RSRC_PLAN	V$RSRC_PLAN_CPU_MTH
V$SQL_SHARED_CURSOR	V$TEMP_EXTENT_MAP
V$TEMP_EXTENT_POOL	V$TEMP_PING
V$TEMP_SPACE_HEADER	V$TEMPFILE
V$TEMPORARY_LOBS	V$TEMPSTAT

Dynamic Performance Views with Added Columns in Version 8

New columns were added to the dynamic performance views listed in the following sections.

Dynamic Performance Views with Added Columns in Release 8.0

New columns were added to the following dynamic performance views in release 8.0:

V$BH	V$CACHE
V$CACHE_LOCK	V$DATABASE
V$DATAFILE	V$FALSE_PING
V$FILESTAT	V$INSTANCE
V$LATCH_MISSES	V$LOADCSTAT
V$LOADTSTAT	V$LOG_HISTORY
V$PING	V$SESSION
V$SESSION_EVENT	V$SGASTAT
V$SORT_SEGMENT	V$SQL
V$THREAD	V$TRANSACTION

Dynamic Performance Views with Added Columns in Release 8.1

New columns were added to the following dynamic performance views in release 8.1:

GV$ARCHIVE_DEST	GV$BACKUP_PIECE
GV$BH	GV$CIRCUIT
GV$DATABASE	GV$DATAFILE
GV$DISPATCHER	GV$DLM_LATCH
GV$DLM_LOCKS	GV$INSTANCE
GV$LOGMNR_CONTENTS	GV$MTS
GV$PROXY_ARCHIVEDLOG	GV$PROXY_DATAFILE
GV$SESSION	GV$SESSION_LONGOPS
GV$SORT_USAGE	GV$SQL
V$ARCHIVE_DEST	V$BACKUP_PIECE

V$BH	V$CIRCUIT
V$DATABASE	V$DATAFILE
V$DISPATCHER	V$DLM_LATCH
V$DLM_LOCKS	V$INSTANCE
V$LOGMNR_CONTENTS	V$MTS
V$PROXY_ARCHIVEDLOG	V$PROXY_DATAFILE
V$SESSION	V$SESSION_LONGOPS
V$SQL	

Dynamic Performance Views with Dropped Columns in Version 8

The columns listed in the following sections were dropped in version 8. If an application requires one or more of the columns listed below, then modify the application accordingly.

Dynamic Performance Views with Dropped Columns in Release 8.1

The columns listed in Table D–3 were dropped in release 8.1.

Table D–3 Dynamic Performance Views with Dropped Columns in Release 8.1

Dynamic Performance View	Dropped Columns
V$ARCHIVE_DEST	ARCMODE
V$DLM_LATCH	IMM_GETS LATCH_TYPE TTL_GETS
V$DLM_LOCKS	RESOURCE_NAME

Table D–3 (Cont.) Dynamic Performance Views with Dropped Columns in Release 8.1

Dynamic Performance View	Dropped Columns
V$SESSION_LONGOPS	APPLICATION_DATA_1 APPLICATION_DATA_2 APPLICATION_DATA_3 COMPNAM CURRENT_TIME MSG OBJID OPID STEPID STEPSOFAR STEPTOTAL UPDATE_COUNT

Dynamic Performance Views with Renamed Columns in Version 8

The columns listed in the following sections were renamed in version 8. If an application requires one or more of the columns listed below, then modify the application accordingly.

Dynamic Performance Views with Renamed Columns in Release 8.1

The columns listed in Table D–4 were renamed in release 8.1.

Table D–4 Dynamic Performance Views with Renamed Columns in Release 8.1

Dynamic Performance View	Pre-Release 8.1 Column Name	Release 8.1 and Higher Column Name
GV$DISPATCHER_RATE and V$DISPATCHER_RATE	NUM_LOOPS_TRACKED	TTL_LOOPS
	NUM_MSG_TRACKED	TTL_MSG
	NUM_SVR_BUF_TRACKED	TTL_SVR_BUF
	NUM_CLT_BUF_TRACKED	TTL_CLT_BUF
	NUM_BUF_TRACKED	TTL_BUF
	NUM_IN_CONNECT_TRACKED	TTL_IN_CONNECT
	NUM_OUT_CONNECT_TRACKED	TTL_OUT_CONNECT
	NUM_RECONNECT_TRACKED	TTL_RECONNECT

Dynamic Performance Views Obsolete in Version 8

The dynamic performance views listed in this section are obsolete in version 8.

Dynamic Performance Views Obsolete in Release 8.1

The following dynamic performance views became obsolete in release 8.1 and are not available in release 8.1 and higher:

GV$CURRENT_BUCKET	GV$RECENT_BUCKET
V$CURRENT_BUCKET	V$RECENT_BUCKET

E

New Internal Datatypes and SQL Functions

This appendix lists the new internal datatypes and SQL functions added in version 8 and Oracle9*i*. This appendix covers the following topics:

See Also: *Oracle9i SQL Reference* for a complete list and descriptions of Oracle internal datatypes and SQL functions.

Internal Datatypes and SQL Functions Added in Oracle9*i*

The following sections list the new internal datatypes and SQL functions added in Oracle9*i*:

- Internal Datatypes Added in Oracle9i
- SQL Functions Added in Oracle9i

Internal Datatypes Added in Oracle9*i*

The internal datatypes listed in this section were added in Oracle9*i*. Each may be used as a function name in a `SELECT` list, but only if it is qualified with a schema (*schema.function*) as in the following example:

```
select hr.true() ...
```

In release 8.0 and higher, if a schema qualification is missing, then these words generate an error, while in Oracle7, their unqualified use did not generate an error.

Internal Datatypes Added in Release 9.0.1

The following internal datatypes were added in release 9.0.1:

AnyData	AnyDataSet
AnyType	INTERVAL DAY TO SECOND
INTERVAL YEAR TO MONTH	TIMESTAMP
XMLType	

SQL Functions Added in Oracle9*i*

The SQL functions listed in this section are new in Oracle9*i*.

SQL Functions Added in Release 9.0.1

The following SQL functions were added in release 9.0.1:

ASCIISTR	BIN_TO_NUM
COALESCE	COMPOSE
CURRENT_DATE	CURRENT_TIMESTAMP
DBTIMEZONE	DECOMPOSE

EXISTSNODE	EXTRACT
FIRST	FROM_TZ
GROUP_ID	GROUPING_ID
LAST	LOCALTIMESTAMP
NULLIF	PERCENTILE_CONT
PERCENTILE_DISC	RAWTONHEX
ROWIDTONCHAR	SESSIONTIMEZONE
SYS_DBURIGEN	SYS_EXTRACT_UTC
SYS_XMLAGG	SYS_XMLGEN
SYSTIMESTAMP	TO_CHAR
TO_CLOB	TO_DSINTERVAL
TO_NCHAR	TO_NCLOB
TO_TIMESTAMP	TO_TIMESTAMP_TZ
TO_YMINTERVAL	TREAT
TZ_OFFSET	UNISTR
WIDTH_BUCKET	

Internal Datatypes and SQL Functions Added in Version 8

The following sections list the new internal datatypes and SQL functions added in version 8:

- Internal Datatypes Added in Version 8
- SQL Functions Added in Version 8

Internal Datatypes Added in Version 8

The internal datatypes listed in this section were added in version 8. Each may be used as a function name in a SELECT list, but only if it is qualified with a schema (*schema.function*) as in the following example:

```
select hr.true() ...
```

In release 8.0 and higher, if a schema qualification is missing, then these words generate an error, while in Oracle7, their unqualified use did not generate an error.

Internal Datatypes Added in Release 8.0

The following internal datatypes were added in release 8.0:

BFILE	BLOB
CLOB	NCHAR
NCLOB	NVARCHAR2

Internal Datatypes Added in Release 8.1

The following internal datatypes were added in release 8.1:

UROWID (universal rowid)

SQL Functions Added in Version 8

The SQL functions listed in this section are new in version 8.

SQL Functions Added in Release 8.0

The following SQL functions were added in release 8.0:

BFILENAME	DEFEF
EMPTY_BLOB	EMPTY_CLOB
MAKE_REF	NLS_CHARSET_DECL_LEN
NLS_CHARSET_ID	NLS_CHARSET_NAME
REF	REFTOHEX
VALUE	

SQL Functions Added in Release 8.1

The following SQL functions were added in release 8.1:

BITAND	CORR
COVAR_POP	COVAR_SAMP
CUME_DIST	DENSE_RANK

FIRST_VALUE	GROUPING
LAG	LAST_VALUE
LEAD	NTILE
NUMTOYMINTERVAL	NUMTODSINTERVAL
NVL2	PERCENT_RANK
RATIO_TO_REPORT	REGR_ (linear regression) functions
STDDEV_POP	STDDEV_SAMP
SYS_CONTEXT	SYS_GUID
TO_LOB	TRIM
VAR_POP	VAR_SAMP

Note: Function names beginning with SYS are reserved by Oracle. You should not begin any function names with SYS; if you do, then you may encounter compatibility problems.

F

Migration and Compatibility for Oracle Net Services

This appendix describes coexistence, migration, and upgrade issues for Oracle Net Services. This appendix covers the following topics:

Overview of Unsupported Oracle Net Services Features in Release 9.0

In an effort to streamline configuration decisions for the Internet, the following subsections describe the features and the configuration file that are no longer being supported:

- Identix and SecurID Authentication Methods
- NDS External Naming and NDS Authentication
- Net8 OPEN
- protocol.ora File
- Prespawned Dedicated Servers
- SPX Protocol

Identix and SecurID Authentication Methods

If you are using Identix or SecurID authentication, provided by Oracle Advanced Security, Oracle Corporation recommends migrating to one of the following authentication methods:

- CyberSafe
- RADIUS
- Kerberos
- SSL

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

NDS External Naming and NDS Authentication

Support for Novell Directory Services (NDS) as an authentication method and as an external naming method are no longer supported. If you are using NDS as an external naming method, Oracle Corporation recommends using directory naming instead.

Net8 OPEN

Net8 OPEN, which provided an application program interface (API) that enabled programmers to develop both database and non-database applications, is no longer supported.

protocol.ora File

Parameters in the `protocol.ora` file have been merged into the `sqlnet.ora` file. These parameters enable you to configure access control to the database, as well as no delays in TCP/IP buffer flushing. These parameters include:

- `TCP.NODELAY`
- `TCP.EXCLUDED_NODES`
- `TCP.INVITED_NODES`
- `TCP.VALIDNODE_CHECKING`

> **See Also:** *Oracle Net Services Reference Guide* for a description of these parameters

If you have a `protocol.ora` file in `$ORACLE_HOME/network/admin` on UNIX and *`ORACLE_HOME`*`\network\admin` on Windows, Oracle Net Manager, when first started, will automatically merge its parameters into the `sqlnet.ora` file.

There may be operating system specific parameters in `protocol.ora` that are node specific. For this reason, Oracle Corporation recommends not sharing `sqlnet.ora` with other nodes after merging or adding these parameters.

Prespawned Dedicated Servers

Prespawned dedicated server processes are no longer supported. Instead, configure shared server (formerly named multi-threaded server) to improve scalability and system resource usage.

SPX Protocol

Protocol addresses using the SPX protocol must be replaced. Net8 provides support for the following network protocols:

- TCP/IP
- TCP/IP with SSL
- Named Pipes
- LU6.2
- VI

> **See Also:** *Oracle Net Services Reference Guide* for protocol parameter information

Unsupported Parameters

Table F–1 describes the parameters no longer supported.

Table F–1 Unsupported Networking Parameters

Parameter	File	Description	Last Supported Release
`COMMUNITY`	`tnsnames.ora`	The parameter was a required part of all network service addresses. Thus, it appears anywhere you might find an address (for example, local naming and listener configuration files).	8.0
`AUTOMATIC_IPC`	`sqlnet.ora`	This parameter was used to force sessions through IPC addresses. Due to performance issues, this parameter has been removed. You should configure an IPC address instead.	8.0
`NAMES.DEFAULT_ZONE`	`sqlnet.ora`	This parameter was used to be included in profiles as slight variants of the `NAMES.DEFAULT_DOMAIN` and `NAMES.PRFERRED_SERVERS` parameters.	8.0
`NAMES.NDS.NAME.CONTEXT`	`sqlnet.ora`	This parameter was used to configure naming contexts for NDS external naming.	8.1
`OSS.SOURCE_MY_WALLET`	`sqlnet.ora`	This parameter's name has changed to `WALLET_LOCATION`.	8.1

`SQLNET.IDENTIX_FINGERPRINT_DATABASE`
`SQLNET.IDENTIX_FINGERPRINT_DATABASE_USER`
`SQLNET.IDENTIX_FINGERPRINT_DATABASE_PASSWORD`
`SQLNET.IDENTIX_FINGERPRINT_METHOD`
	`sqlnet.ora`	These parameters supported the Identix authentication method.	8.1

`CONNECT_TIMEOUT`	`listener.ora`	This parameter specified the amount of time that the listener waited for a client's request after the transport connection had been established.	8.1

Parameter	File	Description	Last Supported Release
PRESPAWN_DESC PRESPAWN_LIST PRESPAWN_MAX	listener.ora	These parameters supported the prespawned dedicated servers. Prespawned dedicated servers are no longer supported. You should use shared server instead.	8.1
USE_PLUG_AND_PLAY_ *listener_name*	listener.ora	This parameter instructed the listener to register database information with an Oracle Names server during startup. Use the Oracle Names Control utility REGISTER command to register this information.	8.1
NAMES.USE_PLUG_AND_PLAY	names.ora	This parameter was used to enable / disable the Dynamic Discovery Option. There are other mechanisms available to discover other Oracle Names servers.	2.3

See Also: *Oracle Net Services Reference Guide* for further information about supported configuration parameters

Unsupported Control Utility Commands

Table F–2 describes the control utility commands not supported in release 9.0.

Table F–2 Unsupported Network Control Utility Commands

Commands	Control Utility	Description
`DOMAIN_HINT`	Oracle Names Control utility	This command created a domain hint. Configure the `NAMES.DOMAIN_HINTS` parameter in the `names.ora` file.

`DBSNMP_START`
`DBSNMP_STATUS`
`DBSNAMP_STOP`
	Listener Control utility	These commands controlled the Oracle Intelligent Agent for use with Oracle Enterprise Manager. You can now control the Oracle Intelligent Agent through the Oracle Enterprise Manager Console.

`SET CONNECT_TIMEOUT`
`SHOW CONNECT_TIMEOUT`
	Listener Control utility	These commands specified the amount of time that the listener waited for a client's request after the transport connection had been established.

`SET USE_PLUGANDPLAY`
`SHOW USE_PLUGANDPLAY`
	Listener Control utility	These commands instructed the listener to register database information with an Oracle Names server. Use the Oracle Names Control utility `REGISTER` command to register this information.

See Also: *Oracle Net Services Reference Guide* for further information about supported control utility commands

Client and Database Coexistence Issues

Clients and database servers require compatible releases of Oracle Net Services or Net8. For example, an Oracle9*i* client requires an installation of Oracle Net Services, and an Oracle9*i* database requires an installation of Oracle Net Services with the Oracle Net Listener.

Consider the following client-to-database connection issues before you decide if migrating or upgrading is appropriate for your environment:

- Oracle9i Database Connections
- Oracle8 or Oracle7 Database Connections
- Oracle Names

Oracle9*i* Database Connections

Connect descriptors, created for connections to an Oracle9*i* or an Oracle8*i* database, identify a database by its service name with the `SERVICE_NAME` parameter.

A connect descriptor to an Oracle9*i* or an Oracle8*i* database uses the parameter `SERVICE_NAME`, as shown in the following example:

```
sales=
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-sun1)(PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com))
```

Connect descriptors that are currently configured with the `SID` parameter can remain. However, to take advantage of new features, such as client load balancing and connect-time failover, Oracle Corporation recommends replacing `SID` with `SERVICE_NAME`.

To modify a connect descriptor to use `SERVICE_NAME`, use the Oracle Net Manager's compatibility mode, as described in "Using the Oracle Net Manager to Handle Compatibility Issues" on page F-10.

> **See Also:** *Oracle Net Services Administrator's Guide* for information about database identification by `SERVICE_NAME` rather than `SID`

Consider the following questions for an environment with Oracle release 8.0 clients connecting to an Oracle9*i* database:

- *Will my third-party applications be able use features of Oracle Net Services?*

  No. You must rebuild or upgrade applications to work with Oracle Net libraries.

- *Do my clients require Oracle Net to connect to a remote Oracle9i database?*

  No. If a client needs to connect to a *remote* Oracle9*i* database, only Net8 Client release 8.0 needs to be configured on the client. However, new features of Oracle Net Services are not available to these clients.

- *Do my clients require Oracle Net to connect to a local Oracle9i database?*

  No. The client requires an installation of Net8 Client release 8.0 in its Oracle home and the Oracle9*i* requires an installation of Oracle Net and Oracle Net Listener in its Oracle home.

Oracle8 or Oracle7 Database Connections

A connect descriptor to an Oracle release 8.0 or Oracle7 database uses `SID`, as shown in the following example:

```
sales=
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-sun1)(PORT=1521))
  (CONNECT_DATA=
     (SID=sales)))
```

In addition, the `listener.ora` file on the database server must be configured with the description of the `SID` for the Oracle release 8.0 database. In the following example, the listener is configured to listener for a database service called `sales.us.acme.com` that has a SID of `sales`:

```
SID_LIST_listener=
(SID_LIST=
 (SID_DESC=
  (GLOBAL_DBNAME=sales.us.acme.com)
  (SID_NAME=sales)))
```

See Also: *Oracle Net Services Administrator's Guide* for information about database identification by `SID`

Consider the following questions for an environment with Oracle9*i* clients connecting to an Oracle release 8.0 database.

- *Do my clients require Net8 Client release 8.0 to connect to a remote Oracle release 8.0 database?*

  No. If a client needs to connect to a *remote* Oracle release 8.0 database, only Net8 Client of a compatible release needs to be configured on the client. The only limitation is that the new features available with Oracle Net Services are unavailable with this connection type.

- *Do my clients require Net8 Client release 8.0 to connect to a local Oracle release 8.0 database?*

  Yes. The client requires an installation of Oracle Net in its Oracle home and the Oracle release 8.0 database requires an installation of Net8 Server in its Oracle home.

Oracle Names

If you migrate or upgrade all or part of your network to Oracle9*i*, you should migrate or upgrade all the Oracle Names Servers in the region to version 9.

- *Can my Oracle8 release 8.0 clients use Oracle Names version 9 to resolve service names?*

  Yes.

- *Can my Oracle8 release 8.0 then use the connect descriptor returned from Oracle Names version 9 to connect to an Oracle version 8 database?*

  Yes, if the connect descriptor was specified correctly when it was entered into Oracle Names.

Note: In future releases, Oracle Names will not be supported as a centralized naming method. Because no new enhancements are being added to Oracle Names, consider using directory naming or migrating an existing Oracle Names configuration to directory naming, as described in the *Oracle Net Services Administrator's Guide*.

Using the Oracle Net Manager to Handle Compatibility Issues

Because some parameters are enabled only for release 9.0.1 and release 8.1, Oracle Net Manager offers two options that permit you to set the proper parameters in the `tnsnames.ora` file for clients connecting to a release 9.0.1, release 8.1, and release 8.0database. These options are described in Table F–3.

Table F–3 Compatibility Options Available with Oracle Net Manager

Oracle Net Manager Option	Description
Use Options Compatible with Net8 8.0 Clients	Enables you to configure multiple addresses parameters for a client. If selected, enables the `SOURCE_ROUTE` parameter for pre-release 8.1 clients requiring Oracle Connection Manager connections. If turned off, enables you to use the `SOURCE_ROUTE`, `LOAD_BALANCE`, and `FAILOVER` parameters for release 9.0.1 and release 8.1 clients. **See Also:** *Oracle Net Services Administrator's Guide* for information about configuring address list parameters
Use Oracle8 Release 8.0 Compatible Identification	Enables you to configure parameters specific to a database release in the `CONNECT_DATA` section of a connect descriptor. If turned on, allows you to enter the SID of the Oracle release 8.0 or Oracle7 database. If turned off, enables you to enter the Oracle9*i* or Oracle8*i* database service name (`SERVICE_NAME`). **Note:** The *Advanced Service Options* dialog box, which is visible when the Advanced button in the Service Identification group is chosen, is also affected by whether this option is turned on or off. Some settings are only available for connections to an Oracle9*i* or Oracle8*i* database service. **See Also:** *Oracle Net Services Administrator's Guide* for information about configuring advanced connect data parameters

Migrating and Upgrading to Oracle Net Services Release 9.0

To migrate from SQL*Net release 2.x to Oracle Net Services release 9.0.1 or upgrade from Net8 release 8.0 or 8.1, complete these tasks:

- Step 1: Verify Service Name and Instance Name
- Step 2: Perform Software Upgrade or Migration on the Server
- Step 3: Perform Software Migration or Upgrade on the Client
- Step 4: Perform Functional Upgrade and Migration

Step 1: Verify Service Name and Instance Name

If you want to identify a service and its instance in the `tnsnames.ora` file, ensure that the `SERVICE_NAMES` and `INSTANCE_NAMES` initialization parameters are set in the initialization parameter file.

Table F–4 Initialization Parameters for Oracle Net Services

Parameter	Description
`SERVICE_NAMES`	Specifies one or more names for the database service to which this instance connects. You can specify multiple services names in order to distinguish among different uses of the same database. For example: `SERVICE_NAMES = sales.us.acme.com, widgetsales.us.acme.com` If you do not qualify the names in this parameter with a domain, Oracle qualifies them with the value of the `DB_DOMAIN` parameter. If `DB_DOMAIN` is not specified, Oracle uses the domain of your local database as it currently exists in the data dictionary. **Note:** You can change the value of `SERVICE_NAMES` parameter dynamically with the SQL `ALTER SYSTEM` when the database is running. See the *Oracle9i Database Reference* for further information about this parameter
`INSTANCE_NAME`	Specifies the unique name of this instance. Set the instance name to the value of the Oracle System Identifier (SID).

Step 2: Perform Software Upgrade or Migration on the Server

To perform a software upgrade or migration on the database server, install the latest release of Oracle Net and Oracle Net Listener from the Oracle Universal Installer to receive the latest executables.

You are prompted to upgrade or migrate a database with the Oracle Data Migration Assistant if the Oracle Universal Installer detects an pre-release 9.0.1 database on your system. If you do not want to upgrade or migrate during the installation process, you can choose to install this assistant and use it later.

The Oracle Universal Installer automatically performs these tasks:

- Stops older listener
- Starts release 9.0.1 listener

Step 3: Perform Software Migration or Upgrade on the Client

To perform a software migration or upgrade on the client, install the latest release of Net8 Client from the Oracle Universal Installer to receive the latest executables.

Step 4: Perform Functional Upgrade and Migration

After the software is upgraded or migrated, it is *not* required to upgrade the configuration files unless you want to use the Oracle9*i* features. To take advantage of new features, review your configuration files:

- `sqlnet.ora`
- `tnsnames.ora`
- `listener.ora`
- `protocol.ora`

Replace obsolete or renamed parameters.

See Also: "Unsupported Parameters" on page F-4

tnsnames.ora

Replace the `SID` parameter with the `SERVICE_NAME` parameter to connect to a release 8.1 or higher service, as in the following example.

```
sales=
(DESCRIPTION=
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-sun1)(PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)))
```

If you have multiple addresses, you can configure client load balancing and connect-time failover features, as in the following example.

```
sales=
(DESCRIPTION=
 (ADDRESS_LIST=
  (FAILOVER=on)
  (LOAD_BALANCE=on)
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-sun1)(PORT=1521)
  (ADDRESS=(PROTOCOL=tcp)(HOST=sales-sun2)(PORT=1521))
  (CONNECT_DATA=
     (SERVICE_NAME=sales.us.acme.com)))
```

See Also:

- "Using the Oracle Net Manager to Handle Compatibility Issues" on page F-10 for information about configuring the service name and multiple address features.
- *Oracle Net Services Administrator’s Guide* for information about multiple addresses

listener.ora

Because instance information is registered with the listener in release 9.0.1, it is no longer necessary to include the instance information with the `SID_LIST_listener_name` section of the `listener.ora` file.

However, Oracle Enterprise Manager still requires static information in the `listener.ora` file. If you are using Oracle Enterprise Manager to manage database objects, the `listener.ora` file must be configured with information about the database in the following manner:

```
SID_LIST_listener_name=
  (SID_LIST=
     (SID_DESC=
        (GLOBAL_DBNAME=global_database_name)
        (ORACLE_HOME=oracle_home)
        (SID_NAME=sid)))
```

Table F–5 Service Settings in listener.ora

Parameter	Description
`SID_NAME`	The Oracle System Identifier (SID) identifies the instance. You can obtain the SID value from the `INSTANCE_NAME` parameter in the initialization parameter file. **Note:** This setting is required on UNIX and Windows NT.
`GLOBAL_DBNAME`	The global database name is comprised of the database name and database domain name. You can obtain the `GLOBAL_DBNAME` value from the `SERVICE_NAMES` parameter, or from the `DB_NAME` and `DB_DOMAIN` parameters in the initialization parameter file.
`ORACLE_HOME`	Identifies the Oracle home location of the database that you are specifying **Note:** This setting is required on UNIX.

Important: If you are using connect-time failover or Transparent Application Failover, such as in an Oracle9*i* Real Application Clusters environment, Oracle Corporation recommends not setting the `GLOBAL_DBNAME` parameter.

See Also: *Oracle Net Services Administrator's Guide* for information about configuring service information and connect-time failover and Transparent Application Failover (TAF)

Using Oracle Names Version 9

Note: In future releases, Oracle Names will not be supported as a centralized naming method. Because no new enhancements are being added to Oracle Names, consider using directory naming or migrating an existing Oracle Names configuration to directory naming, as described in *Oracle Net Services Administrator's Guide*. The material presented here is primarily for reference to enable you to maintain your current Oracle Names environment.

Oracle Names version 9 is backward compatible with Oracle Names versions 2 and 8. If you wish to take advantage of the new features provided with Oracle Names version 9, you must migrate all of your existing Names Servers in a region to version 9 by installing Oracle Names version 9 on every existing Oracle Names server node.

Migration issues to keep in mind are described in the following sections:

- Migrating from Oracle Names Version 2 Using a Database
- Migrating from Oracle Names Version 2 with the Dynamic Discovery Option
- Migrating from ROSFILES
- Migrating Region Checkpoint Files to Domain and Topology Checkpoint Files
- Reviewing Migration Checklist

Migrating from Oracle Names Version 2 Using a Database

To migrate and transfer data from an existing Oracle Names server version 2 database to a version 9 region database, run the `namesupg.sql` script located in `$ORACLE_HOME/network/admin` on UNIX and `*ORACLE_HOME*\network\admin` on Windows platforms on the node where Oracle Network Manager stored your network definition.

In order to run the `namesupg.sql` script, two tables, `NAMES_DOM` and `NAMES_DID` must be created and populated using values from an existing `names.ora` file.

- The `NAMES_DOM` table needs a `DOMAIN` column with one row per domain specified by the `NAMES.DOMAINS` parameter in the `names.ora` file.
- The `NAMES_DID` table needs the ID which is defined in the `NAME_P` column in the `NMO_INFORMATION` table. The `NAME_P` column is the same as the `DOCNAME` specified by the `NAMES.ADMIN_REGION` parameter in the `names.ora` file.

To migrate data:

1. Create the `NAMES_DOM` table as follows:

   ```
   SQL> CONNECT user/password
   SQL> CREATE TABLE NAMES_DOM (domain varchar(256));
   ```

2. Populate the table with the domain names specified by the `NAMES.DOMAINS` parameter in the `names.ora` file. For example, consider the following `NAMES.DOMAIN` parameter setting:

   ```
   NAMES.DOMAINS=
    (DOMAIN_LIST=
     (DOMAIN=
       (NAME=)
       (MIN_TTL=86400))
     (DOMAIN=
       (NAME=com)
       (MIN_TTL=86400))
     (DOMAIN=
       (NAME=acme.com)
       (MIN_TTL=86400))
   ```

   In this example, three rows for the root domain, `acme` subdomain, and `com` domain must be created as follows:

   ```
   SQL> INSERT into NAMES_DOM values ('(root)');
   SQL> INSERT into NAMES_DOM values ('acme');
   SQL> INSERT into NAMES_DOM values ('acme.com');
   ```

3. Create the NAMES_DID table as follows:

```
SQL> CREATE TABLE NAMES_DID (did number(10))
```

4. Find the DOCNAME value under the NAMES.ADMIN_REGION parameter in the names.ora file. The DOCNAME represents the name associated with the region. In the following example, the DOCNAME is sbox.

```
NAMES.ADMIN_REGION= (REGION=
                      (NAME=local_region.world)
                      (TYPE=rosdb)
                      (USERID=names)
                      (PASSWORD=names)
                      (description=
                        (ADDRESS_LIST=
                          (ADDRESS=
                            (PROTOCOL=tcp)
                            (HOST=nineva)
                            (PORT=1387)))
                        (CONNECT_DATA=(SID=em)))
                      (DOCNAME=sbox)
                      (VERSION=34619392) # 2.1.4
                      (RETRY=60))
```

5. Query the NMO_INFORMATION table for the ID associated with the DOCNAME and insert it into the NAMES.DOM table:

```
SQL> SELECT ID from NMO_INFORMATION where name_P=docname;
SQL> INSERT into NAMES_DID
    select DID from NMO_INFORMATION
       where NAME_p='docname';
```

6. Run the namesupg.sql script:

```
SQL> CONNECT user/password
SQL> @oracle_home/network/admin/namesupg.sql;
```

Migrating from Oracle Names Version 2 with the Dynamic Discovery Option

The procedure to migrate Oracle Names version 2 with the Dynamic Discovery Option is dependent upon whether or not you want Oracle Names version 8 to store information in a region database.

- Non-Region Database Migration
- Region Database Migration

Non-Region Database Migration

If you migrate to an Oracle Names version 8 from Oracle Names version 2 with the Dynamic Discovery Option, the new Oracle Names server should be able to obtain registered data from the old checkpoint files. If data is not registered, you can register objects by completing the procedures in the *Oracle Net Services Administrator's Guide*.

Region Database Migration

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

1. Write the information stored in the Oracle Names version 2 local administrative region to a `tnsnames.ora` file from Oracle Network Manager or run the following from the command line with a version 8 Oracle Names Control utility:

   ```
   NAMESCTL
   NAMESCTL> DUMP_TNSNAMES
   ```

2. Run the `namesini.sql` script located in `$ORACLE_HOME/network/admin` on UNIX and `ORACLE_HOME\network\admin` on Windows platforms on the computer where the database resides.

   ```
   SQL> CONNECT user/password
   SQL> @oracle_home/network/admin/namesini.sql;
   ```

3. Use Oracle Net Manager to configure a `NAMES.ADMIN_REGION` parameter in every Oracle Names server configuration file (`names.ora`).

   See Also: *Oracle Net Services Administrator's Guide* for information about creating an Oracle Names server.

4. Load the `tnsnames.ora` file into a version 9 Oracle Names server using either Oracle Net Manager or Oracle Names Control utility:

Use the Oracle Net Manager...	Use the Oracle Names Control utility...

1. Start the Oracle Net Manager:
-On UNIX, run `netmgr` at `$ORACLE_HOME/bin`.
-On Windows platforms, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager.
2. In the navigator pane, expand Oracle Names Servers.
3. Select the Oracle Names server.
4. From the list in the right pane, select Manage Data.
5. Choose the Net Service Names tab.
6. Choose Load.
7. Enter the path and file name of the Oracle Network Manager-generated `tnsnames.ora` file in the File field created in Step 1.
8. Choose Execute.
9. Choose File > Save Network Configuration.
	From the command line, enter:
	`namesctl`
	`NAMESCTL> LOAD_TNSNAMES` *`file_name`*

Migrating from ROSFILES

Oracle Names version 8 and higher do not support older configurations that use Resource Object Store (ROS) files (ROSFILES). ROSFILES must be migrated directly into Oracle Names database tables or first into a `tnsnames.ora` file and then into Oracle Names. The following sections cover both procedures:

- ROSFILES to Database Tables
- ROSFILES to a tnsnames.ora File

ROSFILES to Database Tables

To migrate ROSFILES to database tables:

1. Create a database user account for Oracle Network Manager:

   ```
   SQL> CONNECT system/password
   SQL> CREATE USER user
       IDENTIFIED BY password
       DEFAULT TABLESPACE users
       TEMPORARY TABLESPACE temp;
   ```

2. To build the necessary tables, the scripts described next must be run against the server. Typically, these scripts are run on the Oracle Network Manager node.

   ```
   SQL> CONNECT username/password
   SQL> @oracle_home\dbs\rosbild.sql;
   SQL> @oracle_home\dbs\nmcbild.sql;
   SQL> @oracle_home\dbs\rosgrnt.sql;
   SQL> @oracle_home\dbs\nmcgrnt.sql;
   ```

Script	Description
`rosbild.sql`	Builds tables for use by the ROS
`nmcbild.sql`	Builds tables for use by the Oracle Network Manager Objects (NMO) components
`rosgrnt.sql`	Grants access to the common tables. You will be prompted for the user name. Use the same user name that was used when you set up the Oracle Network Manager account.
`nmcgrnt.sql`	Grants access to the users who will access the Oracle Network Manager tables

3. From the Oracle Network Manager, save the ROSFILES to a database:

   a. Choose Save As from the File menu.
   b. Select Database in the Save Network Definition dialog box, and then choose OK.
   c. Enter the database user name and password created in Step 1 and a net service name for the database in the Connect dialog box.
   d. Choose OK.
   e. Select or enter the name of the network you wish to save in the Save Network Definition dialog box.
   f. Choose File> Generate to save the network definition and create the Oracle Names tables from the saved definition.
   g. Choose File > Exit to exit the Oracle Network Manager.

4. On the server, create the `NAMES_DID` and `NAMES_DOM` tables and run the `namesupg.sql` script, as described in "Migrating from Oracle Names Version 2 Using a Database" on page F-16.

ROSFILES to a tnsnames.ora File

To migrate ROSFILES to a `tnsnames.ora` file, and then import the `tnsnames.ora` file into Oracle Names:

1. Create a `tnsnames.ora` file from ROSFILES:
   a. From the Oracle Network Manager, choose Special > Preferences.
   b. Ensure Oracle Names is *not* selected in the Preferences dialog box.
   c. Choose File > Generate to update the network definition and create a `tnsnames.ora` file.
   d. Choose File > Exit to exit the Oracle Network Manager.

2. Load the `tnsnames.ora` file into the Oracle Names server using either Oracle Net Manager or Oracle Names Control utility:

Use Oracle Net Manager...	Use the Oracle Names Control utility...

1. Start the Oracle Net Manager.
-On UNIX, run `netmgr` at `$ORACLE_HOME/bin`.
-On Windows platforms, choose Start > Programs > Oracle - *HOME_NAME* > Network Administration > Oracle Net Manager
	From the command line, enter:
	`namesctl`
	`NAMESCTL> LOAD_TNSNAMES` *`file_name`*

2. In the navigator pane, expand Oracle Names Servers.	
3. Select the Oracle Names server.	
4. From the list in the right pane, select Manage Data.	
5. Choose the Net Service Names tab.	
6. Choose Load.	
7. Enter the path and file name of the Oracle Network Manager-generated `tnsnames.ora` file in the File field.	
8. Choose Execute.	
9. Choose File > Save Network Configuration.	

See Also:

- *Oracle Network Manager Administrator's Guide*, release 3.1
- *Oracle Names Administrator's Guide*, version 2

Migrating Region Checkpoint Files to Domain and Topology Checkpoint Files

In release 8.1, the region checkpoint file, `ckpreg.ora`, contained both topology and domain authoritative data. In release 9.0.1, this data has been split into two files. The topology checkpoint files, `ckptop.ora`, defines the domains in the administrative region and the Oracle Names servers authoritative for each domain. The domain checkpoint file, `ckpdom.ora`, contains the authoritative data for each domain.

These files are automatically generated if you are using a region database. If you are not using a region database and instead relying on the data in the checkpoint files, you can either disregard the checkpoint files and rely on Oracle Names servers running in the region or move data from the `ckpreg.ora` file to the `ckptop.ora` file.

To rely on data from other Oracle Names servers:

1. Upgrade the Oracle Names servers to release 9.0.1.
2. For each Oracle Names server, ensure the `.sdns.ora` file is in `$ORACLE_HOME/network/names` on UNIX operating systems or the `sdns.ora` file is in *`ORACLE_HOME`*`\network\names` on Windows.

   This file contains the name and address of the first Oracle Names server. If it does not exist, discover the other Oracle Names server with the Oracle Net Manager's Command > Discover Oracle Names servers command or the Oracle Names Control utility's `REORDER_NS` command.
3. Start the Oracle Names servers.

   When an Oracle Names server starts, it finds another Oracle Names server and downloads the topology and domain data information from it.

To copy or move data from the `ckpreg.ora` file to the `ckptop.ora` file:

1. Upgrade the Oracle Names servers to release 9.0.1.
2. Move the `ckpreg.ora` file to the `ckptop.ora` file. For example:

   ```
   cd network/names
   mv ckpreg.ora ckptop.ora
   ```
3. Start the Oracle Names servers.

   When an Oracle Names server starts, it automatically generates the `ckpdom.ora` file.

   > **See Also:** *Oracle Net Services Administrator's Guide* for an example `cktop.ora` file

Reviewing Migration Checklist

The following checklist is provided to ensure proper migration to Oracle Names version 9.

- ❏ Migrate all Oracle Names servers in each region to the same version 8 release.
- ❏ If you were previously running Oracle Names version 2, and you want to update your database as a repository for your Oracle Names information, run the `namesupg.sql` script on the node where the network definition is stored.
- ❏ If you were previously running Oracle Names version 2 with the Dynamic Discovery Option, and you want to configure a region database as a repository for your Oracle Names information:

  1. Run the `namesini.sql` script on the node where you wish to install the database.
  2. Use the Oracle Net Manager to configure a `NAMES.ADMIN_REGION` parameter in every `names.ora` file. See *Oracle Net Services Reference Guide* for more information about the `NAMES.ADMIN_REGION` parameter.
- ❏ Set up at least two Oracle Names servers in each region to provide for fault tolerance.

G

Migration and Compatibility for Replication Environments

This appendix describes the steps that you must complete to migrate a replication environment from Oracle7 to Oracle9*i*. This appendix covers the following topics:

Note: This appendix addresses migrating from Oracle7 to Oracle9*i*. It does not address upgrading. For more information about upgrading from a previous Oracle version 8 release to the current Oracle9*i* release, see Chapter 7, "Upgrading from an Older Release of Oracle to the New Oracle9i Release".

Migration Overview for Replication

In some cases, you may find it easiest to migrate your replication environment, particularly the multimaster component of your environment, in one step. Typically, this type of migration is only possible for small configurations. If you have a large configuration, then you might consider migrating an existing Oracle7 replication environment to Oracle9*i* incrementally. Replication and administrative operations can be run successfully in a mixed Oracle7, Oracle8, Oracle8*i*, and Oracle9*i* replication environment.

To successfully interoperate, however, you must observe the following restrictions:

- Oracle9*i* materialized view sites can only interact with Oracle7 release 7.3.3 or higher master sites.
- Oracle9*i* master sites can only interact with Oracle7 release 7.3.4 or higher materialized view sites.
- Oracle9*i* master sites can only interact with Oracle7 release 7.3.3 or higher master sites.

Note: In past releases of Oracle, "materialized views" were called "snapshots". The terms are synonymous. In this appendix, "materialized view" is used, even when discussing past releases.

The following migration methods are supported for replication environments:

- Using the Migration utility
- Using the Oracle Data Migration Assistant
- Using full database export from Oracle7 and import to Oracle9*i*

After migrating a master site to Oracle9*i*, perform a complete refresh of all of associated materialized view sites. Downgrading a replication environment from Oracle9*i* to Oracle7 is not supported.

Certain Oracle9*i* replication features require that all sites be successfully migrated to at least Oracle8 release 8.0 before the features can be used. For example, before you can use primary key materialized views, both the materialized view site and its associated master site must be migrated to at least Oracle8 release 8.0. The simple materialized views with subqueries feature and the master table reorganization procedures require that you first upgrade from rowid materialized views to primary key materialized views.

Similarly, certain Oracle8*i* replication features require that all sites be successfully migrated to Oracle8*i* or higher before the features can be used, and certain Oracle9*i* replication features require that all sites be successfully migrated to Oracle9*i* before the features can be used. For example, to replicate objects based on user-defined types, all sites must be Oracle9*i*. These features are listed in "Features Requiring Migration to a Higher Release of Oracle" on page G-20.

See Also: Consult the following documentation for information about Oracle Replication:

- For conceptual information about Oracle Replication, see *Oracle9i Replication*. This book also contains information about new features in each major release of Oracle from release 8.0 to Oracle9*i*.
- For information about how to complete the steps described in this appendix using the Replication Management tool in Oracle Enterprise Manager, see the Replication Management tool online help.
- For information about how to complete the steps described in this appendix using the replication management API, see the *Oracle9i Replication Management API Reference.*

Migrating All Sites at Once

This section describes migrating all master sites in your multimaster environment to Oracle9*i* at once. Any materialized view sites that you do not also migrate to Oracle9*i* must be upgraded to Oracle7 release 7.3.4 or higher. If you want to migrate your sites incrementally instead, see "Migrating Incrementally" on page G-7.

Complete the following steps to migrate all master sites and (optionally) materialized view sites at one once:

1. Stop all propagation and refreshing from materialized view sites to all masters that you are migrating. You can do this, for example, by temporarily suspending or "breaking" entries in the job queue that control automated propagation and refreshing at the materialized view sites. You can use the `DBMS_JOB.BROKEN` procedure to break a job.

   All deferred transactions at the materialized view sites must be pushed before the upgrade of the master site begins.

   > **See Also:** The following sections in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*:
   >
   > - Chapter 4, "Asynchronously Propagating DML Changes Among Master Sites"
   > - Chapter 4, "Replication Administration Usage Notes"

2. Resolve and re-execute any errors in the local error queue at each master site until it is empty.

   > **See Also:** The following section in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 7, "Manually Resolving an Error"

3. Quiesce the replication environment by executing `SUSPEND_MASTER_ACTIVITY` procedure in the `DBMS_REPCAT` package at the master definition site for all master replication groups.

   > **See Also:** The following section in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 4, "Suspending Replication Activity"

4. Migrate all master sites using the Oracle7 to Oracle9*i* Migration utility or Oracle Data Migration Assistant, as documented in Chapter 4 and Chapter 5.

   Alternatively, you can use export/import to migrate each database. To export a full database from Oracle7 release 7.3.3 or higher and import to Oracle9*i*, complete these steps:

   a. Export the Oracle7 release 7.3.3 or higher database to a dump file using the release 7.3 Export utility under the `SYSTEM` schema with `FULL=y`.

   b. Import the dump file to the Oracle9*i* database using the Oracle9*i* Import utility under the `SYSTEM` schema with `FULL=y`.

   You may also export data from individual Oracle7 tables, import the data to Oracle9*i* tables, and then configure those tables as masters in an Oracle9*i* replication environment using standard replication procedures.

   If you use export/import, then you may need to drop and recreate the materialized views that are based on the master tables.

   See Also:

   - Chapter 6, "Migrating Using Export/Import"
   - *Oracle9i Database Utilities* for general information about performing an export/import

5. Using the Replication Management tool Setup Wizard or setup scripts, set up the multimaster replication environment:

   a. Create a primary master replication administrator account and register this user as the replication administrator, propagator, and receiver on all master sites.

   b. Set up the appropriate database links connecting all sites.

6. Using Replication Management tool or the replication management API, regenerate replication support for each replication base object. If you use the replication management API, then run the `GENERATE_REPLICATION_SUPPORT` procedure in the `DBMS_REPCAT` package. Among other activities, generating replication support establishes the registered propagator as the owner of generated objects.

7. Using Replication Management tool or the replication management API, resume replication activity for the replication environment. If you use the replication management API, then run the `RESUME_MASTER_ACTIVITY` procedure in the `DBMS_REPCAT` package.
8. You must now upgrade all associated materialized view sites to Oracle7 release 7.3.4 or migrate these sites to Oracle9*i*. Migrating these materialized view sites to Oracle9*i* is preferable.

   See Also:

   - Your Oracle7 documentation for information about upgrading your materialized view sites to release 7.3.4
   - "Incremental Migration of Materialized View Sites" on page G-9 for instructions on migrating your materialized view sites to Oracle9*i*

9. Perform a complete refresh on all materialized views at all materialized view sites that have master sites migrated to Oracle9*i*. Before the refresh, make sure you have “unbroken” any jobs that you may have “broken” during migration of your materialized view sites by calling the `DBMS_JOB.BROKEN` procedure.

   If your materialized views have been defined with the `REFRESH FORCE` option, then their next attempted refresh will be a complete refresh automatically. Materialized views defined with the `REFRESH FAST` option must be manually refreshed using the `DBMS_REFRESH.REFRESH` procedure or other refresh procedures.

   If you are using procedural replication at your master sites that is initiated at materialized view sites, then regenerate materialized view support on all packages and package bodies used for procedural replication.

   Note: If you are able to migrate all of a master’s materialized view sites to Oracle9*i* when the master site is migrated to Oracle9*i* (that is, you do not need to migrate the materialized view sites incrementally), then you can alternatively drop the materialized view logs for the master and recreate them as primary key materialized view logs. The materialized views at each materialized view site should be altered to convert them to primary key materialized views. You can then do a complete refresh for each primary key materialized view. See "Upgrading to Primary Key Materialized Views" on page G-17 for additional details.

10. Drop any administrative accounts and database links that you were using to maintain your Oracle7 multimaster replication environment that are not needed in your Oracle9*i* environment. Unnecessary privileges may also be revoked. Be careful not to drop accounts that are needed to maintain any Oracle7 materialized view sites.

Migrating Incrementally

It is possible to incrementally migrate your replication environment. However, you must carefully analyze the interdependencies between sites to ensure that they can continue to interoperate throughout your migration. Table G–1 describes the conditions that must be met to allow Oracle7 and Oracle9*i* replication sites to interoperate.

Table G–1 Interoperability in a Replication Environment

Environment	Action	Condition
Multimaster	Migrate master site from Oracle7 to Oracle9*i*.	All other master sites must be Oracle7 release 7.3.3 or higher.
Master with dependent materialized views	Migrate master site from Oracle7 to Oracle9*i*.	All dependent materialized view sites must be Oracle7 release 7.3.4 or higher.
Master with dependent materialized views	Migrate materialized view site from Oracle7 to Oracle9*i*.	Associated master sites must be Oracle7 release 7.3.3 or higher.

To avoid interoperability problems within a replication environment, Oracle Corporation strongly recommends that, if you must perform an incremental migration, you perform it in the following order:

1. Upgrade all of your master sites to Oracle7 release 7.3.3 or higher and complete the steps in "Preparing Oracle7 Master Sites for Incremental Migration" on page G-8 to prepare your Oracle7 master sites for incremental migration.
2. Incrementally migrate all materialized view sites to Oracle9*i* by completing the steps in "Incremental Migration of Materialized View Sites" on page G-9.
3. Incrementally migrate all master sites to Oracle9*i* by completing the steps in "Incremental Migration of Master Sites" on page G-12.

See Also: Your Oracle7 documentation for information about upgrading your Oracle7 sites to release 7.3.3 or higher

Preparing Oracle7 Master Sites for Incremental Migration

Before beginning incremental migration of Oracle7 master or materialized view sites, your Oracle7 release 7.3.3 or higher master sites must be configured so that all replication administration and propagation is done within the security context of a single user at each site. Additionally, this primary master replication administrator must have the same username and password at all Oracle7 and Oracle9*i* sites.

Your Oracle7 master sites may already be configured in this manner. If not, then you must complete the following steps:

1. Choose a primary master replication administrator for your replication environment. You may select your current replication administrator or create a new user.
2. At each master site, grant the required privileges to the primary master replication administrator using both `DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_REPGROUP` and `DBMS_REPCAT_AUTH.GRANT_SURROGATE_REPCAT`.
3. If they do not already exist, then you must create the following database links from each master site to all other master sites in the multimaster environment:
   - A public database link, created as `SYS`, that includes a valid global database name, as well as a `USING` clause with a valid SQL*Net 2.3 TNS alias.
   - A private database link, created as `SYS`, that includes a valid global database name, as well as a `CONNECT TO` clause with the username and password of the primary master replication administrator.
   - A private database link, created by the primary master replication administrator, that includes a valid global database name, as well as a `CONNECT TO` clause with the username and password of the primary master replication administrator.

Incremental Migration of Materialized View Sites

Before you can migrate a materialized view site to Oracle9*i*, its associated master site must have been upgraded to Oracle7 release 7.3.3 or higher and the master site must have been fully prepared for incremental migration.

See Also:

- Your Oracle7 documentation for information about upgrading your Oracle7 sites to release 7.3.3 or higher
- "Preparing Oracle7 Master Sites for Incremental Migration" on page G-8 for information about preparing your master sites

To incrementally migrate your Oracle7 materialized view sites to Oracle9*i*, complete the following steps at each materialized view site:

1. Isolate the materialized view site from the replication environment by completing the following steps:

   a. Stop all local updates to updatable materialized views at the materialized view site.

   b. In a separate session, lock each materialized view's base table to prevent further transactions.

   c. Empty the local deferred transaction queue by pushing the queue to the materialized view site's master. See the following section in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 5, "When Changes Are Propagated".

   d. Stop all propagation from the materialized view site to its master, for example, by temporarily suspending or "breaking" entries in the job queue that control automated propagation and refreshing of the materialized views at the materialized view site. You can use the `DBMS_JOB.BROKEN` procedure to break a job. See the following section in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 4, "Replication Administration Usage Notes".

2. Run the Oracle7 to Oracle9*i* Migration utility or Oracle Data Migration Assistant, as documented in Chapter 4 and Chapter 5.

   Alternatively, you can use export/import to migrate the materialized view database. To export a full database from Oracle7 release 7.3.3 or higher and import to Oracle9*i*, complete these steps:

   a. Export the Oracle7 release 7.3.3 or higher database to a dump file using the release 7.3 Export utility under the `SYSTEM` schema with `FULL=y`.

   b. Import the dump file to the Oracle9*i* database using the Oracle9*i* Import utility under the `SYSTEM` schema with `FULL=y`.

   You may also export data from individual Oracle7 tables, import the data to Oracle9*i* tables, and then configure those tables as masters in an Oracle9*i* replication environment using standard replication procedures.

   See Also:

   - Chapter 6, "Migrating Using Export/Import"
   - *Oracle9i Database Utilities* for general information about performing an export/import

3. Use the Replication Management tool Setup Wizard or execute the appropriate replication management API calls to complete the following actions:

   - Register the primary materialized view replication administrator as the replication administrator and propagator for the materialized view site. If you are using the replication management API, then use the `REGISTER_PROPAGATOR` procedure in the `DBMS_DEFER_SYS` package.
   - Register a receiver account at the associated master site. For materialized views sites with Oracle7 master sites, your receiver at the master site must be the primary master replication administrator that you prepared in the previous section. If you are using the Replication Management tool Setup Wizard, then select the customize option to specify this receiver. If you are using the replication management API, then use the `REGISTER_USER_REPGROUP` procedure in the `DBMS_REPCAT_ADMIN` package.

4. Create the appropriate database links from the materialized view site to the master site.

   Specifically, you should create a `PUBLIC` database link from the materialized view site to the master site; doing so makes defining your private database links easier because you do not need to include the `USING` clause in each link. You also need private database links from the materialized view administrator to the proxy administrator at the master site and from the propagator to the receiver at the master site.

5. Use the Replication Management tool or the appropriate replication management API calls to regenerate materialized view replication support. If you use the replication management API, then run the `GENERATE_MVIEW_SUPPORT` procedure in the `DBMS_REPCAT` package. Among other activities, generating replication support establishes the registered propagator as the owner of generated objects.

6. Use the Replication Management tool or the appropriate replication management API calls to reschedule propagation and / or refresh intervals with the master and enable local updates where appropriate. If you use the replication management API, then run the `SCHEDULE_PUSH` procedure in the `DBMS_DEFER_SYS` package to set the propagation schedule and the `MAKE` procedure in the `DBMS_REFRESH` package to set the refresh interval for a refresh group.

7. If you used the `DBMS_JOB.BROKEN` procedure to help isolate your master site in Step 1, then you must “unbreak” your jobs to resume your replication activity from your materialized view sites.

8. Drop any administrative accounts and links that you were using to maintain your Oracle7 replication environment that are not needed in your Oracle9*i* environment. Unnecessary privileges may also be revoked.

9. Complete all of the steps in this procedure for your other materialized view sites that have not yet been migrated, according to your schedule.

Incremental Migration of Master Sites

Before migrating a master site from Oracle7 to Oracle9*i*, you must meet the following conditions:

- All other master sites in a multimaster environment must be running Oracle7 release 7.3.3 or higher.
- You must have completed the instructions in "Preparing Oracle7 Master Sites for Incremental Migration" on page G-8.
- Any dependent materialized view sites must be running Oracle7 release 7.3.4 or higher.

> **See Also:** Your Oracle7 documentation for information about upgrading your Oracle7 sites

To incrementally migrate your Oracle7 master sites to Oracle9*i*, complete the following steps:

1. Pick a master site to migrate. You should migrate your master definition site first.
2. If you are using procedural replication, then record the configuration information and locations (schemas) of existing procedure wrappers. This information will be used later.
3. Isolate the master site from the replication environment. To do this, complete the following steps:

   a. Stop updates to the master site by either calling `DBMS_REPCAT.SUSPEND_MASTER_ACTIVITY` at the master definition site for all master replication groups, or by calling `DBMS_DEFER_SYS.UNSCHEDULE_EXECUTION` (for Oracle7 sites) or `DBMS_DEFER_SYS.UNSCHEDULE_PUSH` (for Oracle8 and higher sites) at every remote master site and dependent materialized view site. You should also refrain from executing any administrative operations at the master definition site that may affect the master site being migrated.

   > **See Also:** The following sections in *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*:
   >
   > - Chapter 4, “Suspending Replication Activity”
   > - Chapter 4, “Removing a Master Site from the Deferred Execution List”

b. Prevent DML activity at the master site being migrated.

   See Also: The following section in *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 4, “Asynchronously Propagating DML Changes Among Master Sites”

c. Empty the local deferred transaction queue by manually pushing the queue to all sites.

   See Also: The following section in *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 4, “Forcing Execution of the Deferred Transaction Queue”

d. Resolve and re-execute any errors in the local error queue until it is empty.

   See Also: The following section in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 7, “Manually Resolving an Error”

e. Stop any refreshes of the dependent materialized view sites from occurring by “breaking” entries in the job queue at each materialized view site that control automated propagation and refreshing at the materialized view site. You can use the `DBMS_JOB.BROKEN` procedure to break a job.

   See Also: The following section in the *Oracle7 Server Distributed Systems Manual, Volume II: Replicated Data*: Chapter 4, “Replication Administration Usage Notes”

4. Migrate the master site using the Oracle7 to Oracle9*i* Migration utility or Oracle Data Migration Assistant, as documented in Chapter 4 and Chapter 5.

   Alternatively, you can use export/import to migrate the database. To export a full database from Oracle7 release 7.3.3 or higher and import to Oracle9*i*, follow these steps:

   a. Export the Oracle7 release 7.3.3 or higher database to a dump file using the release 7.3 Export utility under the `SYSTEM` schema with `FULL=y`.

   b. Import the dump file to the Oracle9*i* database using the Oracle9*i* Import utility under the `SYSTEM` schema with `FULL=y`.

   You may also export data from individual Oracle7 tables, import the data to Oracle9*i* tables, and then configure those tables as masters in an Oracle9*i* replication environment using standard replication procedures.

If you use export/import, then you may need to drop and recreate the materialized views that are based on the master tables.

See Also:

- Chapter 6, "Migrating Using Export/Import"
- *Oracle9i Database Utilities* for general information about performing an export/import

5. Use the Replication Management tool Setup Wizard or the replication management API to register your primary master replication administrator as the replication administrator, propagator, and receiver for the master site.

   Database links from the primary master replication administrator to the primary master replication administrator at all other Oracle7 and Oracle9*i* master sites should already exist if you prepared your Oracle 7 master site for compatibility with Oracle9*i* using the directions in "Preparing Oracle7 Master Sites for Incremental Migration" on page G-8.

6. If you are not already in a quiesced state, then use Replication Management tool or the replication management API to suspend all replication activity for all master groups. If you use the replication management API, then run the `SUSPEND_MASTER_ACTIVITY` procedure in the `DBMS_REPCAT` package at the master definition site for all master groups.

7. Use the Replication Management tool or the replication management API to regenerate replication support for each replicated object.

   If any sites in the replication environment are still running Oracle7, then you must set the `min_communication` parameter to `false` when generating replication support. The `min_communication` parameter should only be set to `true` (the default) when all sites have been migrated to Oracle9*i* (or in a mixed environment with Oracle8 and higher sites). If you use the replication management API, then run the `GENERATE_REPLICATION_SUPPORT` procedure in the `DBMS_REPCAT` package. Among other activities, generating replication support establishes the registered propagator as the owner of generated objects.

   See Also: *Oracle9i Replication* for more information minimum communication

8. If you are using procedural replication, then check your remaining Oracle7 master sites to determine whether the wrappers have been moved (you created a list of wrappers in Step 2). If they have been moved, then create a synonym in their old location (in the schema of either the replication administrator or the table owner, depending on whether the site previously used the system-based or user-based model) pointing to the new location in the schema of the primary replication administrator. Confirm that necessary object privileges have been granted to access the new owner and locations. If you are also using procedural replication that is initiated at materialized view sites, then regenerate materialized view support on all packages and package bodies used for procedural replication at these materialized view sites.
9. If you have isolated the master by unscheduling propagation to other masters and from other masters, then reschedule propagation by executing `DBMS_DEFER_SYS.SCHEDULE_EXECUTION` (for Oracle7 sites) or `DBMS_DEFER_SYS.SCHEDULE_PUSH` (for Oracle8 and higher sites) at all master sites.
10. Use the Replication Management tool or the replication management API to resume replication activity for each master group. If you use the replication management API, then run the `RESUME_MASTER_ACTIVITY` procedure in the `DBMS_REPCAT` package.
11. Perform a complete refresh on all materialized views after their master site has been migrated to Oracle9*i*. Because of the new rowid format introduced in Oracle8, the Oracle7 to Oracle9*i* Migration utility and Oracle Data Migration Assistant both truncate all the materialized view logs of master tables.

12. If you used the DBMS_JOB.BROKEN procedure to help isolate your master site in Step 3, then “unbreak” your jobs to resume your replication activity from your materialized view sites.

    If your materialized views have been defined with the REFRESH FORCE option, then their next attempted refresh will be a complete refresh automatically. Materialized views defined with the REFRESH FAST option must be manually refreshed using the DBMS_REFRESH.REFRESH procedure or other refresh procedures.

    Note: If you are able to migrate all of the master’s materialized view sites to Oracle9*i* when the master site is migrated to Oracle9*i* (that is, you do not need to migrate the materialized view sites incrementally), then you can alternatively drop the materialized view logs for the master and recreate them as primary key materialized view logs. The materialized views at each materialized view site should be altered to convert them to primary key materialized views. You can then do a complete refresh for each primary key materialized view. See "Upgrading to Primary Key Materialized Views" on page G-17 for additional details.

13. Drop any administrative accounts and links that you were using to maintain your Oracle7 multimaster replication environment that are not needed in your Oracle9*i* environment. Unnecessary privileges may also be revoked. Be careful not to drop accounts that are needed to maintain any Oracle7 materialized view sites or master sites.

14. Complete all of the steps in this procedure for your other master sites that have not yet been migrated, according to your schedule.

Upgrading to Primary Key Materialized Views

When a materialized view site and its master have been migrated to Oracle9*i*, you can upgrade your rowid materialized views to Oracle9*i* primary key materialized views. To do this, you must first alter the materialized view logs for each master table to log primary key information, as well as rowid information, when DML is performed on the master. When this is completed at your master sites, you can incrementally convert your Oracle9*i* materialized view sites by altering the materialized views to convert them to primary key materialized views. Oracle9*i* masters that have been altered to log primary key as well as rowid information can support Oracle7 rowid materialized views as well as Oracle9*i* rowid and primary key materialized views simultaneously to allow for incremental migration.

Note: A primary key materialized view cannot be converted or downgraded to a rowid materialized view.

Primary Key Materialized View Conversion at Master Sites

To support primary key materialized views, complete the following steps at the Oracle9*i* master site:

1. Define and enable a primary key constraint on each master table that does not already have a primary key constraint enabled.
2. Alter the materialized view log for each master table supporting fast refresh to include primary key information using the `ALTER MATERIALIZED VIEW LOG` statement.

   For example, the following statement alters an existing rowid materialized view log to also record primary key information:

   ```
   ALTER MATERIALIZED VIEW LOG ON hr.employees
      ADD PRIMARY KEY;
   ```

   See Also: `ALTER MATERIALIZED VIEW LOG` in the *Oracle9i SQL Reference* for additional information

Note: If you do not complete Steps 1 and 2, then an error is raised when you execute the `ALTER MATERIALIZED VIEW` statement at the materialized view sites to convert to primary key materialized views.

Primary Key Materialized View Conversion at Materialized View Sites

After the Oracle9*i* master sites have been configured to support primary key materialized views, complete the following steps at the Oracle9*i* materialized view sites:

1. Isolate the materialized view site from the replication environment by completing the following steps:

   a. Stop all local updates to updatable materialized views at the materialized view site.

   b. Empty the local deferred transaction queue by pushing the queue to the materialized view site's master.You can use the `DBMS_DEFER_SYS.PUSH` procedure to push the deferred transactions. See the *Oracle9i Replication Management API Reference* for more information.

   c. Stop all propagation from the materialized view site to its master by, for example, temporarily suspending or "breaking" entries in the job queue that control automated propagation and refreshing of the materialized views at the materialized view site. You can use the `DBMS_JOB.BROKEN` procedure to break a job. See the *Oracle9i Supplied PL/SQL Packages and Types Reference* for more information.

2. If you are converting any read-only rowid materialized views to primary key materialized views and these rowid materialized views do not include all the columns of the primary key, then drop and recreate the read-only materialized views with all the primary key columns.

   See Also: *Oracle9i Replication* for more information on rowid materialized views

3. Perform a fast refresh of all materialized views that can be fast refreshed to remove the need for any remaining rowid references in the master materialized view log.

4. Use the ALTER MATERIALIZED VIEW statement to convert rowid materialized views to primary key materialized views.

   For example, the following statement changes a rowid materialized view to a primary key materialized view:

   ```
   ALTER MATERIALIZED VIEW hr.employees_mv
      REFRESH WITH PRIMARY KEY;
   ```

   See Also: *Oracle9i SQL Reference* for the complete syntax of ALTER MATERIALIZED VIEW

5. Resume replication by rescheduling propagation and/or materialized view refresh with the master and enabling local updates where appropriate. If you use the replication management API, then run the SCHEDULE_PUSH procedure in the DBMS_DEFER_SYS package to set the propagation schedule and the MAKE procedure in the DBMS_REFRESH package to set the refresh interval for a refresh group.

6. If you used the DBMS_JOB.BROKEN procedure to help isolate you master site in Step 1, then you need to “unbreak” your jobs to resume your replication activity from your materialized view sites.

Features Requiring Migration to a Higher Release of Oracle

Oracle adds new features to each major release of the Oracle server. The following sections list the features that can only be used if you migrate (or upgrade) your database to a higher release of Oracle.

> **See Also:** The "What's New in Replication" section of the *Oracle9i Replication* for more information about these new replication features

Features Requiring Oracle9*i*

All replication sites involved must be running Oracle9*i* to use the following features:

- Add new master sites without quiescing the master group
- Add new columns to a master table without quiescing its master group
- Alter a master table by making a safe change to it in a single master environment without quiescing the master group
- Replication of user-defined types and the objects on which they are based
- Multi-tier updatable materialized views
- Row-level dependency tracking for parallel propagation
- Replication of tables using CHAR column length semantics or Unicode
- Fast refresh of the following types of materialized views:
  - Materialized views with one to many subqueries
  - Materialized views with many to many subqueries
  - Materialized views with unions

Features Requiring Oracle8*i* or Higher

Master sites must be running Oracle8*i* release 8.1.7 or higher to use the following feature:

- Extended availability for single master replication environments. This feature reduces the number of administration operations that require you to quiesce a master group in a single master replication environment. A complete list of these operations is in the "What's New in Replication" section of the *Oracle9i Replication*.

All replication sites involved must be running Oracle8*i* release 8.1.5 or higher to use the following features:

- Instantiation of materialized view sites using deployment templates
- Parameterized materialized view deployment templates
- Column subsetting of updatable materialized views

Features Requiring Oracle8 or Higher

All replication sites involved must be running Oracle8 or higher to use the following features:

- Parallel propagation of deferred transactions
- Reduced data propagation:
  - Use of the `min_communication` parameter in various procedures in the `DBMS_REPCAT` package and the `DBMS_OFFLINE_SNAPSHOT` package
  - Use of the `SEND_OLD_VALUES` and `COMPARE_OLD_VALUES` procedures in the `DBMS_REPCAT` package
- Data subsetting by creating simple materialized views with subqueries
- Replication of LOB data types
- Primary key materialized views
- Global authentication and privileged database links
- Use of the `VALIDATE` function in the `DBMS_REPCAT` package
- Reorganizing tables with capability of fast refresh
- Replication of partitioned tables and indexes

Features That Work with Oracle7 and Higher Releases

The following features work automatically environments where some sites are running Oracle7 while other sites are running Oracle8 and higher, but these features only apply to the Oracle8 and higher sites:

- Fine-grained quiesce
- Materialized view registration at master sites

Note: All master groups at Oracle7 sites are quiesced if any master group at that site is quiesced.

Note: Oracle7 materialized views are not registered automatically at Oracle9*i* sites but can be manually registered using the `DBMS_MVIEW.REGISTER_MVIEW` procedure at the master site. See *Oracle9i Replication Management API Reference* for more information about using this procedure.

Obsolete Procedures

The following replication management API procedures are obsoleted in Oracle8 and higher releases:

- `DBMS_REPCAT.GENERATE_REPLICATION_PACKAGE`
- `DBMS_REPCAT.GENERATE_REPLICATION_TRIGGER`
- `DBMS_REPCAT_ADMIN.GRANT_ADMIN_REPGROUP`
- `DBMS_REPCAT_ADMIN.GRANT_ADMIN_ANY_REPGROUP`
- `DBMS_REPCAT_ADMIN.REVOKE_ADMIN_REPGROUP`
- `DBMS_REPCAT_ADMIN.REVOKE_ADMIN_ANY_REPGROUP`
- `DBMS_REPCAT_AUTH.GRANT_SURROGATE_REPCAT`
- `DBMS_REPCAT_AUTH.REVOKE_SURROGATE_REPCAT`
- `DBMS_DEFER_SYS.EXECUTE`
- `DBMS_DEFER_SYS.SCHEDULE_EXECUTION`

Index